Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія розпізнавання порід котів на основі згорткової нейронної мережі»

Виконав: студент 2-го курсу, групи 1КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напрямку підготовки, спеціальності)
_____ Кузик Я. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
_____ Паночишин Ю.М.
(прізвище та ініціали)
«07» 12 2023 р.

Опонент: к.т.н., професор каф. КСУ
_____ Биков М. М.
(прізвище та ініціали)
«07» 12 2023 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«08» 12 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.
29.08 2023 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Кузику Ярославу Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія розпізнавання порід котів на основі згорткової нейронної мережі

керівник роботи_ к.т.н., доцент кафедри КН Паночишин Ю.М
затверджені наказом вищого навчального закладу від "18"вересня 2023 року № 247
2. Строк подання студентом роботи 13.11 2023 року
3. Вихідні дані до роботи:
Вхідні дані – кількість навчальних образів – не менше 1000, кількість категорій порід – не менше 5, формат вхідних зображень – .jpeg, вид класифікатора – нейронна мережа, використання об'єктно-орієнтованої мови програмування.

4. Зміст текстової частини:
Вступ, аналіз предметної області розпізнавання порід котів, розробка інформаційної технології розпізнавання порід котів, програмна реалізація інформаційної технології розпізнавання порід котів, тестування та аналіз результатів роботи програми розпізнавання порід котів на основі згорткової нейромережі, економічна частина, висновки, список використаних джерел, додатки

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
Алгоритм роботи програми розпізнавання порід котів, структура інформаційної технології розпізнавання порід котів на основі згорткової нейронної мережі, структура згорткової нейронної мережі SqueezeNet, UML діаграма класів програми розпізнавання порід котів, робочі вікна програми розпізнавання порід котів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціалита посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Паночишин Ю.М., к.т.н., доц. каф. КН	29.08.23	10.11.23
5	Адлер О. О., к.т.н., доц. каф. ЕПВМ	21.10.23	29.10.23

7. Дата видачі завдання 29.08 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасного рівня інформаційних технологій розпізнавання порід котів. Постановка задач дослідження	01.09.23 - 07.09.23	
2	Побудова моделей розпізнавання порід котів на основі нейромереже та функціонування нейронної мережі	08.09.23 - 15.10.23	
3	Практичне застосування та оцінка ефективності розроблених моделей	16.10.23 - 20.10.23	
4	Підготовка економічної частини	21.10.23 - 29.10.23	
5	Апробація та/або впровадження результатів дослідження	30.10.23 - 05.11.23	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	06.11.23 - 10.11.23	

Студент _____ (підпис) Кузик Я. О.

Керівник роботи _____ (підпис) Паночишин Ю.М.

АНОТАЦІЯ

УДК 004.8

Кузик Я. О. Інформаційна технологія розпізнавання порід котів на основі згорткової нейронної мережі. Магістерська кваліфікаційна робота зі спеціальності 122 – «Комп'ютерні науки», освітня програма – «Системи штучного інтелекту». Вінниця: ВНТУ, 2023. 91 с.

На укр. мові. Бібліогр.: 22 назв; рис.: 25; табл. 7.

У даній магістерській кваліфікаційній роботі обґрунтовано вибір згорткової нейронної мережі SqueezeNet v.1.1 для розпізнавання порід котів, яка приймає вхідне зображення розміром 224x224x3 пікселів та може розпізнавати 35 порід котів. Програмну реалізацію інформаційної технології розпізнавання порід котів створено на мові програмування Python у програмному середовищі Raspberry PI. Навчання згорткової нейронної мережі відбувалось з використанням бази даних зображень ImageNet. Навчальна вибірка складалась із 2800 зображень (по 80 зображень на кожну із 35 порід котів). Тестова вибірка складалась із 700 зображень (по 20 зображень на кожну із 35 порід котів). Розроблена програма має достовірність розпізнавання порід котів 94,3%, а програма-аналог Cat Scanner має достовірність розпізнавання порід котів 89,9%.

Графічна частина складається із 6 плакатів.

У економічному розділі доведено доцільність проведення науково-дослідної роботи за даною темою. Термін окупності становить 0,8 р., що свідчить про комерційну привабливість науково-технічної розробки.

Ключові слова: розпізнавання, породи котів, ImageNet, згорткова нейронна мережа.

ABSTRACT

Kuzyk Y. O. Information technology for recognition of cat breeds based on a convolutional neural network. Master's thesis in the specialty 122 - «Computer sciences», educational program – «Artificial intelligence systems». Vinnytsia: VNTU, 2023. 109 p.

In Ukrainian language. Bibliogr .: 22 titles; fig . 25; table 7.

In this master's thesis, the choice of the SqueezeNet v.1.1 convolutional neural network for recognition of cat breeds is substantiated, which accepts an input image with a size of 224x224x3 pixels and can recognize 35 cat breeds. The software implementation of the information technology for recognizing cat breeds was created in the Python programming language in the Raspberry PI software environment. The convolutional neural network was trained using the ImageNet image database. The training sample consisted of 2800 images (80 images for each of the 35 cat breeds). The test sample consisted of 700 images (20 images for each of 35 cat breeds). The developed program has a recognition accuracy of cat breeds of 94.3%, and the analog program Cat Scanner has an accuracy of 89.9% recognition of cat breeds.

The graphic part consists of 6 posters.

In the economic section, the expediency of conducting research work on this topic is proven. The payback period is 0.8 years, which indicates the commercial attractiveness of scientific and technical development.

Key words: recognition, cat breeds, ImageNet, convolutional neural network

ЗМІСТ

ВСТУП

Актуальність дослідження. На сучасному етапі спостерігається істотне збільшення потужності обчислювальних засобів та бурхливий розвиток інтелектуальних інформаційних технологій і систем штучного інтелекту (ШІ). Їх застосовують, зокрема, для розпізнавання образів різної природи на фото та відео. Розпізнавання образів – вельми актуальна задача у сучасному світі. У даній роботі ця задача розв’язується на прикладі розпізнавання порід котів.

Це нетривіальна задача, тому для неї найкраще підходять засоби штучного інтелекту. Одним з напрямків штучного інтелекту, що активно розвиваються, є штучні нейронні мережі, які працюють за принципами біологічних нейронних мереж і являють собою систему взаємодіючих між собою штучних нейронів. Серед основних галузей застосування штучних нейронних мереж можна виділити: класифікацію зображень, обробку звукової та текстової інформації, прийняття рішень, прогнозування, аналіз даних, оптимізація.

Значна частина останніх досліджень присвячена глибоким згортковим нейронним мережам (ЗНМ) і зосереджена на підвищенні точності комп’ютерного зору. Існує багато архітектур ЗНМ, які досягають достатнього рівня точності для поставленої задачі.

Враховуючи ці переваги, було зосереджено роботу безпосередньо на проблемі визначення архітектури ЗНМ з меншою кількістю параметрів, але еквівалентною точністю в порівнянні з відомими моделями. В процесі роботи було використано таку архітектуру, як SqueezeNet.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних

інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета і завдання досліджень. Метою магістерської кваліфікаційної роботи є підвищення достовірності розпізнавання порід котів за рахунок використання згорткової нейронної мережі.

Для досягнення мети розробки необхідно виконати такі задачі:

- провести аналіз предметної області розпізнавання порід котів;
- розглянути існуючі методи розпізнавання порід котів та обрати й обґрунтувати вибір методу, який задовольняє мету даної магістерської кваліфікаційної роботи;
- розробити структуру згорткової нейронної мережі;
- розробити структуру процесів обробки інформації при розпізнаванні порід котів та алгоритм роботи програмного засобу;
- виконати програмну реалізацію запропонованої інформаційної технології;
- провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об’єкт дослідження – процес комп’ютеризованого розпізнавання порід котів за допомогою згорткової нейронної мережі.

Предмет дослідження – інформаційна технологія та програмні засоби розпізнавання порід котів за допомогою згорткової нейронної мережі та достовірність їх роботи.

Методи дослідження. У роботі використані наступні методи наукових досліджень: системного аналізу, теорії нейронних мереж, методи математичної статистики для розробки процесу розв'язання задачі розпізнавання порід котів на основі згорткової нейронної мережі та обрахунків результатів експериментів із програмним засобом.

Наукова новизна одержаних результатів.

Набула подальшого розвитку інформаційна технологія розпізнавання порід котів, яка відрізняється використанням згорткових нейронних мереж,

що дозволило підвищити достовірність програмних засобів розпізнавання порід котів.

Практичне значення одержаних результатів полягає в тому, що на основі проведених досліджень розроблено програмне забезпечення розпізнавання порід котів.

Запропонована інформаційна технологія сприяє підвищенню достовірності програмних засобів розпізнавання порід котів, зокрема:

- розроблено алгоритм роботи програмного забезпечення розпізнавання порід котів;
- розроблено програмні засоби для розпізнавання порід котів.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю використання математичного апарату методів дослідження, експериментальними дослідженнями тестування програмної реалізації інформаційної технології розпізнавання порід котів. Адекватність розроблених математичних моделей підтверджується результатами експериментальних досліджень.

Особистий внесок здобувача. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У працях, написаних у співавторстві, здобувачу належать: аналіз процесу розпізнавання порід котів на основі згорткової нейронної мережі та методів підвищення достовірності програмних засобів розпізнавання порід котів [1].

Апробація результатів роботи. Результати роботи були апробовані на Всеукраїнській науково-практичній Інтернет-конференції «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ (МН-2024)» (м. Вінниця, Україна, 2023-2024) [1].

Публікації. За результатами досліджень опубліковано одні тези доповіді на науково-технічній конференції [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗПІЗНАВАННЯ ОБРАЗІВ

1.1 Постановка задачі

Штучні нейронні мережі [2] (artificial neural networks, ANN) – системи з`єднаних елементарних процесорів (штучних нейронів), прототипом яких вважаються біологічні нейромережі, які є частиною мозку тварин. Такі нейромережі «навчаються» виконувати завдання, користуючись прикладами, а не програмуються на конкретні задачі. Вони автоматично визначають образи за характеристиками, які у такі мережі надходять.

Для штучного інтелекту [3] на теперішній час навіть задача розпізнавання тварини на зображенні – важкий процес, який потребує зусиль зі сторони розробників (і навіть коли цей процес працює, то говорять не про гарантований результат, а радше про деякий процент або коректних відповідей, або таких, що мають певний сенс).

Враховуючи складність такої проблеми, на даний момент основні дослідження зосереджені на розв’язанні конкретних практичних завдань, як от розпізнавання зображень, що і розглядається у даній роботі.

Галузь машинного навчання [4] тісно зв’язане із завданням класифікації. Фактично, машинне навчання є частиною довільного класифікатора. Наприклад, машинне навчання із учителем – це процес тренування функції з метою примусити її видавати бажані результати, основуючись на прикладах коректних пар «вхідні параметри – вихідний сигнал», що попередньо вивчені.

Функція системи створюється за допомогою помічених навчальних даних, які складено із набору типових об’єктів розпізнавання з наперед відомим результатом. Кожен тренувальний зразок є парою, що містить вхідний об’єкт (зазвичай вектор параметрів) та бажану відповідь. Алгоритм контрольованого машинного навчання оглядає отримані через «вчителя»

дані та видає функцію, яку можна використати для аналізу нових невідомих прикладів.

Таким чином, якісно навчений алгоритм зможе правильно визначати клас об’єктів (назву класу), які ще жодного разу не надходили на обробку. Висувається вимога, щоб результат тренування узагальнював отриману під час навчання інформацію та міг реагувати на неочікувані ситуації таким чином, щоб це мало сенс з огляду людини.

Практична задача – розробка методу розпізнавання породи котячих за їх зображенням. Вхідними даними для інформаційної технології будуть файли зображень у форматі JPEG, на яких зображено кота у повний зріст без сторонніх об’єктів, і бажано з однорідним фоном. Вихідними даними для інформаційної технології є назва породи та ймовірність того, що такий висновок є істиною. Таким чином, інформаційна технологія для розпізнавання порід котів отримує на вході зображення, і визначає породу, до якої відноситься кіт з певною імовірністю.

1.2 Огляд відомих методів класифікації зображень

Задача розпізнавання зображень є окремим випадком задачі класифікації об’єктів. При розв’язанні класичної задачі класифікації зображень часто застосовують математичні методи, які ґрунтуються на логічних міркуваннях і принципах математичної статистики. На противагу цьому підходу, існують методи класифікації зображень на основі машинного навчання і штучних нейромереж, сформовані не на суто логічних та формалізованих підходах до класифікації, які демонструють не гірший, а іноді і значно кращий результат.

Існує багато методів для розпізнавання образів[5]. Головна відмінність між ними полягає у наборі ознак, що розпізнаються. Прийнято поділ на чотири загальних групи методів розпізнавання образів [6]:

- методи співставлення шаблонів,

- статистичні методи,

- структурні методи,

- нейромережеві методи.

При співставленні шаблонів визначається ступінь подібності між двома векторами (групами пікселів, значеннями ознак, вигинів, фігур і т.п.) у просторі ознак.

Статистичні методи грунтуються на статистичних функціях прийняття рішень і наборах оптимальних критеріїв, які визначають ймовірність належності зображення, що спостерігається, до відповідного класу. Такі популярні підходи розпізнавання образів належать до цієї групи:

Метод k-найближчих сусідів (k-NN).

Метод k-найближчих сусідів (англ. k-nearest neighbor) — це (непараметричний) метод керованого навчання, вперше розроблений (Евеліном Фіксом) та (Джозефом Ходжесом) у 1951 році. Метод використовується і для класифікації, і для регресії. В обох випадках вхідні дані формуються з k найближчих тренувальних прикладів з набору даних. Результат обробки залежатиме від того, для чого саме застосовується метод k-NN: для регресії чи для класифікації:

Метод k-NN — це вид класифікації, у якому функція апроксимується лише локально, а всі обрахунки відкладаються до оцінювання функції. Так як даний алгоритм використовує функцію відстані для класифікації, то у тому випадку, коли ознаки являють собою різні фізичні величини або мають занадто різні масштаби, то нормалізація тренувальних даних може суттєво збільшити їх точність.

Як для регресії, так і для класифікації, корисним буде призначення вагових коефіцієнтів внескам сусідів так, щоб внесок у середнє значення у найближчих сусідів був більшим, ніж у віддалених. Наприклад, суть загальної схеми зважування полягає в наданні кожному сусіду ваги 1/d, де d — відстань до сусіда.

Сусіди обираються із множини об'єктів, для котрих відомо клас (у випадку класифікації k-NN) або відомо значення властивості об'єкта (у випадку регресії k-NN). Це розглядається як навчальний набір даних для алгоритму, хоч ніякого реального кроку навчання виконувати не потрібно.

До особливостей алгоритму k-NN відноситься те, що він є чутливим до локальної структури даних. Це популярний непараметричний метод розпізнавання, де апостеріорна ймовірність приблизно визначається через частоту найближчих сусідів невідомого шаблону. Такий підхід дає достатньо хороші результати, хоч треба зазначити, що він вимагає значних обчислювальних витрат в процесі розпізнавання.

Приховані моделі Маркова (ПММ).

ПММ є ще одним популярним методом вирішення задачі. Прихована марковська модель, ПММ (англ. hidden Markov model) — статистична марковська модель, у якій система, яка моделюється, буде розглядатися як марковський процес із прихованими (неспостережуваними) станами. ПММ можна представити як найпростішу баєсову динамічну мережу. Для ПММ математичний апарат розробив Леонард Баум та співробітники. Він був тісно пов'язаний із попередньою працею про нелінійну оптимальну проблему фільтрування Стратоновича Руслана, який був першим, хто описав послідовно-зворотній алгоритм.

У більш простих марковських моделях (ланцюгах Маркова) стан є видимим безпосередньо спостерігачу, а тому ймовірності переходів станів є єдиними параметрами. У прихованій моделі Маркова стан є невидимим безпосередньо, але вихід, який залежить від стану, є видимим. Кожен стан має імовірнісний розподіл усіх вихідних значень. Таким чином, згенерована ПММ послідовність символів дає деяку інформацію про послідовність станів. Ознака «прихований» відноситься до послідовності станів, якими модель проходить, а не параметрів моделі. Модель все-одно називають «прихованою» марковською моделлю навіть тоді, коли ці параметри точно відомі.

Приховані марковські моделі в першу чергу відомі, завдячуючи їхньому використанню у розпізнаванні часових шаблонів, до якких відносять розпізнавання рукописного введення, мовлення, жестів, мелодій для акомпонуваня, морфологічної розмітки, часткових розрядів та у біоінформатиці.

Приховані марковські моделі можна розглядати як узагальнення сумішевої моделі, де латентні (або приховані) змінні, які контролюють, яка саме складова суміші буде обиратися для кожного спостереження, зв'язані марковським процесом, а не є незалежними одна від іншої. Не так давно приховані марковські моделі було зведено до подвійних марковських моделей (pairwise Markov models) та до триплетних марковських моделей (triplet Markov models), що дозволило моделювати нестаціонарні дані та розглядати більш складні структури даних.

Модель є двічі стохастичним процесом: вона містить неспостережуваний процес (а тому – «прихована»), що, також, може спостерігатись і через інший випадковий процес, що видає послідовність спостережень. Найважливішою властивостю є існування ефективних алгоритмів автоматизованого тренування моделі без необхідності маркувати попередньо сегментовані дані.

Метод опорних векторів (SVM – Support Vector Machine).

Метод опорних векторів у машинному навчанні — це метод обробки даних для регресійного аналізу та класифікації за допомогою алгоритмів керованого навчанням, які називають машинами опорних векторів (МОВ, англ. support vector machines, SVM, також опорно-векторними мережами, англ. support vector networks). Для заданого набору навчальних прикладів, кожен із яких помічений як той, що належить до однієї із двох категорій, алгоритм навчання МОВ будує модель, що буде відносити нові приклади до першої чи другої категорії, і робити це неімовірнісним лінійним бінарним класифікатором. Модель МОВ є представленням прикладів як точок у багатовимірному просторі, які відображені так, що приклади із окремих категорій розділені проміжком, який є найширшим. Нові приклади тоді

відносяться до цього ж простору, і робиться пророкування про їх приналежність до певної категорії на основі того, на який бік проміжку вони потрапляють.

МОВ можуть ефективно виконувати не тільки лінійну класифікацію, але і нелінійну при застосуванні так званого ядрового підходу, відображуючи неявно свої входи на простори ознак великої вимірності.

Коли дані є нерозміченими, то кероване навчання буде неможливим, і виникає потреба у некерованому навчанні, що намагається виконати природне кластеризування даних на групи, а потім відображати нові дані на ці сформовані кластери (групи). Алгоритм кластеризації, який забезпечує вдосконалення машинам опорних векторів, називається опорно-векторною кластеризацією (англ. support vector clustering), і часто використовується в практичних застосуваннях, коли дані або є нерозміченими, або лише деякі дані є розміченими як попередня обробка перед класифікацією.

Метод опорних векторів засновано на квадратичній оптимізації та статистичній теорії навчання. SVM – це, фактично, бінарний класифікатор, і декілька SVM можна об'єднати, щоб отримати систему для багатокласової класифікації. Має відмінну продуктивність щодо узагальнення.

Відомо також кілька інших статистичних підходів: поліноміальний дискримінантний або байєсівский класифікатори, але вони менш популярні для задачі, яка розглядається в роботі, і часто видають гірші результати.

У структурних методах образи представлено у виді об’єднань структурних примітивів. Вони підлягають кількісній оцінці, і може бути знайдений взаємозв'язок між ними. Структурні методи розділяють на два класи: графічні та граматичні методи [7].

Найбільш універсальний підхід до поставленої задачі – нейромережевий. Штучна нейромережа – це обчислювальне середовище, яке складене зі штучних нейронів, які є моделями нервових клітин людини [8]. Ці структури створено при спробах імітації людського мозку та широко застосовуються в обробці даних, розпізнаванні образів та задачах

апроксимації функцій. Основні переваги нейронних мереж: здатність автоматично навчатися на прикладах, є ефективними на зашумлених даних, можлива паралельна реалізація, а також вони є ефективними інструментами для обробки великих наборів даних. Нейромережі часто використовуються в галузі, яка розглядається, і показують гарні результати. Є багато різних видів нейромереж. Найбільш вдалими є мережа Хеммінга, мережа Хопфілда, нечіткі нейронні мережі, самоорганізувальні карти Кохонена та ін [2].

Через все вищезазначене оберемо для розв'язання поставленої задачі розпізнавання порід котів саме метод на основі штучних нейронних мереж.

1.3 Обгрунтування вибору аналогу до програми розпізнавання порід котів

На сьогоднішній день існує декілька відомих програмних додатків, що здатні здійснювати розпізнавання порід котів.

Одним з найбільш популярних додатків є «Cat Scanner: Breed Recognition» [9]. Основні його характеристики: точність розпізнавання 90%, доступність використання на різних платформах. Також є можливість використання API даної програми. Додаток Cat Scanner визначає породу кота всього за кілька секунд. Окрім фотозйомки, ви також можете записати відео або завантажити зображення зі своєї галереї. Додаток Cat Scanner також розпізнає змішані породи. Надає докладні дані та цікаві факти про різні породи вашої змішаної породи котів. Вид вікна цієї програми представлено на рис. 1.1.

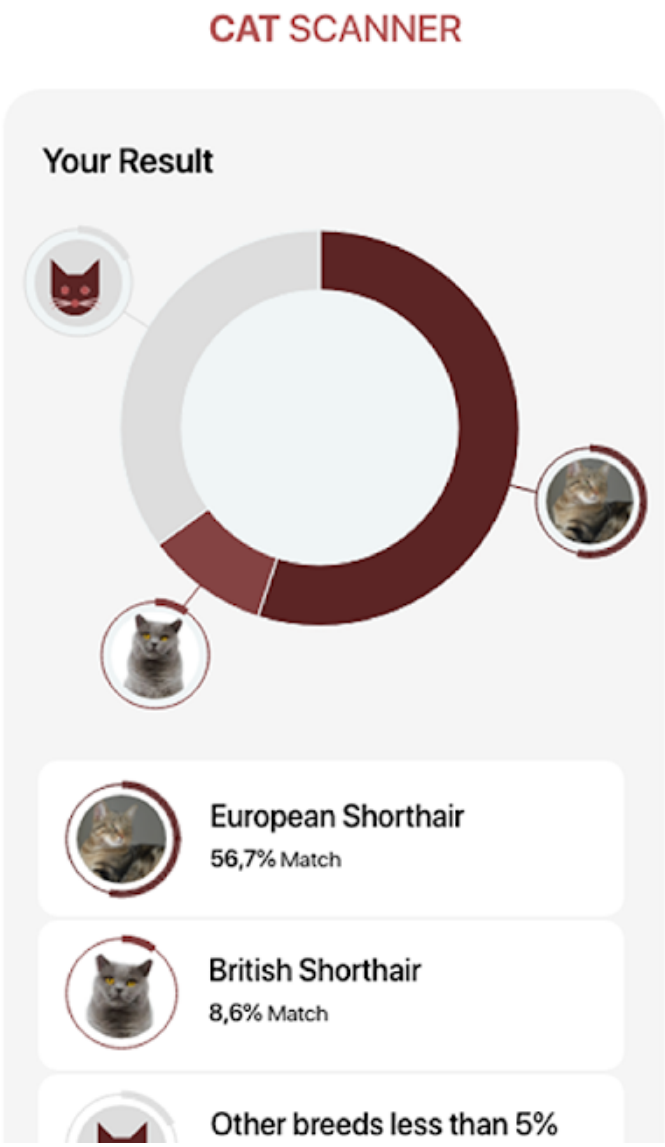

Рисунок 1.1 – Результат роботи додатку Cat Scanner

До недоліків цієї програми можна віднести невисоку достовірність розпізнавання порід котів.

Інші програмні реалізації розпізнавання порід котів також мають такий недолік як невисока достовірність. З огляду на це, постає завдання розробки нового програмного засобу для розпізнавання порід котів з підвищеною достовірністю роботи.

1.4 Висновок до розділу 1

У розділі було розглянуто постановку задачі розпізнавання порід котів, проведено огляд відомих методів класифікації зображень, які можна використовувати для поставленої задачі. Обґрунтовано доцільність вибору методу розпізнавання зображень на основі нейронних мереж, зокрема згорткових нейронних мереж для розпізнавання порід котів. Було проаналізовано різні програмні додатки для розпізнавання порід котів та обрано аналог, головним недоліком якого є невисока достовірність розпізнавання порід котів, що ставить мету дослідження – підвищення достовірності розпізнавання порід котів.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ПОРІД КОТІВ ЗА ДОПОМОГОЮ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

2.1 Обґрунтування вибору згорткової нейронної мережі для розпізнавання порід котів

При роботі з вхідними даними високої розмірності, такими як зображення, непрактично з'єднувати нейрони з усіма нейронами в попередньому об'ємі (як є у традиційних багатошарових персептронах). Замість цього варто з'єднувати кожен нейрон лише з локальною областю вхідного об'єму. Просторова протяжність цієї зв'язності є гіперпараметром, який називається рецептивним полем нейрона (еквівалентно розміру фільтра). Протяжність зв'язку вздовж осі завжди є рівною глибині вхідного обсягу. Важливо підкреслити ще раз цю асиметрію тому, що ми розглядаємо просторові розміри (висоту і ширину) та вимір глибини: З'єднання є локальними у двовимірному просторі (по ширині та висоті), але завжди повними по всій глибині вхідного зображення.

Згорткові нейронні мережі [10] є схожими на традиційні нейромережі: вони містять нейрони, що мають вагу та зміщення та можуть навчатися. Кожен нейрон отримує певні вхідні дані, виконує скалярний добуток і, за потреби, обробляє його за нелінійністю. Вся мережа все ще виражає єдину диференційовану функцію оцінки: від пікселів сирого зображення на одному кінці до оцінок класів на іншому. І вони мають ще функцію втрат (напр., SVM/Softmax) на останньому повнозв'язному шарі, і всі поради/підказки, які розроблено для навчання звичайних нейронних мереж, застосовуються і для навчання згорткових нейронних мереж.

Згорткова нейромережа (ConvNet/CNN) — це модель глибокого навчання, яка приймає вхідне зображення, привласнює важливість (вивчені ваги та зміщення) аспектам або об'єктам зображення і відрізняти одне від

одного. За цим, зображення вимагають набагато менше попередньої обробки порівняно з іншими об`єктами. У примітивних мережах фільтри розробляють вручну, але навчені CNN мережі тренуються використовувати ці фільтри/характеристики.

Архітектура CNN будується за аналогією зі структурою зв'язків нейронів у людському мозку, а також з організацією зорової кори головного мозку. Деякі нейрони реагують на стимули лише у деякій ділянці поля зору, також відомого як рецептивне поле. Безліч рецептивних полів перекривається повністю покриваючи поле зору CNN. Приклад структури згорткової нейронної мережі наведено на рис. 2.1.

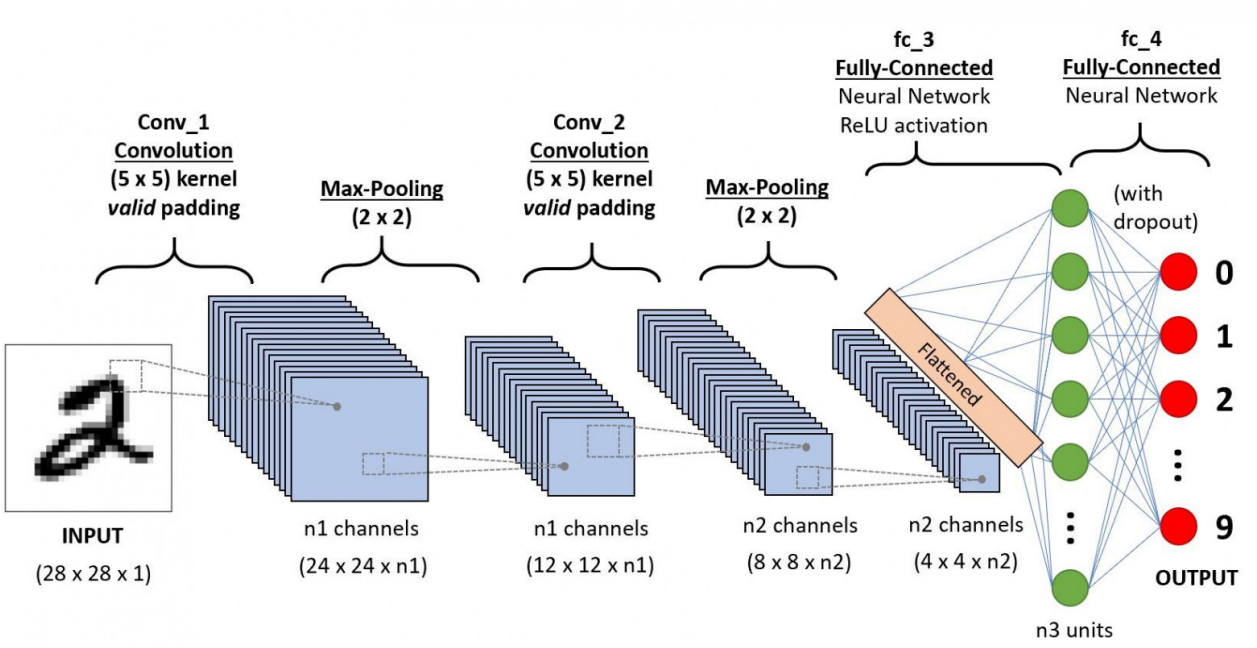

Рисунок 2.1 – Приклад структури згорткової нейронної мережі

Зображення - це матриця значень пікселів. Але можна зробити його плоским (наприклад, матрицю 3×3 зробити вектором 9×1) та згодувати цей вектор багатошаровому перцептрону, щоб він виконав класифікацію? Виявляється, все не так просто.

У випадках найпростіших двійкових зображень при виконанні прогнозування класів метод може показати середню точність, але на практиці,

коли мова піде про складні зображення, в яких піксельні залежності, він виявиться неточним.

Згорткова нейромережа здатна успішно виявляати часові та просторові залежності у зображенні завдяки застосуванню відповідних фільтрів. Така архітектура дає кращу відповідність набору зображень завдяки скороченню кількості задіяних параметрів та здатності повторного використання ваг. Тобто нейромережу можна натренувати розуміти краще складність зображення.

На рис. 2.2 подано RGB-зображення, розділене на три колірні площини (синю, зелену та червону), яке може бути описане у різних колірних просторах - у Grayscale (відтінках сірого), HSV, RGB, CMYK і т.п.

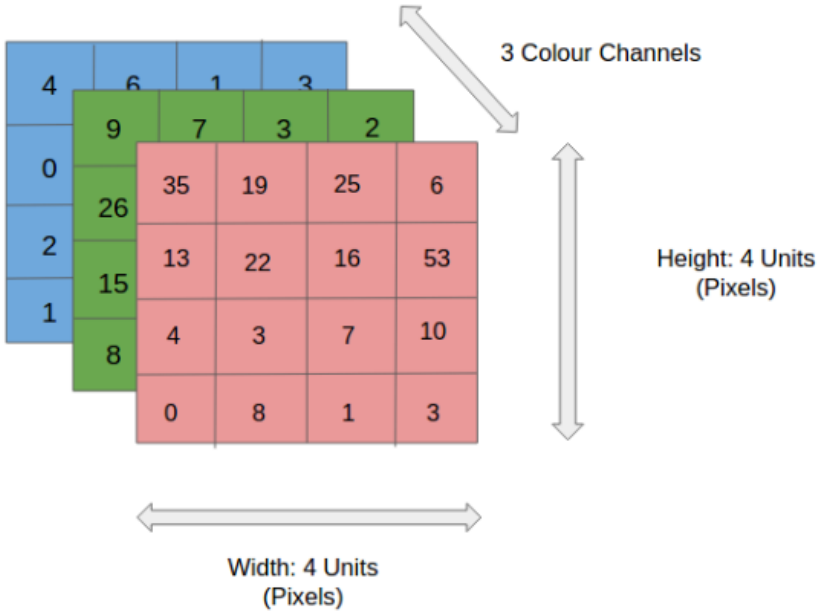

Рисунок 2.2 – RGB-зображення, розділене на три колірні площини (червону, зелену та синю)

Коли зображення досягнуть розмірів, наприклад, 8 K (76804320), то обчислення будуть набагато інтенсивнішими, Призначення згорткової нейромережі полягає в тому, щоб перетворити зображення у форму, яку легко обробляти, не втрачаючи при цьому ознаки, які мають вирішальне значення в

одержанні гарного прогнозу. Це важливо при синтезі архітектур, які не тільки добре відтворюють функції, а й масштабуються для великих наборів даних.

Шар згортки – ядро.

В основі згорткових нейромереж лежить операція згортки (рис. 2.3). Розміри зображення:

5 - висота;

5 - ширина;

1 — кількість каналів, наприклад RGB.

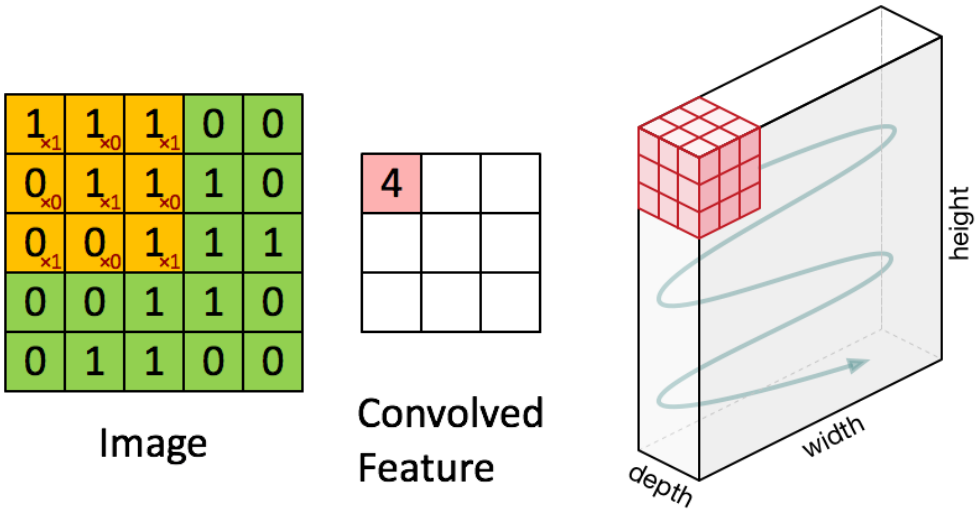

Рисунок 2.3 – Операція згортки зображення 5x5x1 з ядром згортки 3x3x1

На рис. 2.3 вхідне зображення 5×5×1. Елемент, який бере участь у здійсненні операції згортки у першій частині згорткового шару, має назву ядра (фільтру) K. Нехай K є матрицею 3×3×1:

Kernel/Filter, K =

1 0 1

0 1 0

1 0 1

Ядро зміщується 9 разів із довжину кроку одиниця, і щоразу виконує операцію множення матриць K та P, над якою знаходиться ядро.

Фільтр переміщується з певним значенням кроку праворуч доти, доки не проаналізує всю ширину зображення. Далі рухаючись, він переходить ліворуч до початку зображення і з тим самим значенням кроку повторює процес, доки не пройде все зображення.

У випадку багатоканальних зображень (напр., RGB) ядро і вхідне зображення мають однакову глибину (рис. 2.4). Матричне множення виконується між стеками Kn та In ([K1, I1]; [K2, I2]; [K3, I3]), всі результати підсумовуються зі зміщенням, щоб отримати сплощений канал виведення згорнутих ознак з глибиною 1.

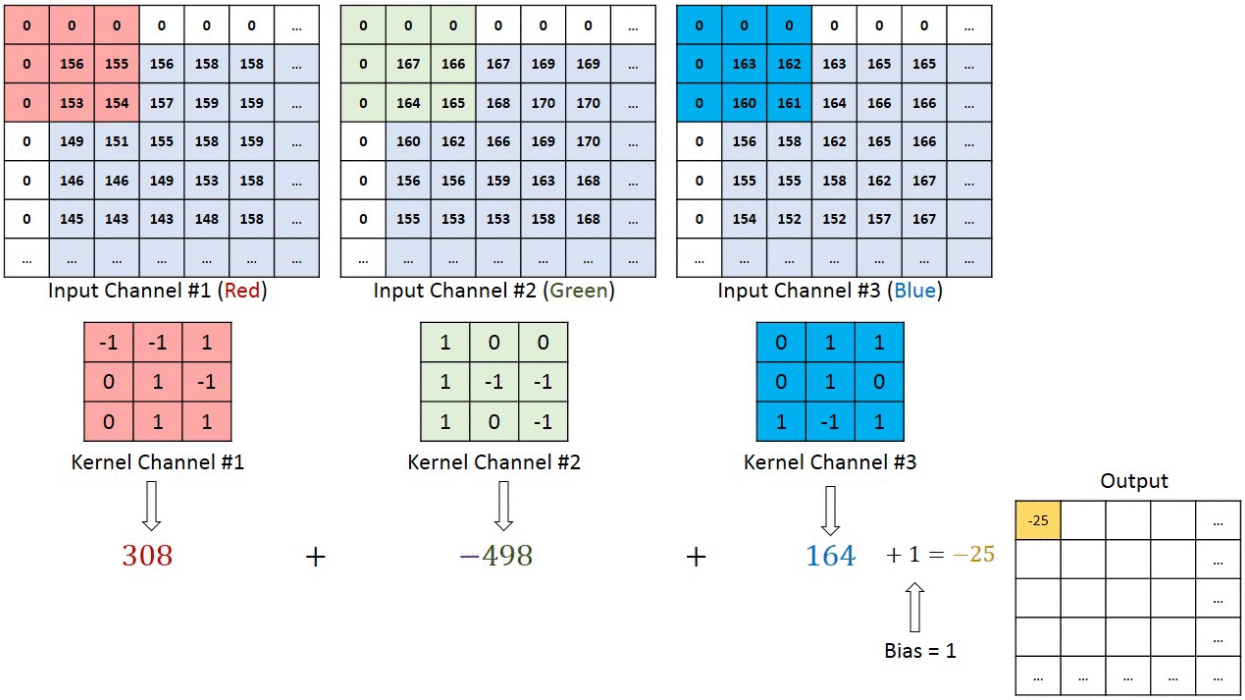

Рисунок 2.4 – Рух ядра згортки 3x3x3 по зображенню 7x7x3

Згортка робиться з метою витягти високорівневі ознаки, наприклад, контури вхідного зображення. Мережу не потрібно обмежувати єдиним шаром. Перший шар умовно несе відповідальність за схоплення ознак низького рівня, таких як межі, орієнтація градієнта, колір і т. п. Через додаткові шари ця архітектура адаптується до високорівневих ознак, і ми

отримуємо нейромережу зі здоровим розумінням зображень у наборі даних, схожих на наше.

У результатів згортки є два типи ознак: 1-ий - згорнута ознака зменшується у розмірі порівняно з вхідним розміром, 2-ий тип стосується розмірності - вона залишається якою була, або збільшується. Робиться це шляхом застосування допустимого заповнення у 1-му випадку або нульового заповнення – у 2-му.

Збільшуючи зображення 5×5×1 до 6×6×1, а потім проходячи над ним ядром 3×3×1, ми виявимо, що згорнута матриця матиме роздільну здатність 5×5×1. Звідси й назва – нульове наповнення. З іншого боку, зробивши те саме без заповнення, ми виявимо матрицю з розмірами самого ядра (3×3×1); ця операція називається допустимим наповненням.

Шар об`єднання.

Подібно згортковому шару, шар об'єднання відповідає за зменшення в просторі розміру згорнутого об'єкта. Це робиться для зменшення обчислювальної потужності, необхідної під час обробки даних, за рахунок скорочення розмірності. На додачу, це є корисним для отримання домінуючих ознак, що є обертальними та позиційними інваріантами. Цим дозволяється підтримувати процес ефективного тренування моделі (рис. 2.5).

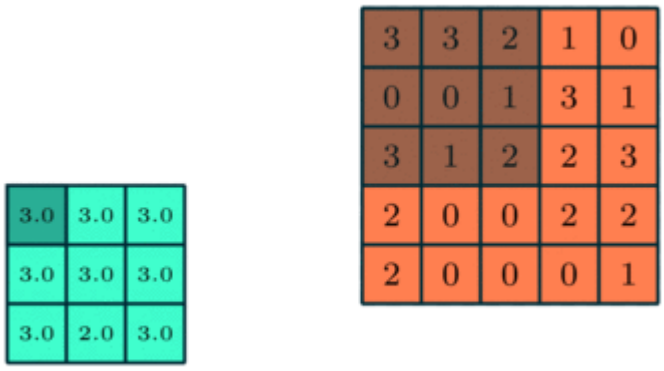

Рисунок 2.5 – Об'єднання 3×3 над згорнутою ознакою 5×5

Є два типи об'єднання: максимальне та середнє (рис. 2.6). Перший тип повертає максимальне значення із частини зображення, покритої ядром. А другий тип об'єднання повертає середнє значення з усіх значень частини, покритої ядром.

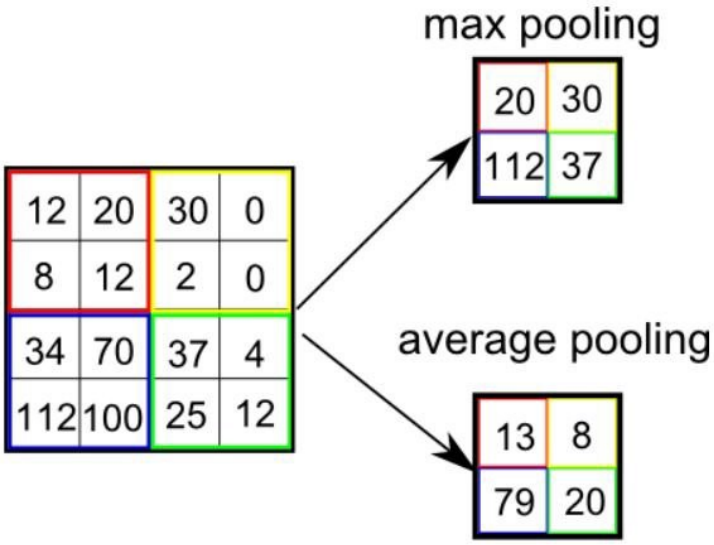

Рисунок 2.6 – Типи об'єднання

Максимальне об'єднання, до того ж, виконує функцію шумоподавлення. Воно відкидає зашумлені активації повністю, а також усуває шум на додачу до зменшення розмірності. З іншого боку, для придушення шуму середнє об'єднання просто знижує розмірність. Таким чином, можна казати, що максимальне об'єднання набагато краще працює за середнє об'єднання.

Шари об'єднання та згортки утворюють разом і-тий шар згорткової нейромережі. Число таких шарів можна збільшувати залежно від складності зображень для того, щоб краще виокремлювати деталі, але це досягається за рахунок збільшення обчислювальних витрат.

Виконання описаного вище процесу дозволяє моделі розуміти особливості зображення. Потім результат перетворюється на стовпцевий вектор і подається на вхід звичайної нейронної мережі, що класифікує.

Класифікація - повнозв'язковий шар.

Додавання повнозв'язкового шару (рис. 2.7) - це (зазвичай) обчислювально невитратний спосіб навчання на нелінійні комбінації

високорівневих ознак, що представлені на виході згорткового шару. Повнозв'язний шар має нелінійну функцію активації у цьому просторі.

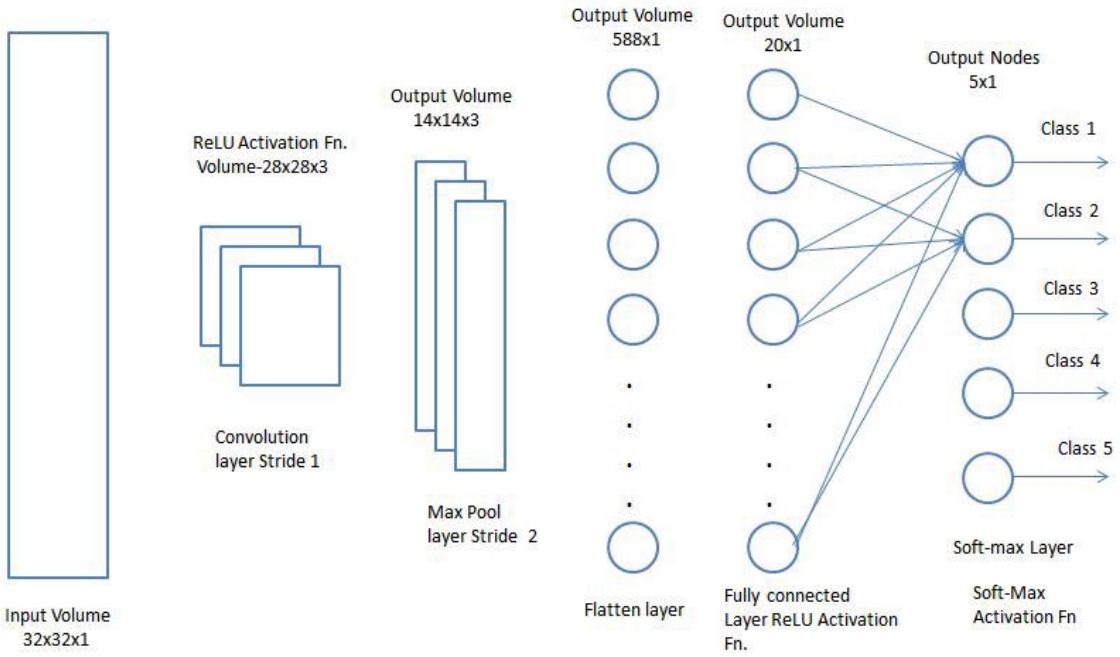

Рисунок 2.7 – Додавання багатошарового персептрону на виході згорткової нейронної мережі

Після перетворення вхідного зображення у придатну для багатошарового персептрону форму ми повинні згладити зображення у стовпець вектор. Згладжений вихідний сигнал подається на прямозв'язану нейромережу, для якої застосовується зворотне поширення на кожній ітерації навчання. За кілька епох мережа набуває здатність розрізняти домінуючі та деякі низькорівневі ознаки на зображеннях та класифікувати їх методом Softmax.

2.2 Обґрунтування вибору типу архітектури згорткової нейронної мережі

Згорткові нейронні мережі (ЗНМ) [10,11] використовують різновид багатошарових перцептронів, розроблений так, щоби вимагати використання

мінімального обсягу попередньої обробки. Згорткові нейромережі використовують мало попередньої обробки відносно інших алгоритмів класифікації зображень. Це означає, що нейромережа навчає фільтри, які в звичайних алгоритмах створювали вручну. Ця незалежність у формуванні ознак від апріорних знань та зусиль розробників є великою перевагою. Тому з розвитком технологій процес розробки нейромереж стає більш поширеним та багато різних компаній створюють свої програмні застосунки по розпізнаванню різноманітних об`єктів, не виключенням є і розпізнавання породи котів.

Було вирішено реалізувати дану інформаційну технологію розпізнавання порід котів за допомогою згорткової нейронної мережі [10] у програмному середовищі Raspberry Pi [12]. Тому вибір реалізації нейромережі достатньо не великий, так як Raspberry містить не достатньо високі характеристики. Відповідно до цього було виділено 7 мереж для реалізації інформаційної технології розпізнавання порід котів, а саме:

- MobileNet;
- MobileNet_depthwise;
- Res50;
- GoogleNet;
- SqueezeNet v.1.0;
- SqueezeNet v.1.1;
- ShuffleNet.

Проблема реалізації нейронної мережі у вбудованій системі полягає в обмеженнях на пам’ять та обчислювальні ресурси. Тобто нейромережа повинна мати невеликі обчислювальні потреби без втрати точності. Для цього більш детально було розглянуто три досить нові архітектури: SqueezeNet, MobileNet і ShuffleNet.

Mobilenet є реалізацією MobileNet від Google. Mobilenet має точність Top-1 70,81% і точність Top-5 89,5% порівняно з провідною моделлю за точністю, Densenet201, з 77,31% для Top-1 і 93,64% для Top-5. Архітектура

MobileNet продемонструвала досить мінімальні втрати точності при зменшенні займаного обсягу з 4,7 Гфлопс до 0,56 Гфлопс.

Але результат виявився досить невтішним. Незважаючи на те, що мобільна мережа швидша за Densenet201, за швидкістю вона далеко не серед провідних моделей. Причина полягає у Vanilla реалізації згрупованих згорток у Caffe. Спеціальний перепис глибинних згорток (змінений з BVLC/caffe#5665) дав прискорення на порядок, зробивши MobileNet знову придатною для використання.

ShuffleNet є більш ефективною нейронною мережею завдяки поглибленню згортки та спеціальному перетасовуванню каналів.

Відомим є факт, що більша частина глибокого навчання пов’язана з налаштуванням параметрів. Потрібно посиленно вивчити область проекціювання згорткової нейронної мережі, щоб різко зменшити кількість параметрів, з якими доведеться мати справу. Пропонується зменшити мережу, починаючи з розумнішого дизайну замість використання розумної схеми стиснення. Існує 3 основні стратегії для зменшення кількості параметрів при максимальному підвищенні точності.

Стратегія 1. Зробити мережу меншою, замінивши фільтри 3x3 на фільтри 1x1. Ця стратегія зменшує кількість параметрів у 9 разів, замінюючи групу фільтрів 3x3 на фільтри 1x1. Спочатку це здається дуже заплутаним. Переміщуючи фільтри 1x1 по зображенню, кожен фільтр має менше інформації для перегляду, і тому він працюватиме погано, однак це не так! Зазвичай більший згортковий фільтр 3x3 фіксує просторову інформацію пікселів, розташованих близько один до одного. З іншого боку, згорткові фільтри 1x1 зосереджуються на одному пікселі та фіксують зв’язки між його каналами на відміну від сусідніх пікселів.

Стратегія 2. Зменшити кількість входів для решти фільтрів 3x3. Ця стратегія зменшує кількість параметрів за рахунок простого використання меншої кількості фільтрів. Систематичний спосіб це робити - шляхом подачі

«стиснутих» шарів (“squeeze” layers) у те, що називається «розширеними» шарами, як показано на рис. 2.8:

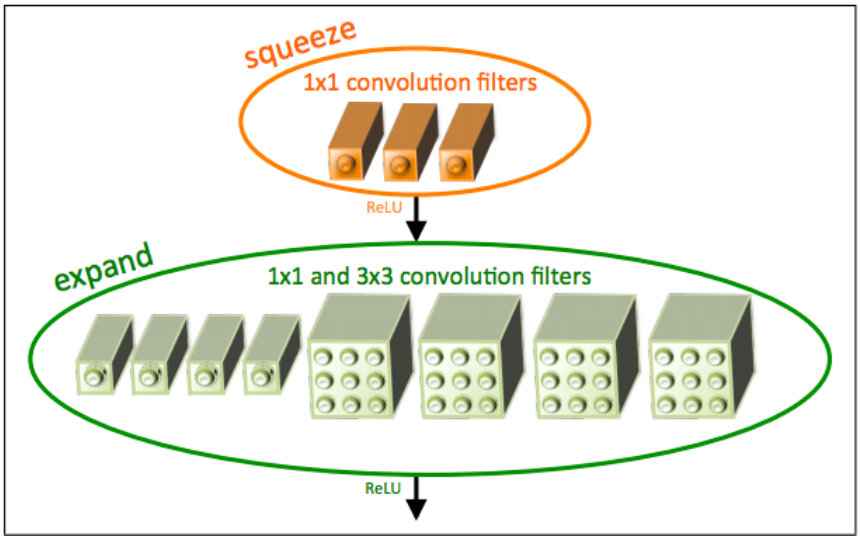

Рисунок 2.8 – Подача «стиснутих» шарів (“squeeze” layers) у «розширені» шари

Як видно з рис. 2.8, шари «стиснення» — це шари згортки, які складаються лише з фільтрів 1x1, а шари «розгортання» — це шари згортки з сумішшю фільтрів 1x1 і 3x3. Зменшуючи кількість фільтрів у шарі «стиснення», які надходять у шар «розгортання», вони зменшують кількість з’єднань, що входять до цих фільтрів 3x3, таким чином зменшуючи загальну кількість параметрів. Ця специфічна архітектура називається «пожежним модулем» (“fire module”), і вона служить базовим будівельним блоком для архітектури SqueezeNet.

Стратегія 3. Зменшення рівня дискретизації на крайніх шарах мережі, щоб згорткові шари отримали великі карти активації. Тепер, коли описано способи зменшення величезної кількості параметрів, з якими нейромережа працює, треба подумати як можна отримати максимальну віддачу від набору менших параметрів, що залишився? За рахунок зменшення кроку з пізнішими шарами згортки та створення більшої карти функцій активації пізніше в мережі точність класифікації фактично підвищується. Наявність більших карт

активації біля кінця мережі різко контрастує з такими мережами, як VGG, де карти активації стають меншими, коли ви наближаєтеся до кінця мережі. Цей різний підхід є дуже цікавим, подібним чином застосовано відстрочену вибірку, що призводить до вищої точності класифікації.

Щоб коректно обрати мережу для реалізації інформаційної технології розпізнавання порід котів потрібно перевірити ефективність кожної з них у даному програмному середовищі. В результаті було виведено наступний графік тестування ефективності нейронних мереж для реалізації інформаційної технології розпізнавання порід котів (див. рис. 2.9).

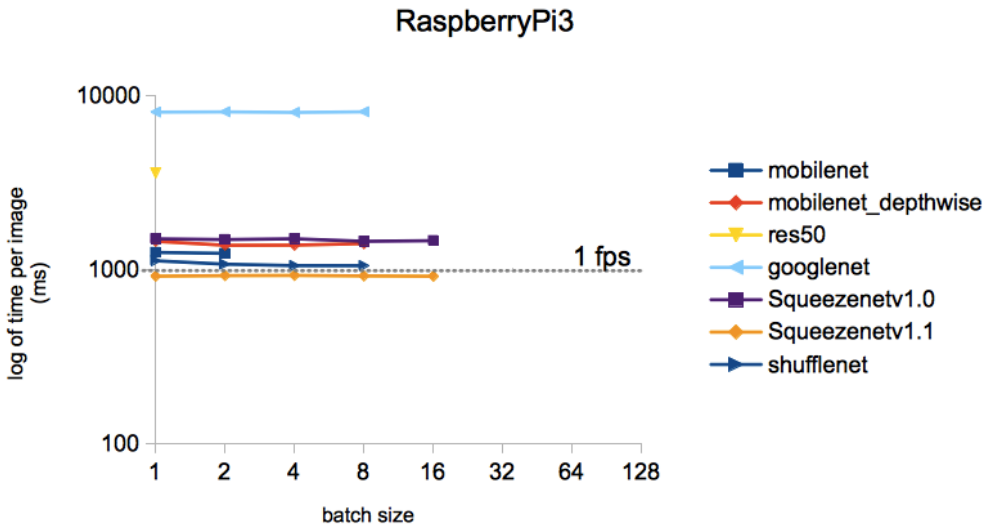

Рисунок 2.9 – Графік результатів тестування ефективності різних нейронних мереж

З даного графіку можна зробити висновок, що для обробки одного зображення для різних моделей, навчених по набору даних ImageNet [13], найшвидшим та ефективнішим буде SqueezeNet v.1.1 [14].

2.3 Архітектура згорткової нейронної мережі SqueezeNet

Штучна нейромережа – матмодель, принцип роботи якої імітує роботу мережі біологічних нейронів.

Кожен шар нейромережі представлений множиною вузлів – штучних нейронів, які видокремлюють ознаки все більшого рівня, доки крайній шар не скомбінує ці ознаки, щоб зробити висновок класифікації (див. рис. 2.10).

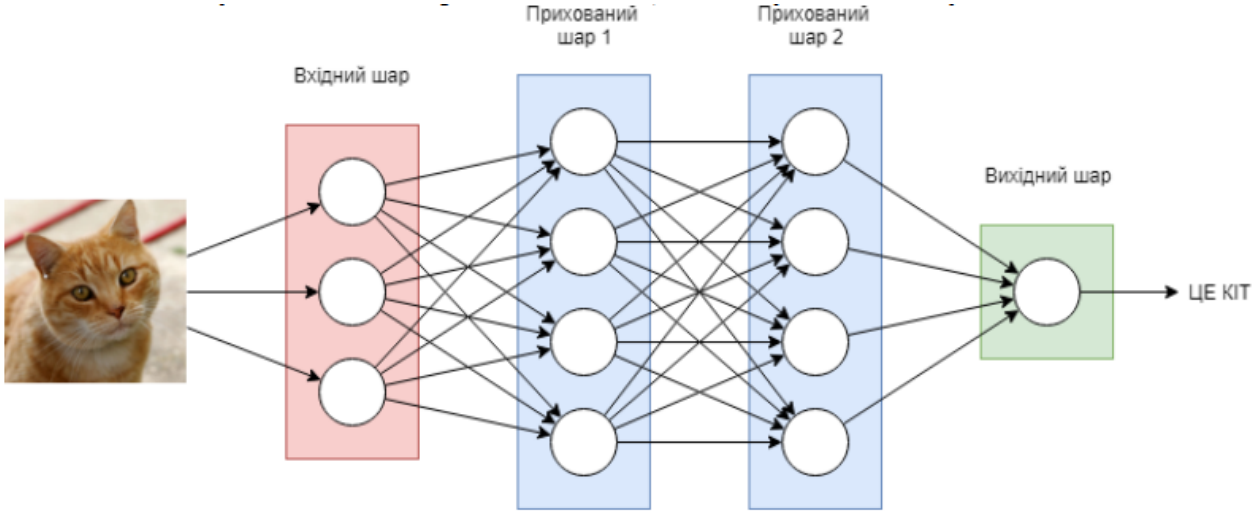

Рисунок 2.10 – Будова багатошарової нейронної мережі

Згорткові нейромережі – це клас глибоких штучних нейромереж прямого поширення, який вдало використовувався для аналізу зображень.

Згорткові нейромережі застосовують різновид багатошарових персептронів, який розроблено так, щоб потребувати використання мінімальної попередньої обробки. Ці мережі також є інваріантними відносно зсуву або просторово інваріантними штучними нейромережами, з огляду на їхню архітектуру спільних ваг та характеристик інваріантності відносно зсуву.

У згорткових нейронних мережах взято за основу біологічний прототип, а саме - схема з'єднань нейронів зорової кори тварин. Окремі нейроннв клітини зорової кори реагують на стимули від обмеженої області зорового поля, відомого як рецептивне поле. Рецептивні ділянки різних нейронів перекриваються частково так, що вони покривають повністю усе зорове поле.

Згорткові нейромережі потребують мало попередньої обробки в порівнянні з іншими методами класифікації зображень. Це означає, що нейромережа навчає фільтри, які в традиційних мережах обирали вручну. Ця незалежність у синтезі ознак від апріорних знань та зусиль розробників є значною перевагою.

ЗНМ мають широке застосування в розпізнаванні зображень та відео, у рекомендаційних системах та для обробки природної мови.

Згорткова нейромережа складається з вхідного та вихідного шарів, а також із кількох прихованих шарів. Приховані шари згорткової нейромережі звичайно складаються зі згорткових, агрегувальних, повнозв’язних шарів та шарів нормалізації.

Цей процес описується в нейромережах як згортка за домовленістю. З математичного погляду він є швидше взаємною кореляцією, ніж згорткою. Але це має значення лише для індексів у ваговій матриці, в також які ваги на якому індексі розташовуються.

Згорткові шари застосовують до вхідного зображення операцію згортки та передають результат до наступного шару. Згортка моделює реакцію окремої нейронної клітини на зоровий стимул.

Кожен нейрон згорткового шару обробляє пікселі тільки свого рецептивного поля. Хоча повнозв'єязні нейромережі прямого поширення можливо використовувати як для навчання ознак, так і для класифікації даних, застосування цієї традиційної архітектури до зображень є неефективним. Потребувалося би дуже велика кількість нейронів, навіть у поверхневій архітектурі, через занадто великі розміри вхідного шару нейронів, пов'язані із зображеннями, де кожен піксель відповідає нейрону. Наприклад, повнозв'язний шар для зображення розміром 100 × 100 мав би 10 000 нейронів. Операція згортки дає змогу вирішити цю проблему, так як вона зменшує кількість параметрів, дозволяючи нейромережі бути глибшою за умови меншої кількості параметрів. Наприклад, незалежно від розміру вхідного зображення, області фільтрів розміру 5 × 5, кожен з яких з одними й

тими ж спільними вагами, вимагають всього лиш 25 вільних параметрів. Тобто, це вирішує проблему зникнення або вибуху градієнтів у навчанні традиційних багатошарових нейромереж з кількома шарами за допомогою алгоритму зворотного поширення.

Агрегувальні шари. Згорткові нейромережі можуть містити шари локального або глобального агрегування, що об'єднують виходи кластерів нейронів одного шару до входів одного з нейронів наступного шару. Наприклад, агрегування по максимуму використовує максимальне значення із кожного з кластерів відповідних нейронів попереднього шару. З іншого боку, є агрегування за середнім значенням з кожного з кластерів нейронів попереднього шару.

Повнозв'язні шари по'єднують кожен нейрон одного шару з кожним нейроном наступного шару. Це, загалом, є таким же, як і у традиційній нейромережі багатошаровий персептрон (БШП).

Опишемо архітектуру SqueezeNet CNN [15,16]. На рис. 2.11 проілюстровано, що на початку SqueezeNet стоїть окремий шар згортки (conv1), за яким слідують 8 модулів Fire (fire2-9), завершуючи кінцевим згортковим шаром (conv10). Поступово збільшується кількість фільтрів на один модуль Fire від початку до кінця мережі. SqueezeNet виконує максимальний пул із кроком 2 після шарів conv_1, fire_4, fire_8 та conv_10. Повна архітектура SqueezeNet представлено в таблиці 2.1.

Деталі і варіанти дизайну SqueezeNet з таблиці 2.1 і рисунку 2.11:

• Щоб вихідні активації з фільтрів 1x1 і 3x3 мали однакову висоту і ширину, ми додаємо 1-піксельну межу нульового заповнення у вхідних даних до фільтрів 3x3 модулів розширення.

• Активаційна функція ReLU застосовується для активації шарів стиснення та розширення.

• Dropout із співвідношенням 50% застосовується після модуля fire9.

• У SqueezeNet відсутні повнозв'язні шари; цей вибір дизайну був натхненний архітектурою NiN [17].

• Під час навчання SqueezeNet ми починаємо з швидкості навчання 0,04 і лінійно зменшуємо швидкість навчання протягом тренування, як описано в [18].

• Структура Caffe не підтримує шар згортки, який містить кілька роздільних можливостей фільтрів (наприклад, 1x1 і 3x3). Щоб обійти це, ми реалізуємо шар розширення з двома окремими шарами згортки: шаром з фільтрами 1x1 і шаром з фільтрами 3x3. Потім ми об'єднуємо виходи цих шарів разом у розмірі каналу. Це чисельно еквівалентно реалізації одного шару, який містить фільтри 1x1 і 3x3.

Таблиця 2.1 - Архітектурні дані нейромережі SqueezeNet

layer name/type	output size	filter size / stride (if not a fire layer)	depth	s_{1x1} (#1x1 squeeze)	e_{1x1} (#1x1 expand)	e_{3x3} (#3x3 expand)	s_{1x1} sparsity	e_{1x1} sparsity	e_{3x3} sparsity	# bits	#parameter before pruning	#parameter after pruning
input image	224x224x3										-	-
conv1	111x111x96	7x7/2 (x96)	1				100% (7x7)			6bit	14,208	14,208
maxpool1	55x55x96	3x3/2	0									
fire2	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	11,920	5,746
fire3	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	12,432	6,258
fire4	55x55x256		2	32	128	128	100%	100%	**33%**	6bit	45,344	20,646
maxpool4	27x27x256	3x3/2	0									
fire5	27x27x256		2	32	128	128	100%	100%	**33%**	6bit	49,440	24,742
fire6	27x27x384		2	48	192	192	100%	**50%**	**33%**	6bit	104,880	44,700
fire7	27x27x384		2	48	192	192	**50%**	100%	**33%**	6bit	111,024	46,236
fire8	27x27x512		2	64	256	256	100%	**50%**	**33%**	6bit	188,992	77,581
maxpool8	13x12x512	3x3/2	0									
fire9	13x13x512		2	64	256	256	**50%**	100%	**30%**	6bit	197,184	77,581
conv10	13x13x1000	1x1/1 (x1000)	1					**20%** (3x3)		6bit	513,000	103,400
avgpool10	1x1x1000	13x13/1	0									
	activations			parameters				compression info			1,248,424 (total)	**421,098** (total)

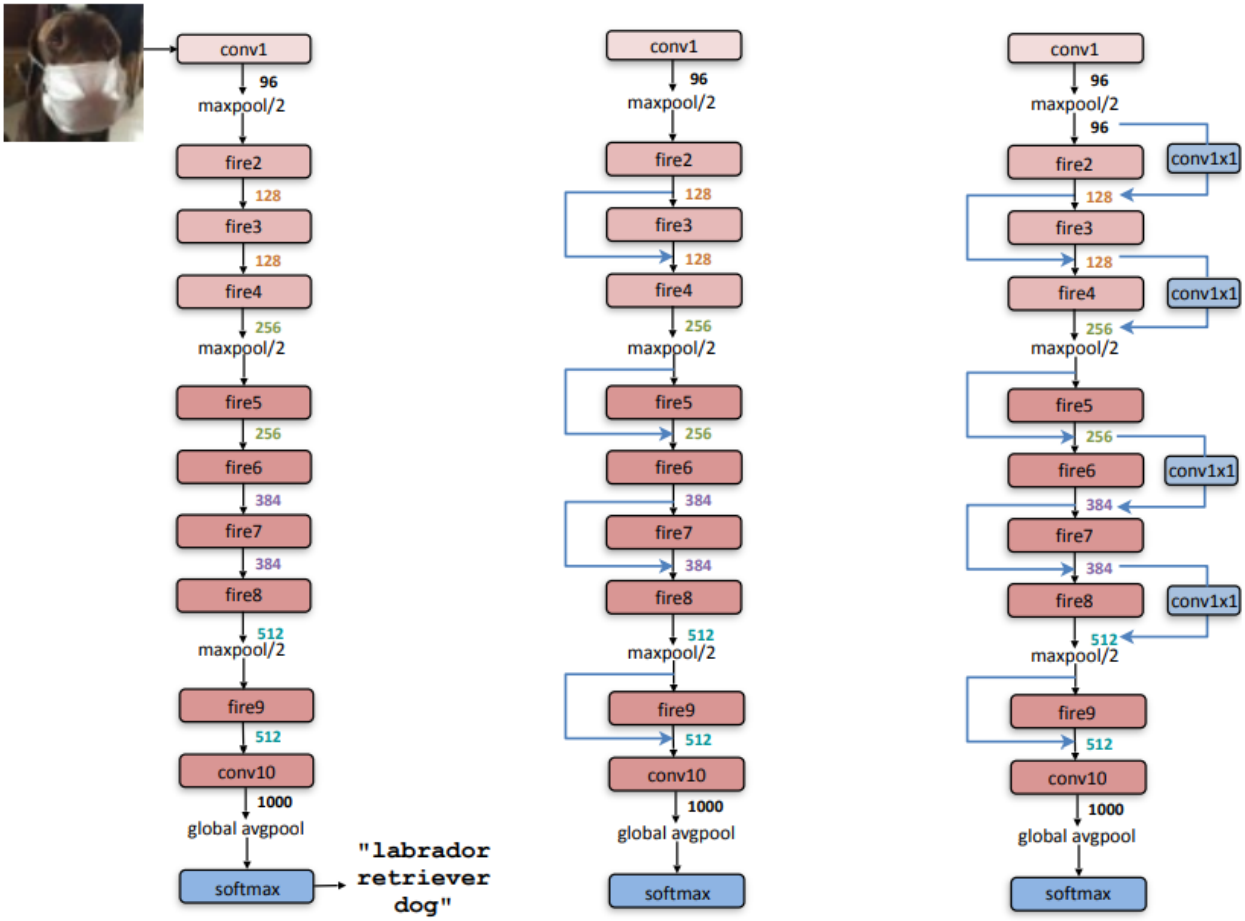

Рисунок 2.11 – Структура згорткової нейронної мережі SqueezeNet. Ліворуч: звичайна SqueezeNet, посередині: SqueezeNet з простим обходом, праворуч: SqueezeNet зі складним байпасом

При навчанні нейромережі отримана модель буде залежати не тільки від структури, але і від методу встановлення параметрів нейромережі. Сам по собі метод навчання може мати чимало гіперпараметрів. Параметри мережі оновлюються стохастичним градієнтним спуском із застосуванням градієнтного спуску на множині навчальних даних, які періодично перемішуються. Нехай t < T – це певна ітерація, тоді параметри θ оновлюються на даній ітерації, як показано в рівнянні 2.1:

$$\theta^{(t)} \leftarrow \theta^{(t-1)} - \epsilon_t \frac{1}{B} \sum_{t'=Bt+1}^{B(t+1)} \frac{\partial L(z_{t'}, \theta)}{\partial \theta}, \tag{2.1}$$

де zt` – це приклад навчальної множини. В даному випадку гіперпараметрами є: L - функція втрат, t – ϵ_t - темп навчання на ітерації, B - розмір пакету прикладів, та T - кількість ітерацій.

Темп навчання.

Темп навчання ϵ_t задає швидкість прямування градієнту оновлення до глобального мінімуму. Якщо темп навчання дуже малий, то модель буде сходитись дуже повільно, а якщо великий, тоді модель буде невірно натренована. Коли ϵ_t поступово змінюється з ітераціями, потрібно визначити початковий коефіцієнт швидкості навчання ϵ_0 та правило обчислення ϵ_t. Для нейромереж із нормованими вхідними даними або даними, які мають значення в діапазоні [0,1], ϵ_0 часто встановлюється в межах 1 та 10^{-6} , а найкраще значення для першої ітерації є 0,01. Проте, з огляду на важливість ϵ_0, цей параметр бажано налаштувати точно [17].

Темп навчання ϵ_t прийнято зменшувати зі зміною ітерації. Одним з підходів є ініціалізація $\epsilon_t=\epsilon_0$, де $t < \tau$ (τ - новий гіперпараметр), після чого нове значення коефіцієнту темпу навчання обчислюється за формулою $\epsilon t=\epsilon_0/t\alpha$ (α - ще один гіперпараметр). Потрібно використовувати малі значення α, якщо функція, на якій знаходиться глобальний мінімум, не є опуклою, і застосовується відповідний метод градієнтного усереднення (напр., імпульс). Загальноприйнятою практикою також є адаптивне встановлення значення τ залежно від ітерації.

Функція втрат.

Функція втрат порівнює відомі вже вихідні дані для навчального прикладу та значення, яке було отримане при навчанні. У загальному випадку функція втрат – це квадрат відстані між двома точками, яка знаходиться за рівнянням 2.2:

$$L = \frac{1}{2}\sum_i (y_i - z_i)^2 , \qquad (2.2)$$

де y_i – і-те значення вихідного шару нейромережі, а z_i - це і-те значення відгуку, що визначені в навчальній множині. Коефіцієнт 1/2 використовується для того, щоб зпростити градієнт. Коли вихідні значення нейромережі розглядаються як розподіл імовірності (наприклад, у вихідному шарі використовується Softmax), то загальним і доцільним теоретично є застосування як функції втрат перехресної ентропії, яка має вид (2.3):

$$L = -\sum_i y_i \log(z_i) \quad , \tag{2.3}$$

Число тренувальних ітерацій.

Часто T визначається за способом ранньої зупинки. Рання зупинка працює за принципом зупинки навчання після того, як похибка навчання на етапі валідації перестає зменшуватись (тобто похибка починає поступово рости). Це досить потужний засіб попередження явища перенавчання, так як налаштуваннями інших гіперпараметрів можна знехтувати.

Імпульс.

«Гладке» оновлення градієнта є найпоширенішою методикою, яка використовує інтеграційний фільтр разом з параметром β, як показано у (2.4):

$$\bar{g} \leftarrow (1-\beta)\bar{g} + \beta \frac{\partial L(z_t, \theta)}{\partial \theta} \quad , \tag{2.4}$$

де $\bar{g}$ може використовуватись замість «дійсного» оновлення градієнта при застосуванні методу градієнтного спуску. Окремі математичні підходи можуть надавати більш швидкий спуск градієнта при використанні певного імпульсу, однак для стохастичного градієнтного спуску, стандартне градієнтне оновлення (β = 1) з гармонійно зменшеною швидкістю навчання є оптимальнішим.

Сама структура нейромережі має в собі численні гіперпараметри, включно з розміром і нелінійністю кожного шару. Множинні властивості ваг також деяким чином обмежені, а їх ініціалізація може впливати сильно на показники продуктивності. Також може бути важливою для забезпечення спуску градієнта попередня обробка вхідних даних. Багато гіперпараметрів можуть варіюватися в різних шарах [16].

Кількість прихованих шарів. Велика кількість прихованих шарів дає змогу нейромережі добре навчитись, тому навіть при використанні регуляризації дуже важливим є створення прихованих шарів необхідної розмірності. Як правило, для всіх прихованих шарів використовують однакову розмірність, оскільки цей спосіб покращує роботу. При використанні неконтрольованого попереднього навчання, шари слід робити набагато більшими, ніж при виключно контрольованій оптимізації.

Відсікання вагових компонент

Часто, для того, щоб зменшити перенавчання, до критерію тренування додають регуляризацію ваг нейромережі. Для збереження значення ваг мережі θ близькими до нуля, регуляризація L2 додає $\lambda_2\Sigma_i\theta_i^2$, а L1 додає $\lambda_1\Sigma_i|\theta_i|$, де λ_2 та λ_1 - множники Лагранжа, які визначають важливість даної регуляризації.

2.4 Структура процесів обробки інформації при розпізнаванні порід котів

Завданням цієї роботи є створення інформаційної технології, яка розпізнає породу котів. Тому вхідними данними інформаційної технології є зображення кота. Подальші кроки обробки цього зображення представлені на рис. 2.12.

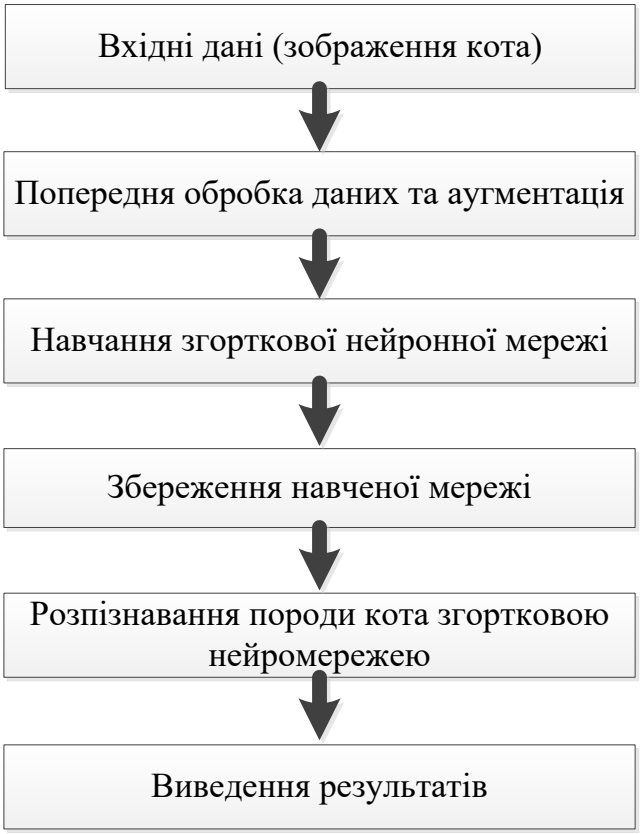

Рисунок 2.12 – Процеси обробки інформації при розпізнаванні порід котів

Всі зображення, що подаються на вхід нейромережі, мають пройти попередню обробку, яка полягає у приведенні зображення до єдиного розміру, який сприймає нейронна мережа. А ті зображення, на яких навчається нейромережа, ще мають піддаватися аугментації (внесення шуму та невеликих афінних перетворень) з метою збільшення ентропії навчальної інформації. Після цього здійснюється навчання нейронної мережі на розпізнавання порід котів. У підрозділі 2.1 ми з’ясували, що найперспективнішим є застосування згорткової нейромережі. Далі здійснюється збереження навченої нейромережі, після чого її можна використовувати для розпізнавання порід котів на вхідному зображенні. І, нарешті, виведення результату класифікації.

2.5 Розробка алгоритму розпізнавання порід котів

Згідно з результатами тестування ефективності роботи різних мереж у програмному середовищі Raspberry Pi, для поставлених цілей, розробки інформаційної технології розпізнавання порід котів за допомогою згорткової нейронної мережі, найкраще всього використовувати SqueezeNet v1.1.

SqueezeNet – це назва глибокої нейронної мережі для комп'ютерного зору, яка була випущена в 2016 році. SqueezeNet розроблено дослідниками з DeepScale, Каліфорнійського університету, Берклі та Стенфордського університету. При розробці SqueezeNet метою авторів було створення меншої нейронної мережі з меншою кількістю параметрів, яка могла б легше вписуватися в пам'ять комп'ютера та легше передаватися по комп'ютерній мережі.

Оригінальна версія SqueezeNet була реалізована на основі програмного забезпечення глибокого навчання Caffe. Незабаром після цього дослідницька спільнота з відкритим кодом перенесла SqueezeNet на ряд інших фреймворків глибокого навчання.

SqueezeNet постачається «власно» як частина вихідного коду ряду фреймворків глибокого навчання, таких як PyTorch, Apache MXNet та Apple CoreML. Крім того, сторонні розробники створили реалізації SqueezeNet, сумісні з такими фреймворками, як TensorFlow.

Відповідно до дослідженої інформації за темою даної роботи розробки інформаційної технології розпізнавання порід котів за допомогою згорткової нейронної мережі було розроблено алгоритм роботи програми, представлений на рис. 2.13.

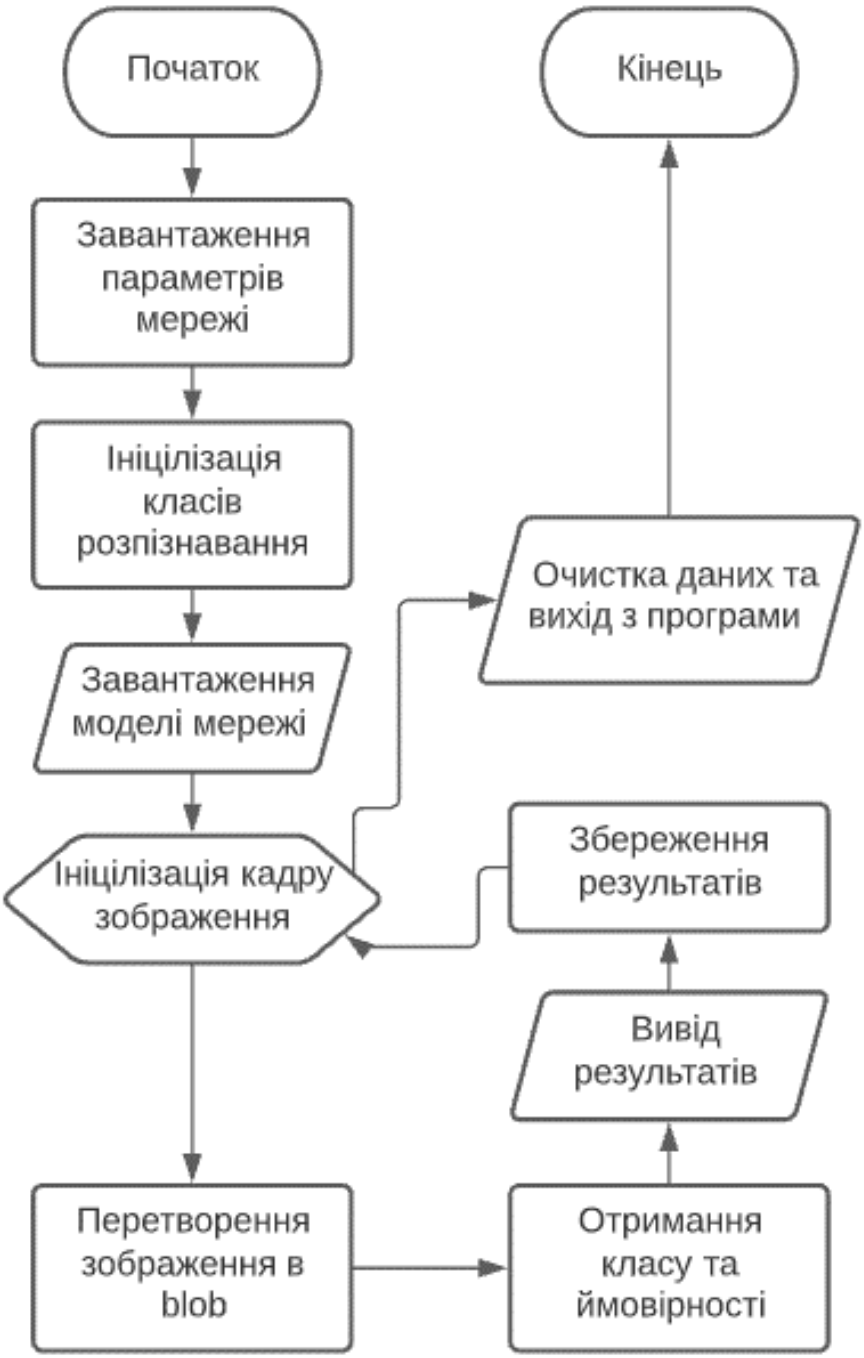

Рисунок 2.13 – Схема алгоритму розпізнавання порід котів

UML-діаграма класів розробленої програми розпізнавання порід котів представлена на рис. 2.14.

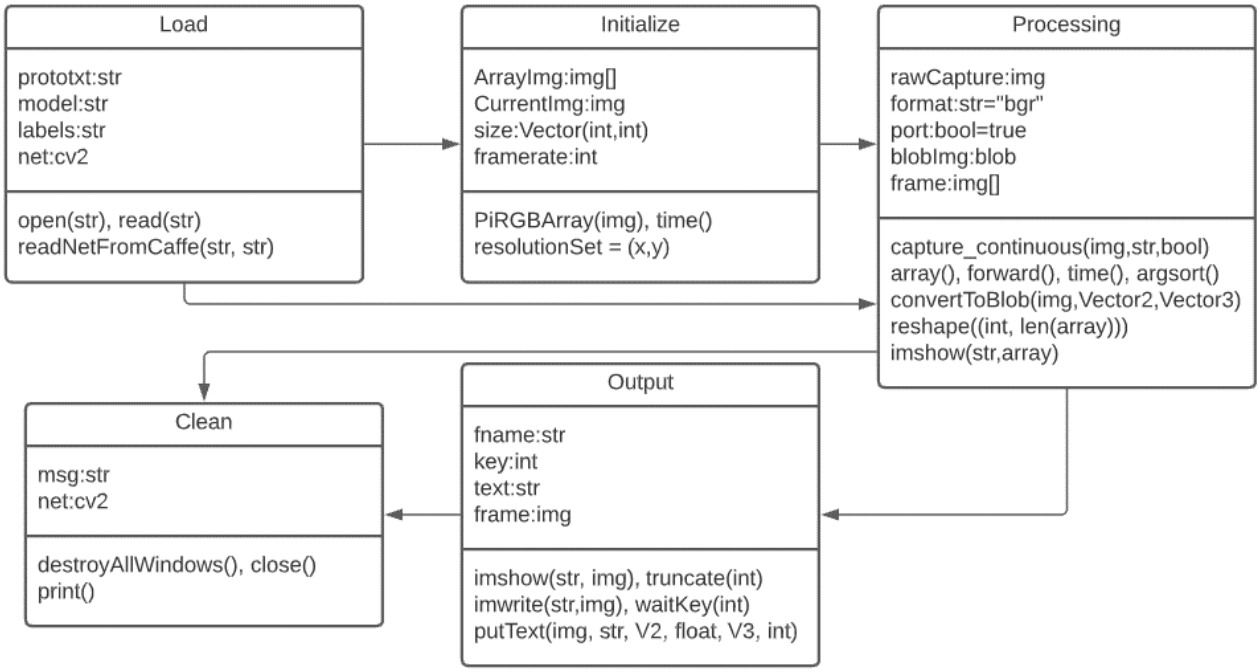

Рисунок 2.14 – UML-діаграма класів розпізнавання порід котів

З отриманої схеми алгоритму можна розкласти розробку програмного забезпечення розпізнавання порід котів за допомогою згорткової нейронної мережі на наступні ключові моменти:

- Підготовка – завантаження потрібних параметрів, класів та моделей головного модуля розпізнавання порід котів. На даному етапі оголошуються змінні та створюється відповідна модель нейронної мережі.

- Цикл обробки зображення – отримання та обробка зображення для подальшої роботи з ним. Основна робота із зображенням зводиться до трьох стратегій, що дозволяють збільшити ефективність роботи мережі: Згортки 3*3 замінені на згортки 1*1. Кожна така заміна у 9 разів зменшує кількість параметрів; На вхід згорток, що залишилися, 3*3 пробують подавати тільки невелике число каналів; Зменшення розміру робиться якнайпізніше, щоб згорткові шари мали велику площу активації. Ці стратегії називають “fire module” який можна подати у виді схеми, що зображено на рис. 2.15.

- Вивід, збереження та очистка сміття – відповідно до назви даний компонент виводить результат обробки зображення, зберігає його та

прибирає сміття, надлишкові данні та вікна, що були задіяні при роботі програмного забезпечення.

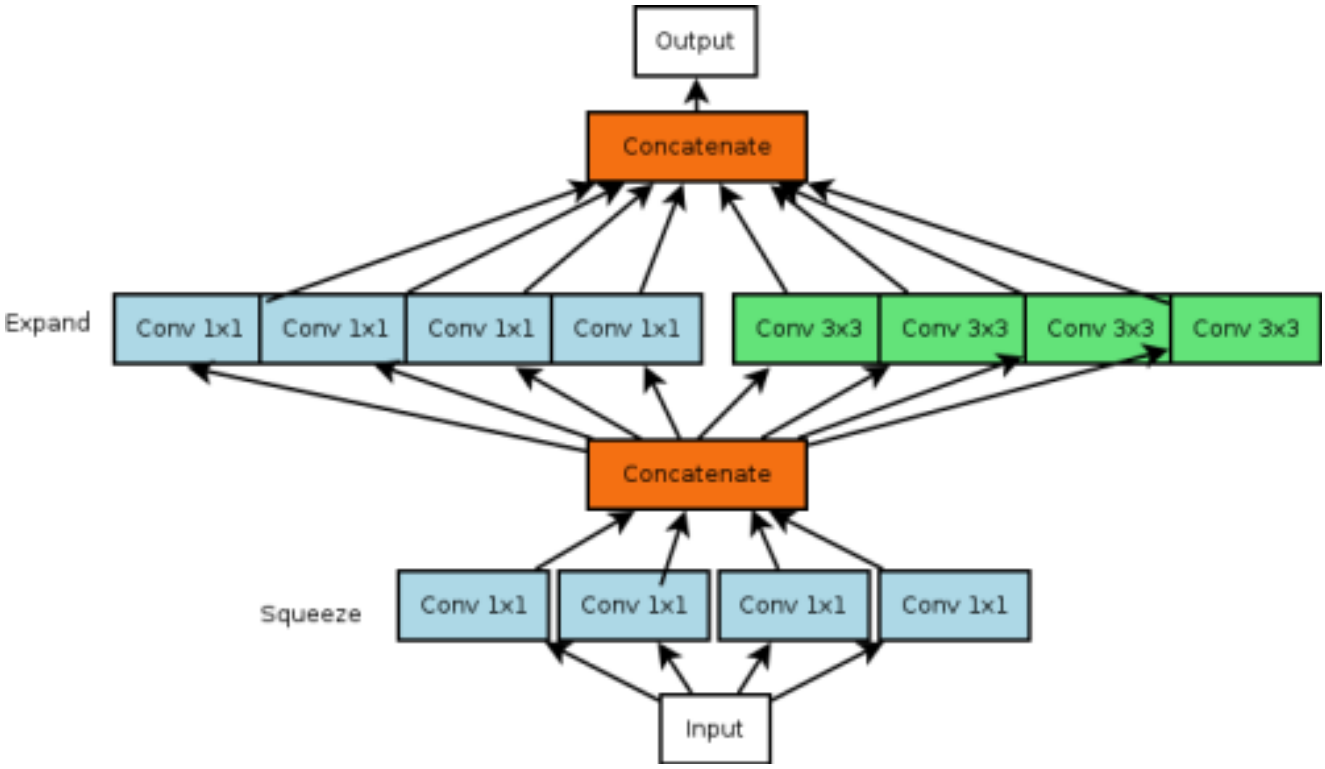

Рисунок 2.15 – Схема “fire module”

Створення основного компоненту програми розпізнавання порід котів за допомогою згорткової нейронної мережі можна розділити на певні етапи, згруповані відповідно до виду роботи.

- Підготовка набору даних – як описувалось у роботі, як набір даних доцільно використовувати ImageNet – фреймворк система для глибинного навчання CAFFE. Цей набір з усіх доступних має максимальну кількість класів 1000 шт., з них породи котів кількістю 35 (Абісинська, Австралійська димчаста, Азійська, Бобтейл, Керл, Арабська мау, Ашера, Бенгальська, Британська, Бурмила, Гімалайська, Донський сфінкс, Дракон Лі, Егейська, Європейська, Єгипетська мау, Йоркська шоколадна, Канаані, Кімрійська, Манчкін, Мейн-кун, Нібелунг, Німецький рекс, Оцикет, Перська, Піксибоб, Савання, Сіамська, Скотіш фолд, Сноу-шу, Сомалійська, Ангорська,

Турецький ван, Шартрез, Японський бобтейл) тому результати роботи нейромережі повинні бути коректними.

- Попередня обробка зображень (крок 1) – оскільки було вирішено в даній роботі використовувати програмне середовище Raspberry Pi, тому доцільно зменшити розмір зображень, відповідно зменшивши навантаження на Raspberry, таким чином можна залучити більше обчислювальної спроможності для роботи нейромережі.

- Попередня обробка зображень (крок 2) – поступове завантаження зображень. Даний крок буде повільніше виконуватися, але при завантаженні зображення великих розмірів дасть змогу Raspberry обробити його.

- Створення базової моделі – у цьому розділі ми можемо розробити базову модель згорткової нейронної мережі для набору даних котів. Базова модель встановить мінімальну продуктивність моделі, з якою можна порівняти всі інші наші моделі, а також архітектуру моделі, яку ми можемо використовувати як основу для вивчення та вдосконалення. Гарною відправною точкою є загальні архітектурні принципи моделей VGG.

- Вдосконалення моделі – є два підходи, щоб спробувати подолати переобладнаність: регуляризація відсіву та збільшення даних. Очікується, що обидва ці підходи сповільнять темпи покращення під час навчання та протидіють переобладнанню набору навчальних даних. Таким чином, буде збільшена кількість епох навчання з 20 до 50, щоб дати моделі більше простору для вдосконалення.

- Завершення – підготовка, збереження та тестування фінальної моделі програми розпізнавання порід котів за допомогою згорткової нейронної мережі.

2.6 Висновок до розділу 2

У розділі було обґрунтовано вибір типу згорткової нейронної мережі SqueezeNet v.1.1 для інформаційної технології розпізнавання порід котів. Спроектовано архітектуру згорткової нейронної мережі SqueezeNet, яка приймає вхідне зображення розміром 224x224x3 пікселів та може розпізнавати 35 порід котів. Розроблено структуру процесів обробки інформації інформаційної технології. Розроблено алгоритм роботи розпізнавання порід котів та UML діаграму класів, розглянуто особливості шару “fire module”.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ПОРІД КОТІВ НА ОСНОВІ ЗГОРТКОВОЇ НЕЙРОМЕРЕЖІ

3.1 Обґрунтування вибору мови та середовища програмування

Для реалізації програмного модуля розпізнавання порід котів за допомогою згорткової нейронної мережі обрано мову Python для програмного середовища Raspberry PI [12]. Через слабку швидкодію Raspberry було обрано мережу SqueezeNet v1.1.

Python [19] (вжива’ється прочитання - «Пайтон», назву запозичено з британського шоу Монті Пайтон) - це інтерпретована об'єктно-орієнтована мова програмування високого рівня із суворою динамічною типізацією. Була розроблена в 1990 році. Привабливою для швидкої розробки програм її роблять структури даних високого рівня разом із динамічною семантикою та зв'язуванням, а також поєднування наявних компонентів. Мова Python підтримує модулі та пакети модулів, Це сприяє модульності та надає можливість повторного використання коду. Інтерпретатор мови Python та його стандартні бібліотеки доступні і у скомпільованій, і у вихідній формі на всіх основних платформах. У мові Python підтримується декілька парадигм програмування, а саме: об'єктно-орієнтоване, процедурне, функціональне та аспектно-орієнтоване.

Переваги мови Python:

- чистий синтаксис (для виділення блоків потрібно застосовувати відступи);
- переносимість програм (властиве більшості інтерпретованих мов);
- стандартний дистрибутив вже має велике число корисних модулів (модуль для розробки графічного інтерфейсу включено);
- можливість користування Python-ом у діалоговому режимі (є дуже корисним для експериментів та розв'язування простих завдань);

- стандартний дистрибутив має просте, але одночасно досить потужне середовище для розробки, яке називається IDLE і яке написане на Python;
- зручний для вирішення математичних завдань (наявні засоби роботи з комплексними числами, оперує з цілими числами довільної довжини, може використовуватися у діалоговому режимі як ефективний калькулятор);
- відкритий код (інші користувачі можуть його редагувати).

Недоліки мови Python:

Низька швидкодія. Python, так само як і багато інших інтерпретованих мов, які не застосовують, JIT-компілятори, мають як загальний недолік порівняно низьку швидкодію виконання програмних кодів. Однак, у випадку мови Python цей недолік компенсується скороченням часу розробки програм. У середньому, програма, що написана на Python, у 2-4 рази менша, ніж її аналог на мові C++ або Java. Збереження байт-кодів (файли .pyc та .pyo) дозволяє інтерпретатору Python не втрачати зайвий час на перекомпіляцію кодів модулів при повторному запуску. Крім того, є спеціальна JIT-бібліотека psyco (хоча це призводить до збільшення споживання ресурсів оперативної пам'яті). Ефективність psyco значно залежить від архітектури програм.

Існують проекти реалізацій Python, які використовують високопродуктивні віртуальні машини (ВМ) як компілятори заднього плану. Прикладом таких реалізацій може бути PyPy, який базується на LLVM. Очікується, що застосування ВМ типу LLVM призведе до аналогічних результатів, що й використання подібних підходів при реалізації Java, де невелика обчислювальна продуктивність майже подолана. Наявність набору програм (бібліотек) для інтеграції з іншими мовами програмування дають можливість використовувати інші мови програмування для написання критичних ділянок програм.

У самій популярній реалізації Python інтерпретатор є досить великим і більш вимогливим до ресурсів комп`ютера, ніж в подібних популярних

реалізаціях Forth, Tcl, Lua або LISP, що дещо обмежує його застосування у вбудованих системах. Не зважаючи на це, Python знайшов застосування у планшетах та деяких моделях смартфонів.

Raspberry Pi - це одноплатний комп'ютер, який розроблено британським фондом Raspberry Pi Foundation. Його головне призначення - сприяти вивченню школярами базових комп'ютерних навичок.

Основою Raspberry Pi є система-на-чипі (SoC) Broadcom BCM2835, що містить процесор ARM із тактовою частотою 700 МГц, графічний процесор VideoCore IV і 512 або 256 мегабайт оперативної пам'яті. Жорсткого диску немає, натомість є SD карта. Таке апаратне забезпечення дозволяє відтворювати відео у форматі H.264 з роздільною здатністю 1080p, і запускати комп'ютерні ігри подібні до Quake III Arena.

Ініціатором проекту Raspberry Pi був британський благодійний фонд Raspberry Pi Foundation. Такий комп'ютер задумувався як пристрій для навчання школярів програмуванню, однак набув популярності і в інших галузях, зокрема, на його базі виготовляють домашні медіацентри. Найдешевший варіант Raspberry Pi постачається без корпуса і являє собою плату розміру не більше кредитної карти. Вага плати 45 грамів. У комп'ютері Raspberry Pi задіяно процесор 700 мегагерц з архітектурою ARM; є роз'єм для навушників та слот для карти пам'яті. Є молодша (A) і старша (B) моделі Raspberry Pi, які відмінні об'ємом оперативної пам'яті (відповідно 256 і 512 мегабайт) і кількістю USB-портів (відповідно один та два). Крім цього, старша модель має роз'єм Ethernet 10/100, а молодша споживає енергії на третину менше.

Старша модель Raspberry Pi вперше з`явилася у продажі в лютому 2012 року за 35 доларів США. У лютому 2013 надійшла в продаж у Європі молодша модель Raspberry Pi вартістю 25 доларів США. У період з лютого 2012 року було реалізовано понад мільйон комп'ютерів Raspberry Pi.

Для Raspberry Pi було випущено спеціалізований дистрибутив Linux, Raspbian OS (заснований на дистрибутиві Debian), створено магазин

застосунків Pi Store, де є і платні, і безплатні програми. Також Raspberry Pi використовує веб-браузер Iceweasel та KOffice, які постачаються у комплекті. Офіційна мова програмування Raspberry Pi для навчання — Python.

SqueezeNet – це назва глибокої нейронної мережі для комп'ютерного зору, яка була випущена в 2016 році. SqueezeNet розроблено дослідниками з DeepScale, Каліфорнійського університету, Берклі та Стенфордського університету. При розробці SqueezeNet метою авторів було створення меншої нейронної мережі, яка мала б меншу кількість параметрів,і яка могла б легше вписуватися в пам'ять комп'ютера та легше передаватися по комп'ютерній мережі.

Спочатку SqueezeNet був випущений 22 лютого 2016 року. Ця оригінальна версія SqueezeNet була реалізована на основі програмного забезпечення глибокого навчання Caffe. Незабаром після цього дослідницька спільнота з відкритим кодом перенесла SqueezeNet на ряд інших фреймворків глибокого навчання. 26 лютого 2016 року Едді Белл випустив порт SqueezeNet для платформи глибокого навчання Chainer. 2 березня 2016 року Гуо Харіа випустив порт SqueezeNet для фреймворку Apache MXNet. 3 червня 2016 року Таммі Янг випустила порт SqueezeNet для платформи Keras. У 2017 році такі компанії, як Baidu, Xilinx, Imagination Technologies і Synopsys, продемонстрували SqueezeNet, що працює на малопотужних платформах обробки, таких як смартфони, FPGA та користувацькі процесори.

Станом на 2018 рік, SqueezeNet постачається «власно» як частина вихідного коду ряду фреймворків глибокого навчання, таких як PyTorch, Apache MXNet та Apple CoreML.

3.2 Вибір бази даних для навчання та тестування згорткової нейронної мережі

Цифрова ера принесла з собою величезний вибух даних. За останніми підрахунками на Flickr розміщено понад 3 мільярди фотографій, така ж кількість відеокліпів на YouTube і навіть більша кількість зображень у базі даних Google Image Search. Використовуючи ці зображення, можна запропонувати більш складні та надійні моделі та алгоритми, що призведе до кращих програм для користувачів для індексування, пошуку, упорядкування та взаємодії з цими даними. Але як саме такі дані можуть бути використані та організовані, це проблема, яку ще належить вирішити. Було обрано нову базу даних зображень під назвою «ImageNet», великомасштабну онтологію зображень. Великомасштабна онтологія зображень є критично важливим ресурсом для розробки просунутих алгоритмів пошуку та розуміння зображень на основі великомасштабного вмісту, а також для забезпечення важливого навчання та порівняльних даних для таких алгоритмів.

ImageNet використовує ієрархічну структуру WordNet [13].

Кожне значуще поняття в WordNet, яке, можливо, описується у кількох словах або словосполученнях, носить назву «набір синонімів» або «синсет». У WordNet існує близько 80 000 синсетів іменників. У ImageNet надано в середньому 500-1000 зображень для ілюстрації кожного синсету. Зображення кожної концепції перевіряються якістю та анотуються людиною. Таким чином, ImageNet пропонує десятки мільйонів чітко відсортованих зображень. Поточна версія ImageNet складається з 12 «піддерев»: ссавець, птах, риба, рептилія, амфібія, транспортний засіб, меблі, музичний інструмент, геологічна формація, інструмент, квітка, фрукт. Ці піддерева містять 5247 синсетів і 3,2 мільйона зображень. На рис. 3.1 показано знімок двох гілок піддерев ссавців і транспортних засобів. База даних знаходиться у відкритому доступі [13].

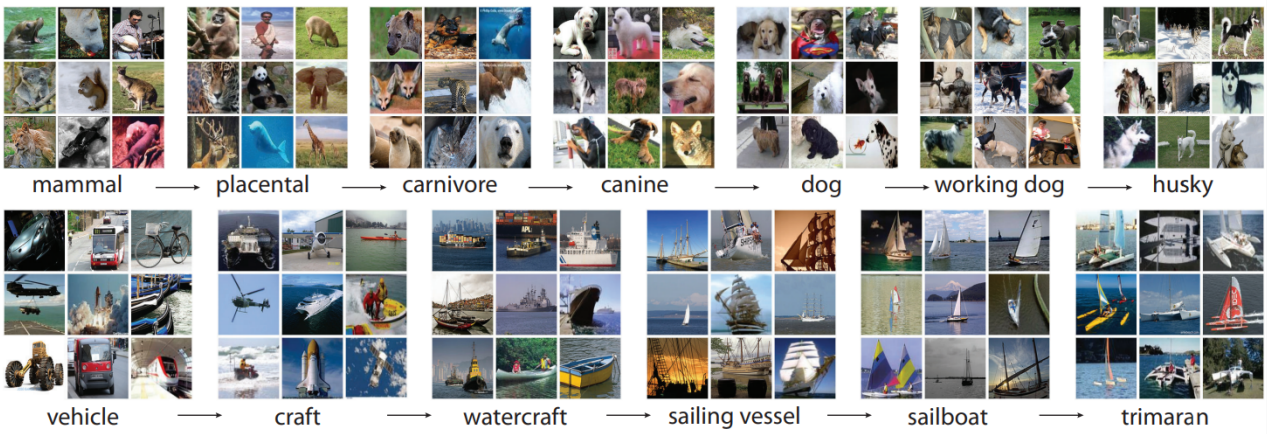

Рисунок 3.1 – Знімок двох гілок ImageNet від кореня до листа: верхній рядок із піддерева ссавців; нижній рядок — із піддерева транспортних засобыв. Для кожного синсету представлено 9 зображень із випадковою вибіркою

ImageNet побудовано на ієрархічній структурі WordNet. ImageNet містить близько 50 мільйонів чітко позначених зображень повної роздільної здатності (500-1000 на синсет). На поточний момент ImageNet складається з 12 піддерев. Більшість аналізів базуватиметься на піддеревах ссавців і транспортних засобів.

ImageNet має на меті забезпечити найбільш повне та різноманітне охоплення світу зображень. Поточні 12 піддерев складаються з 3,2 мільйона чітко анотованих зображень, розподілених по 5247 категоріям (рис. 3.2). У середньому для кожного синсету збирається понад 600 зображень. На рис. 3.2 показано розподіли кількості зображень на синсет для поточноъ версії ImageNet 1. Наскільки відомо, це вже найбільший чистий набір даних зображень, доступний спільноті дослідників комп`ютерного зору, як по загальній кількості зображень, так і по кількості зображень на категорію, а також по кількості категорій.

У ImageNet в середньому досягається точність маркування зображень 99,7%. Досягти високої точності для всіх глибин дерева ImageNet є складним завданням, оскільки чим нижче в ієрархії знаходиться синсет, тим важче його класифікувати, наприклад, Сіамська кицька проти бірманської кицьки.

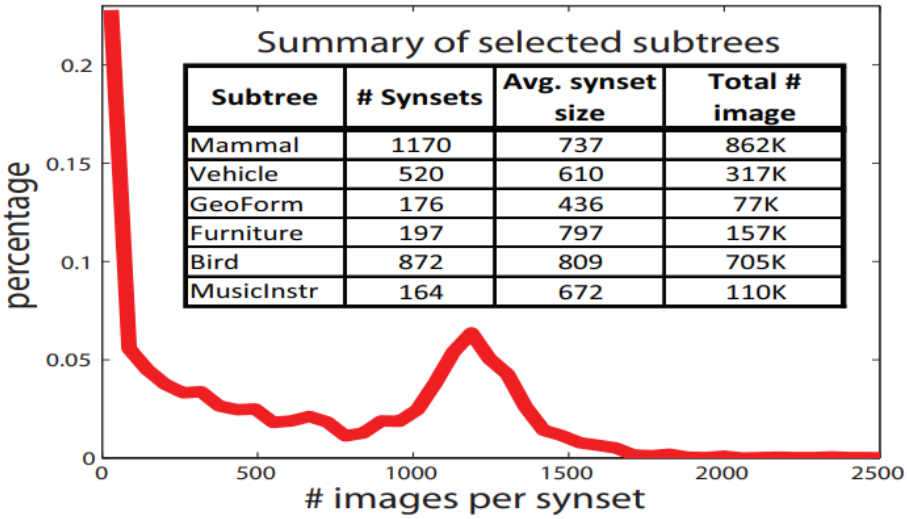

Рисунок 3.2 – Масштаб ImageNet. Червона крива: гістограма кількості зображень на синсет. Близько 20% синсетів мають дуже мало зображень. Понад 50% синсетів мають понад 500 зображень. Таблиця: Зведення вибраних піддерев

ImageNet створено так, щоб об’єкти на зображеннях мали різний зовнішній вигляд, положення, точки огляду, пози, а також фоновий безлад і оклюзії. Намагаючись вирішити складну проблему кількісного визначення різноманітності зображень, обчислюємо середнє зображення кожного синсету та вимірюємо розмір файлу JPG без втрат, який відображає кількість інформації в зображенні. Ідея полягає в тому, що синсет, що містить різноманітні зображення, призведе до більш розмитого середнього зображення, екстремальним буде сіре зображення, тоді як синсет із невеликою різноманітністю призведе до більш структурованого, чіткішого середнього зображення. Тому очікується побачити менший розмір файлу JPG середнього зображення більш різноманітного синсету.

ImageNet не має собі рівних. Наприклад, жоден існуючий набір даних зору не пропонує 147 категорій зображень собак. Що стосується котів, то у ImageNet 35 категорій (порід) котів (Абісинська, Австралійська димчаста, Азійська, Бобтейл, Керл, Арабська мау, Ашера, Бенгальська, Британська,

Бурмила, Гімалайська, Донський сфінкс, Дракон Лі, Егейська, Європейська, Єгипетська мау, Йоркська шоколадна, Канаані, Кімрійська, Манчкін, Мейн-кун, Нібелунг, Німецький рекс, Оцикет, Перська, Піксибоб, Саванна, Сіамська, Скотіш фолд, Сноу-шу, Сомалійська, Ангорська, Турецький ван, Шартрез, Японський бобтейл). тому для навчання та тестування згорткової нейронної мережі у даній роботі було обрано саме базу даних ImageNet.

3.3 Реалізація програми розпізнавання порід котів

Першим кроком в програмному забезпеченні розпізнавання порід котів за допомогою згорткової нейронної мережі буде ініціалізація компонентів програми. Для цього потрібно завантажити:

Параметри мережі:

```
rototxt = 'models/squeezenet_v1.1.prototxt'
model = 'models/squeezenet_v1.1.caffemodel'
labels = 'models/synset_words.txt'
```

Класи розпізнавання:

```
rows = open(labels).read().strip().split("\n")
classes = [r[r.find(" ") + 1:].split(",")[0] for r in rows]
```

Модель мережі:

```
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(prototxt, model)
```

Далі перетворення та цикл обробки зображення. Спочатку перетворюєм зображення на масив кадрів у raw форматі, та оброблюємо кожен елемент з масиву як кадр blob, завантажуючи відповідний елемент до нейромережі:

```
    rawCapture = PiRGBArray(camera)
    t0 = time.time()
    for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
        frame = rawCapture.array
        blob = cv2.dnn.blobFromImage(frame, 1, (224, 224), (104, 117,
124))
        net.setInput(blob)
        preds = net.forward()
        preds = preds.reshape((1, len(classes)))
        idxs = int(np.argsort(preds[0])[::-1][:1])
```

Останні кроки – вивід результату та очищення сміття. Розміщуємо інформацію про клас та ймовірність на відповідний кадр і зберігаємо отриманий результат.

```
    text = "Label: {}, p = {:.2f}%, fps =
{:.2f}".format(classes[idxs], preds[0][idxs] * 100, FPS)
        cv2.putText(frame, text, (5, 25), cv2.FONT_HERSHEY_SIMPLEX,
0.7, (0, 0, 255), 2)
        print(text)
        cv2.imshow("Frame", frame)
        fname = 'pic_' + dt.datetime.now().strftime('%Y-%m-%d_%H-%M-
%S') + '.jpg'
        cv2.imwrite(fname, frame)
        key = cv2.waitKey(1) & 0xFF
```

Розмір пакету прикладів при навчанні. Проводились експерименти з розмірами пакетів від 32 до 1024. У цій реалізації розмір пакета за замовчуванням становить 512. Якщо наївно реалізувати на одному GPU, такий великий розмір пакета може призвести до браку пам’яті. Ефективним обхідним шляхом є використання ієрархічного пакетування (іноді його називають

«відкладене пакетування»). Система Caffe підтримує ієрархічне пакетування, виконуючи навчальні зразки train_val.prototxt>batch_size одночасно в пам’яті. Після ітерацій solver.prototxt>iter_size градієнти підсумовуються, а модель оновлюється. З математичної точки зору розмір пакету дорівнює batch_size * iter_size. У включених файлах prototxt ми встановили (batch_size=32, iter_size=16), але будь-яка комбінація batch_size та iter_size, яка множиться на 512, дасть еквівалентні результати. Насправді, з тим самим початковим кодом генератора випадкових чисел, модель буде повністю відтворюваною, якщо її навчити кілька разів. Насамкінець, в системі Caffe iter_size застосовується під час навчання на навчальному наборі, але не під час тестування на тестовому наборі.

Впровадження модулів Fire. У 2 розділі описанно розширену частину шару Fire як набір фільтрів 1x1 і 3x3. Система Caffe спочатку не підтримує шар згортки, який має кілька розмірів фільтра. Щоб вирішити цю проблему, було реалізуовано шари expand1x1 і expand3x3 і об’єднано результати разом у розмірі каналу.

3.4 Висновок до розділу 3

У розділі було обґрунтовано вибір мови Python та програмного середовища Raspberry PI для програмної реалізації інформаційної технології розпізнавання порід котів. Як базу даних для навчання та тестування згорткової нейронної мережі було обрано набір зображень ImageNet . Описано основні етапи програмної реалізація та функціонування розпізнавання порід котів на основі згорткової нейронної мережі, що складається з ініціалізації компонентів програми, завантаження параметрів нейронної мережі, класів розпізнавання, моделі мережі, перетворення та циклу обробки зображення, виведення результату та очищення сміття.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМИ РОЗПІЗНАВАННЯ ПОРІД КОТІВ НА ОСНОВІ ЗГОРТКОВОЇ НЕЙРОМЕРЕЖІ

Результатом цієї роботи є програмний додаток, що розпізнає породу кота по його зображенню. Для того щоб впевнитися, що програма працює правильно, проведемо її тестування. Для цього потрібно лише включити Raspberry Pi та подати потік зображень до програми. Це можна робити за допомогою відео або завчасно підготовленому спеціальному архіві із зображеннями.

Підключивши Raspberry Pi, відкриваєм його консоль (рис. 4.1).

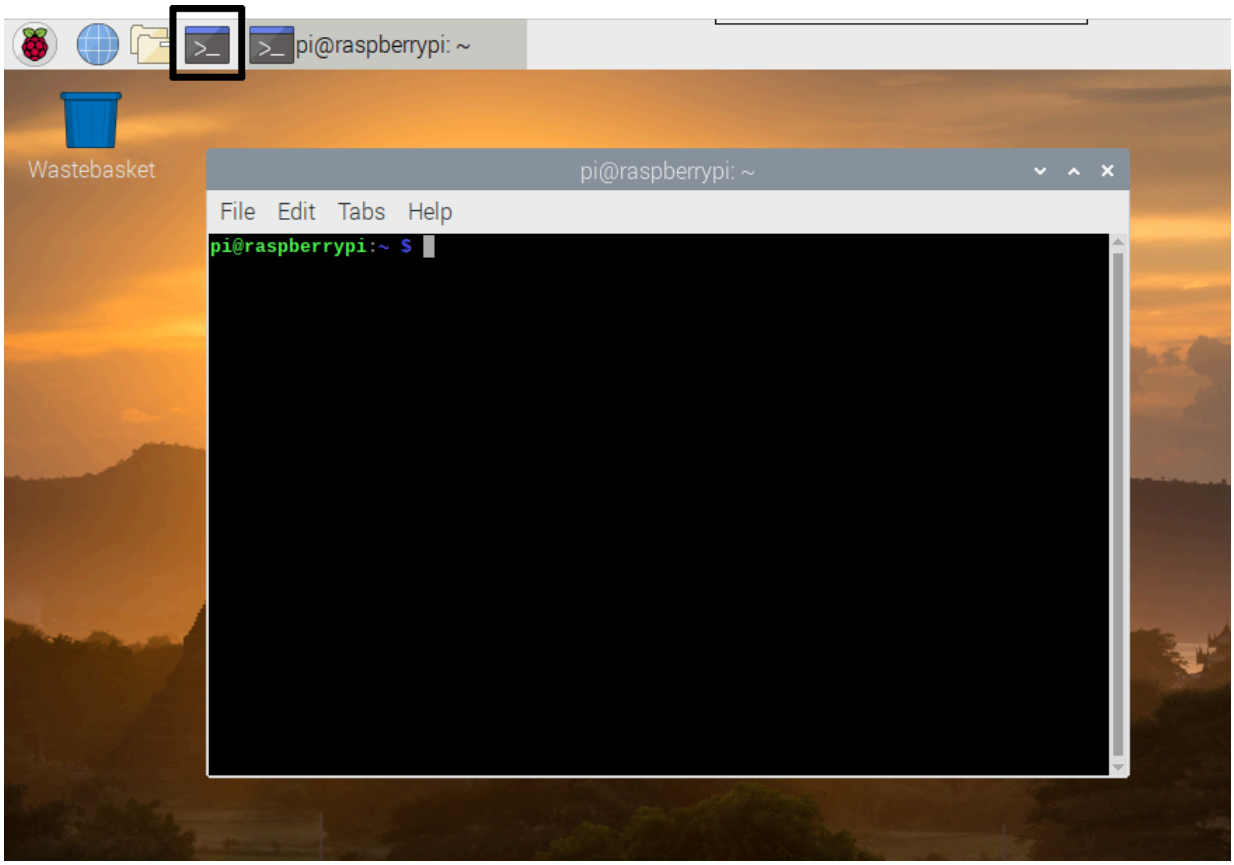

Рисунок 4.1 – Зображення консолі у програмному середовищі Raspberry Pi

Raspberry Pi працює на операційній системі Linux. Тому використовуючи відомі команди завантажуємо необхідні для роботи програмного модуля компоненти та ініціалізуємо їх для даного програмного середовища (рис. 4.2).

```
root@raspberrypi:~# apt-get install screen
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  screen
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 616 kB of archives.
After this operation, 1,024 kB of additional disk space will be used.
Get:1 http://ftp.uk.debian.org/debian/ squeeze/main screen armel 4.0.3-1
6 kB]
Fetched 616 kB in 0s (1,194 kB/s)
Selecting previously deselected package screen.
(Reading database ... 49685 files and directories currently installed.)
Unpacking screen (from .../screen_4.0.3-14+b1_armel.deb) ...
Processing triggers for man-db ...
Processing triggers for install-info ...
Setting up screen (4.0.3-14+b1) ...
root@raspberrypi:~# screen bash
```

Рисунок 4.2 – Завантаження та ініціалізація програмних компонентів

Використовуючи заздалегідь підготовлений архів зображень, подаємо їх до розробленої програми розпізнавання порід котів за допомогою згорткової нейронної мережі.

Приклад розпізнавання породи кота представлений на рис. 4.3. Із даних результатів видно, що згорткова нейромережа визначила на зображенні породу кота як перську з ймовірністю 92%, інші схожі породи та імовірності їх схожості подані нижче. Спробуємо обманути мережу, подавши зображення цього ж кота, але в іншому положенні (див. рис. 4.4).

Рисунок 4.3 – Приклад результату роботи програми

Рисунок 4.4 – Інший приклад результату роботи програми

В результаті обман не вдався. Дана програма розпізнавання порід котів за допомогою згорткової нейронної мережі визначив породу даного кота як перську, але тепер вже з імовірністю в 99%.

Змінимо кота так, щоб мережі було складніше розпізнати породу кота, вибираємо зображення з іншими об’єктами і так, щоб кота було видно не повністю (див. рис. 4.5).

Рисунок 4.5 – Наступний результат роботи програми

На даному етапі мережа розпізнала, що порода даного кота – сіамська, з ймовірністю в 66%. Тепер стає зрозумілим, що мережа здає позиції при поданні зображення з нечіткими контурами. Тому наступним кроком, для ускладнення «життя» мережі, зменшуємо кота, подаючи зображення лише голови кота (див. рис. 4.6).

Рисунок 4.6 – Результат роботи програми

Таким чином, «впевненість» мережі зменшилась ще більше. В даному випадку ймовірність того, що на зображенні саме сіамська кішка, зменшилась на 40%, а саме 26%.

Отже, можна сказати, що нейронна мережа зі своєю задачею розпізнавання порід котів за допомогою згорткової нейронної мережі справляється. Але для більш точних результатів потрібно виконувати наступні умови:

- бажано, щоб фон був контрастний та однорідний;

- кіт має бути представлений на фото у повний зріст, бажано мордою до камери;

- кіт повинен займати більшу частину площі зображення;

- на зображенні не повинно бути ніяких об’єктів, рівних за розміром до кота або навіть більших.

Розроблену програму розпізнавання порід котів було навчено з використанням бази даних зображень ImageNet [13]. ImageNet — це анотована база даних зображень, яка містить не менше ніж по 1000 зображень

для кожного із 100000 класів, причому кожне зображення описано кількома словами або словосполученнями.

Навчальна вибірка складалась із 2800 зображень (по 80 зображень на кожну із 35 порід котів). Тестова вибірка складалась із 700 зображень (по 20 зображень на кожну із 35 порід котів). Результати тестування запропонованого програмного модуля наведено у табл. 4.1.

Таблиця 4.1 – Порівняння параметрів розробленої програми із програмою-аналогом Cat Scanner

	Кількість зображень у тестовій вибірці	Кількість вірно розпізнанних зображень	Достовірність розпізнавання
Cat Scanner	700	629	89,9 %
Розроблене програмне забезпечення	700	660	94,3%

Із табл. 4.1 видно, що розроблене програмне забезпечення має достовірність розпізнавання порід котів 94,3%, а програма-аналог Cat Scanner має достовірність розпізнавання порід котів 89,9%.

Таким чином, розроблене програмне забезпечення має порівняно з програмою-аналогом Cat Scanner збільшену на 4,4% достовірність розпізнавання порід котів. Тобто мета роботи досягнута – достовірність розпізнавання підвищена.

Висновок до розділу 4

У четвертому розділі в результаті тестування програми було доведено її повну працездатність та відповідність поставленому завданню. Навчання

згорткової нейронної мережі відбувалось з використанням бази даних зображень ImageNet. Навчальна вибірка складалась із 2800 зображень (по 80 зображень на кожну із 35 порід котів). Тестова вибірка складалась із 700 зображень (по 20 зображень на кожну із 35 порід котів). Розроблена програма має достовірність розпізнавання порід котів 94,3%, а програма-аналог Cat Scanner має достовірність розпізнавання порід котів 89,9%. Таким чином, розроблена програмна має порівняно з програмою-аналогом Cat Scanner збільшену на 4,4% достовірність розпізнавання порід котів. Тобто мета роботи досягнута – достовірність розпізнавання підвищена.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення комерційного та технологічного аудиту інформаційної технології розпізнавання порід котів на основі згорткової нейронної мережі.

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедри [20].

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 5.1.

Таблиця 5.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу засобу поляриметричного аналізу оптично активних рідни

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Технічна здійсненність концепції	2	2	3
Ринкові переваги (наявність аналогів)	3	2	2
Ринкові переваги (ціна продукту)	2	3	3
Ринкові переваги (технічні властивості)	3	2	3
Ринкові переваги (експлуатаційні витрати)	2	2	3
Ринкові перспективи (розмір ринку)	2	3	3
Ринкові перспективи (конкуренція)	3	3	2
Практична здійсненність (наявність фахівців)	4	3	2

Продовження таблиці 5.1

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Практична здійсненність (наявність фахівців)	4	3	2
Практична здійсненність (наявність фінансів)	3	2	2
Практична здійсненність (необхідність нових матеріалів)	2	3	2
Практична здійсненність (термін реалізації)	2	3	3
Практична здійсненність (розробка документів)	3	2	2
Сума балів	31	30	30
Середньоарифметична сума балів, СБ	30		

За результатами розрахунків, наведених в таблиці 5.1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інформаційної технології розпізнавання порід котів на основі згорткової нейронної мережі – середній.

5.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати на оплату праці. Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці» [21].

Основна заробітна плата дослідників. Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р},$$

де к – кількість посад дослідників, залучених до процесу дослідження; $М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.; $Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні; t_i – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 5.2.

Таблиця 5.2 – Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	38000	1810	6	10860
Розробник	33000	1571	25	39275
Всього:	50135			

Додаткова заробітна плата. Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}1 \cdot (50135 + 0) = 5013{,}5 \text{ грн.} \qquad (5.1)$$

Відрахування на соціальні заходи. Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) =$$
$$= 0{,}22 \cdot (50135 + 0 + 5013{,}5) = 12133 \text{ грн.} \quad (5.2)$$

де $З_о$ – основна заробітна плата розробників, грн.; $З_р$ – основна заробітна плата робітників, грн.; $З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.; β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

Програмне забезпечення. До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{1}^{к} Ц_{іпрг} \cdot С_{прг.і} \cdot К_і, \quad (5.3)$$

де $Ц_{іпрг}$ – ціна придбання програмного забезпечення і-го виду, грн.; $С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного виду, шт.; $К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_і$ = (1,1…1,12); к – кількість видів програмного забезпечення.

Таблиця 5.3 – Витрати на придбання програмного забезпечення

Найменування програмного забезпечення	Ціна за одиницю, грн.	Витрачено	Вартість програмного забезпечення, грн.
Raspberry Pi IDE Geany	0	1	0
Всього, з врахуванням коефіцієнта інсталяції та налагодження			0

Амортизація обладнання. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$\text{А} = \frac{\text{Ц}_\text{б}}{\text{Т}_\text{в}} \cdot \frac{t}{12}, \tag{5.4}$$

де $\text{Ц}_\text{б}$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.; t – термін використання основного фонду, місяці; $\text{Т}_\text{в}$ – термін корисного використання основного фонду, роки.

Таблиця 5.4 – Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Ноутбук	34000	5	1,1	623
Всього	623			

Витрати на електроенергію для науково-виробничих цілей. Витрати на силову електроенергію $\text{В}_\text{е}$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

Таблиця 5.5 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Ноутбук	0,13	170

$$\text{Ве} = \sum \frac{W_i \cdot t_i \cdot \text{Ц}_\text{е} \cdot \text{К}_\text{впі}}{\text{ККД}} = \frac{0{,}13 \cdot 170 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} = 161 \text{ грн.}, \qquad (5.5)$$

W_i – встановлена потужність обладнання, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год.; Це – вартість 1 кВт електроенергії, грн.; $К_{впі}$ – коефіцієнт використання потужності; ККД – коефіцієнт корисної дії обладнання.

Інші витрати. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{Ів} = \left(З_\text{о} + З_\text{р}\right) \cdot \frac{\text{Н}_\text{ів}}{100\%} = (50135 + 0) \cdot \frac{75}{100} = 37601 \text{ грн.}, \qquad (5.6)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

Накладні (загальновиробничі) витрати. До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{Внзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%} = (50135 + 0) \cdot \frac{150}{100} = 75203 \text{ грн.}, \qquad (5.7)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення науково-дослідної роботи. Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + В_{прг} + А_{обл} + В_е +$$
$$+ I_в + В_{нзв} = 50135 + 0 + 5013{,}5 + 12133 + 0 + 623 + 161 + 37601 + 75203 =$$
$$= 180870 \text{ грн.} \qquad (5.8)$$

Загальні витрати. Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{180870}{0{,}9} = 200967 \text{ грн.}, \qquad (5.9)$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи. Оскільки, якщо науково-технічна розробка знаходиться на стадії технічного впровадження, то η=0,9.

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження

результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

В даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N_i)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \qquad (5.10)$$

де ±ΔЦ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки); $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни); N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки; $Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році; $Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової)

науково-технічної розробки у році до впровадження результатів; ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки); λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 1 рік, тому:

$$\Delta П = ((3400 - 3000) \cdot 8000 - (8000 - 8000) \cdot 3000) \cdot 0{,}8333 \cdot 0{,}3 \cdot \left(1 - \frac{18}{100}\right) = 719972 \text{ грн.} \quad (5.11)$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{719972}{(1+0{,}1)^1} = 647247 \text{ грн.}, \quad (5.12)$$

де ΔП – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.; Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо Т=1 рік); τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту

отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} \cdot \text{ЗВ} = 2 \cdot 200967 = 401934 \text{ грн.} \qquad (5.13)$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай кінв=2...5, але може бути і більшим; ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $\text{Е}_{\text{абс}}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$\text{Е}_{\text{абс}} = \text{ПП} - PV = 647247 - 401934 = 245313 \text{ грн.}, \qquad (5.14)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн.; PV – теперішня вартість початкових інвестицій, грн.

Оскільки $\text{Е}_{\text{абс}}>0$, то можемо припустити про потенційну зацікавленість інвесторів у розробці.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар’єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню

економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{Eабс}{PV}} = \sqrt[1]{1 + \frac{245313}{401934}} = 1{,}26, \qquad (5.16)$$

де $T_ж$ – життєвий цикл розробки, роки.

Визначимо бар'єрну ставку дисконтування $\tau_{мін}$, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f = 0{,}9 + 0{,}5 = 1{,}4, \qquad (5.17)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,9...0,12; f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f =0,05...0,5, але може бути і значно вищою.

Оскільки $E_в$=1,26>$\tau_{мін}$=1,4, то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок, тобто в її комерціалізації.

Далі розраховуємо період окупності інвестицій $T_о$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_о = \frac{1}{E_в} = \frac{1}{1,26} = 0,8 \text{ року.} \quad (5.18)$$

Оскільки $T_о$=0,8<1…3-х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

5.4 Висновок до розділу 5

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія розпізнавання порід котів на основі згорткової нейронної мережі» становить 30,0 балів, що вказує на комерційну вагомість проведення даних досліджень (рівень комерційного потенціалу розробки середній). Термін окупності становить 0,8 р., що менше 3-х років, що говорить про комерційну привабливість науково-технічної розробки і може спонукати потенційних інвесторів на фінансування впровадження цієї розробки та виведення її на ринок. Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія розпізнавання порід котів на основі згорткової нейронної мережі».

ВИСНОВКИ

У роботі було розглянуто задачу розпізнавання порід котів на основі згорткових нейромереж. В ході аналізу предметної області було розглянуто постановку задачі розпізнавання порід котів, проведено огляд відомих методів класифікації зображень, які можна використовувати для поставленої задачі. Обґрунтовано доцільність вибору методу розпізнавання зображень на основі нейронних мереж, зокрема згорткових нейронних мереж для розпізнавання порід котів. Було проаналізовано різні програмні додатки для розпізнавання порід котів та обрано аналог, головним недоліком якого є невисока достовірність розпізнавання порід котів.

У другому розділі було обґрунтовано вибір типу згорткової нейронної мережі SqueezeNet v.1.1 для інформаційної технології розпізнавання порід котів. Спроектовано архітектуру згорткової нейронної мережі SqueezeNet, яка приймає вхідне зображення розміром 224x224x3 пікселів та може розпізнавати 35 порід котів. Розроблено структуру процесів обробки інформації інформаційної технології. Розроблено алгоритм роботи розпізнавання порід котів та UML діаграму класів, розглянуто особливості шару “fire module”.

У третьому розділі було обґрунтовано вибір мови Python та програмного середовища Raspberry PI для програмної реалізації інформаційної технології розпізнавання порід котів. Як базу даних для навчання та тестування згорткової нейронної мережі було обрано набір зображень ImageNet. Описано основні етапи програмної реалізація та функціонування розпізнавання порід котів на основі згорткової нейронної мережі, що складається з ініціалізації компонентів програми, завантаження параметрів нейронної мережі, класів розпізнавання, моделі мережі, перетворення та циклу обробки зображення, виведення результату та очищення сміття.

У четвертому розділі в результаті тестування програми було доведено її повну працездатність та відповідність поставленому завданню. Навчання згорткової нейронної мережі відбувалось з використанням бази даних зображень ImageNet. Навчальна вибірка складалась із 2800 зображень (по 80 зображень на кожну із 35 порід котів). Тестова вибірка складалась із 700 зображень (по 20 зображень на кожну із 35 порід котів). Розроблена програма має достовірність розпізнавання порід котів 94,3%, а програма-аналог Cat Scanner має достовірність розпізнавання порід котів 89,9%. Таким чином, розроблена програмна має порівняно з програмою-аналогом Cat Scanner збільшену на 4,4% достовірність розпізнавання порід котів. Тобто мета роботи досягнута – достовірність розпізнавання підвищена.

У п`ятому розділі було визначено рівень комерційного потенціалу розробки за темою «Інформаційна технологія розпізнавання порід котів на основі згорткової нейронної мережі» становить 30,0 балів, що вказує на комерційну вагомість проведення даних досліджень (рівень комерційного потенціалу розробки середній). Термін окупності становить 0,8 р., що менше 3-х років, що говорить про комерційну привабливість науково-технічної розробки і може спонукати потенційних інвесторів на фінансування впровадження цієї розробки та виведення її на ринок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Я. О. Кузик, Ю. М. Паночишин, Інформаційна технологія розпізнавання порід котів на основі згорткової нейронної мережі в Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)», Вінниця, 2023, [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/19577/16225

2. Руденко О.В. Штучні нейронні мережі: Навчальний посібник / О.В.Руденко, Є.В.Бодянський. - Харків: ТОВ «Компанія СМІТ», 2006. — 404 с. - ISBN 966-8630-73-Х.

3. Куссуль Н. М. Інтелектуальні обчислення: навчальний посібник (із грифом МОН України) / Н. М. Куссуль, А. Ю. Шелестов, А. Н. Лавренюк. - К.: «Наукова думка», 2006. — 186 с. ISBN 966-00-0592-Х.

4. Гудфеллоу. Я. Глибоке навчання / пер. з. анг. А. А. Слинкіна. 2-ге вид., виправ. К..: ТОВ «Пресс», 2018. 652 с.:кольор.іл.

5. Machine vision [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Machine_vision.

6. Кононова К. Ю. Машинне навчання: методи та моделі: підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка» / К. Ю. Кононова. – Харків: ХНУ імені В. Н. Каразіна, 2020. – 301 с.

7. Обробка растрових зображень [Електронний ресурс]. Режим доступу: http://pzs.dstu.dp.ua/ComputerGraphics/raster/index.html

8. Нейронні мережі: теорія і практика [Електронний ресурс]. Режим доступу: https://intalent.pro/article/neyronnye-setiteoriya-i-praktika.html

9. Cat Scanner: Breed Recognition [Електронний ресурс]. Режим доступу: [https://siwalusoftware.com/ru/cat-scanner/].

10. Згорткова нейронна мережа [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Згорткова_нейронна_мережа.

11. Багаторівнева архітектура [Електронний ресурс] – режим доступу: https://metanit.com/sharp/mvc5/23.5.php.

12. Raspberry Pi. «What Linux distros will be supported at launch? Debian, Fedora and ArchLinux will be supported from the start.» [Електронний ресурс] – http://www.raspberrypi.org/faqs

13. Krizhevsky A. ImageNet Classification with Deep Convolutional Neural Networks [Електронний ресурс]. Режим доступу: https://papers.nips.cc/paper/4824imagenet-classification-with-deep-convolutional-neural-networks.pdf.

14. SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE / Forrest N. Iandola, Song Han, Matthew W. Moskewicz

15. Ganesh, Abhinav. "Deep Learning Reading Group: SqueezeNet". KDnuggets. Retrieved [Електронний ресурс] – https://www.kdnuggets.com/2016/09/deep-learning-reading-group-squeezenet.html.

16. "SqueezeNet". GitHub. [Електронний ресурс] – https://github.com/DeepScale/SqueezeNet

17. Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. arXiv:1312.4400, 2013/

18. Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. Systematic evaluation of CNN advances on the imagenet. arXiv:1606.02228, 2016.

19. Python [Електронний ресурс] – режим доступу: https://uk.wikipedia.org/wiki/Python

20. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

21. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

22. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. К. Колесницький. – Вінниця : ВНТУ, 2023. – (58 с.)

Додаток А (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія розпізнавання порід котів на основі згорткової нейронної мережі

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 93,23% Схожість 6,77%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Кузик Я.О.

Керівник роботи Паночишин Ю.М.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
from pic.array import PiRGBArray
import numpy as np
import time
from time import sleep
import datetime as dt
import cv2
prototxt = 'models/squeezenet_v1.1.prototxt'
model = 'models/squeezenet_v1.1.caffemodel'
labels = 'models/synset_words.txt'
rows = open(labels).read().strip().split("\n")
classes = [r[r.find(" ") + 1:].split(",")[0] for r in rows]
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(prototxt, model)
print("[INFO] starting video stream...")
rawCapture = PiRGBArray(camera)
t0 = time.time()
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
   frame = rawCapture.array
   blob = cv2.dnn.blobFromImage(frame, 1, (224, 224), (104, 117, 124))
   net.setInput(blob)
   preds = net.forward()
   preds = preds.reshape((1, len(classes)))
   idxs = int(np.argsort(preds[0])[::-1][:1])
   FPS = 1/(time.time() - t0)
   t0 = time.time()
   text = "Label: {}, p = {:.2f}%, fps = {:.2f}".format(classes[idxs], preds[0][idxs] * 100, FPS)
   cv2.putText(frame, text, (5, 25), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
   print(text)
   cv2.imshow("Frame", frame)
   fname = 'pic_' + dt.datetime.now().strftime('%Y-%m-%d_%H-%M-%S') + '.jpg'
   cv2.imwrite(fname, frame)
   key = cv2.waitKey(1) & 0xFF
   if key == ord("q"): break
   rawCapture.truncate(0)
print("[INFO] video stream is terminated")
cv2.destroyAllWindows()
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.2,
help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
```

```
CLASSES = ["Abyssinian", "Australian Smokey", "Asian", "Bobtail", "Curl", "Arabian Mau",
"Asher", "Bengal", "British", "Burmila", "Himalayan", "Don Sphynx", "Lee Dragon", "Aegean",
"European", "Egyptian Mau", "York Chocolate", "Canaan", "Cymraean", "Munchkin", "Maine
Coon ", "Nibelung", "German Rex", "Ociket", "Persian", "Pixiebob", "Savannah", "Siamese",
"Scottish Fold", "Snow Shoe", "Somali", "Angora", "Turkish Van", "Chartreuse", "Japanese
Bobtail"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)
fps = FPS().start()
while True:
frame = vs.read()
frame = imutils.resize(frame, width=400)
(h, w) = frame.shape[:2]
blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
0.007843, (300, 300), 127.5)
net.setInput(blob)
detections = net.forward()
for i in np.arange(0, detections.shape[2]):
confidence = detections[0, 0, i, 2]
if confidence > args["confidence"]:
idx = int(detections[0, 0, i, 1])
box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
(startX, startY, endX, endY) = box.astype("int")
label = "{}: {:.2f}%".format(CLASSES[idx],
confidence * 100)
cv2.rectangle(frame, (startX, startY), (endX, endY),
COLORS[idx], 2)
y = startY - 15 if startY - 15 > 15 else startY + 15
cv2.putText(frame, label, (startX, y),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF
if key == ord("q"):
break
fps.update()
fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
cv2.destroyAllWindows()
vs.stop()
from imutils.video import VideoStream
from imutils.video import FPS
from multiprocessing import Process
from multiprocessing import Queue
import numpy as np
```

```
import argparse
import imutils
import time
import cv2
def classify_frame(net, inputQueue, outputQueue):
# keep looping
while True:
# check to see if there is a frame in our input queue
if not inputQueue.empty():
# grab the frame from the input queue, resize it, and
# construct a blob from it
frame = inputQueue.get()
frame = cv2.resize(frame, (300, 300))
blob = cv2.dnn.blobFromImage(frame, 0.007843,
(300, 300), 127.5)
# set the blob as input to our deep learning object
# detector and obtain the detections
net.setInput(blob)
detections = net.forward()
# write the detections to the output queue
outputQueue.put(detections)
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.2,
help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
# initialize the list of class labels MobileNet SSD was trained to
# detect, then generate a set of bounding box colors for each class
CLASSES = [["Abyssinian", "Australian Smokey", "Asian", "Bobtail", "Curl", "Arabian Mau",
"Asher", "Bengal", "British", "Burmila", "Himalayan", "Don Sphynx", "Lee Dragon", "Aegean",
"European", "Egyptian Mau", "York Chocolate", "Canaan", "Cymraean", "Munchkin", "Maine
Coon ", "Nibelung", "German Rex", "Ociket", "Persian", "Pixiebob", "Savannah", "Siamese",
"Scottish Fold", "Snow Shoe", "Somali", "Angora", "Turkish Van", "Chartreuse", "Japanese
Bobtail"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
# initialize the input queue (frames), output queue (detections),
# and the list of actual detections returned by the child process
inputQueue = Queue(maxsize=1)
outputQueue = Queue(maxsize=1)
detections = None
# construct a child process *indepedent* from our main process of
# execution
```

```
print("[INFO] starting process...")
p = Process(target=classify_frame, args=(net, inputQueue,
outputQueue,))
p.daemon = True
p.start()
# initialize the video stream, allow the cammera sensor to warmup,
# and initialize the FPS counter
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
# vs = VideoStream(usePiCamera=True).start()
time.sleep(2.0)
fps = FPS().start()
# loop over the frames from the video stream
while True:
# grab the frame from the threaded video stream, resize it, and
# grab its dimensions
frame = vs.read()
frame = imutils.resize(frame, width=400)
(fH, fW) = frame.shape[:2]
# if the input queue *is* empty, give the current frame to
# classify
if inputQueue.empty():
inputQueue.put(frame)
# if the output queue *is not* empty, grab the detections
if not outputQueue.empty():
detections = outputQueue.get()
# check to see if our detectios are not None (and if so, we'll
# draw the detections on the frame)
if detections is not None:
# loop over the detections
for i in np.arange(0, detections.shape[2]):
# extract the confidence (i.e., probability) associated
# with the prediction
confidence = detections[0, 0, i, 2]
# filter out weak detections by ensuring the `confidence`
# is greater than the minimum confidence
if confidence < args["confidence"]:
continue
# otherwise, extract the index of the class label from
# the `detections`, then compute the (x, y)-coordinates
# of the bounding box for the object
idx = int(detections[0, 0, i, 1])
dims = np.array([fW, fH, fW, fH])
box = detections[0, 0, i, 3:7] * dims
(startX, startY, endX, endY) = box.astype("int")
# draw the prediction on the frame
label = "{}: {:.2f}%".format(CLASSES[idx],
confidence * 100)
cv2.rectangle(frame, (startX, startY), (endX, endY),
```

```
COLORS[idx], 2)
y = startY - 15 if startY - 15 > 15 else startY + 15
cv2.putText(frame, label, (startX, y),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
# show the output frame
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF
# if the `q` key was pressed, break from the loop
if key == ord("q"):
break
# update the FPS counter
fps.update()
# stop the timer and display FPS information
fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗПІЗНАВАННЯ ПОРІД КОТІВ НА ОСНОВІ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

Виконав: студент 2-го курсу,
групи 1КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Кузик Я. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
_____ Паночишин Ю.М.
(прізвище та ініціали)
«04» 12 2023 р.

Вінниця ВНТУ - 2023 рік

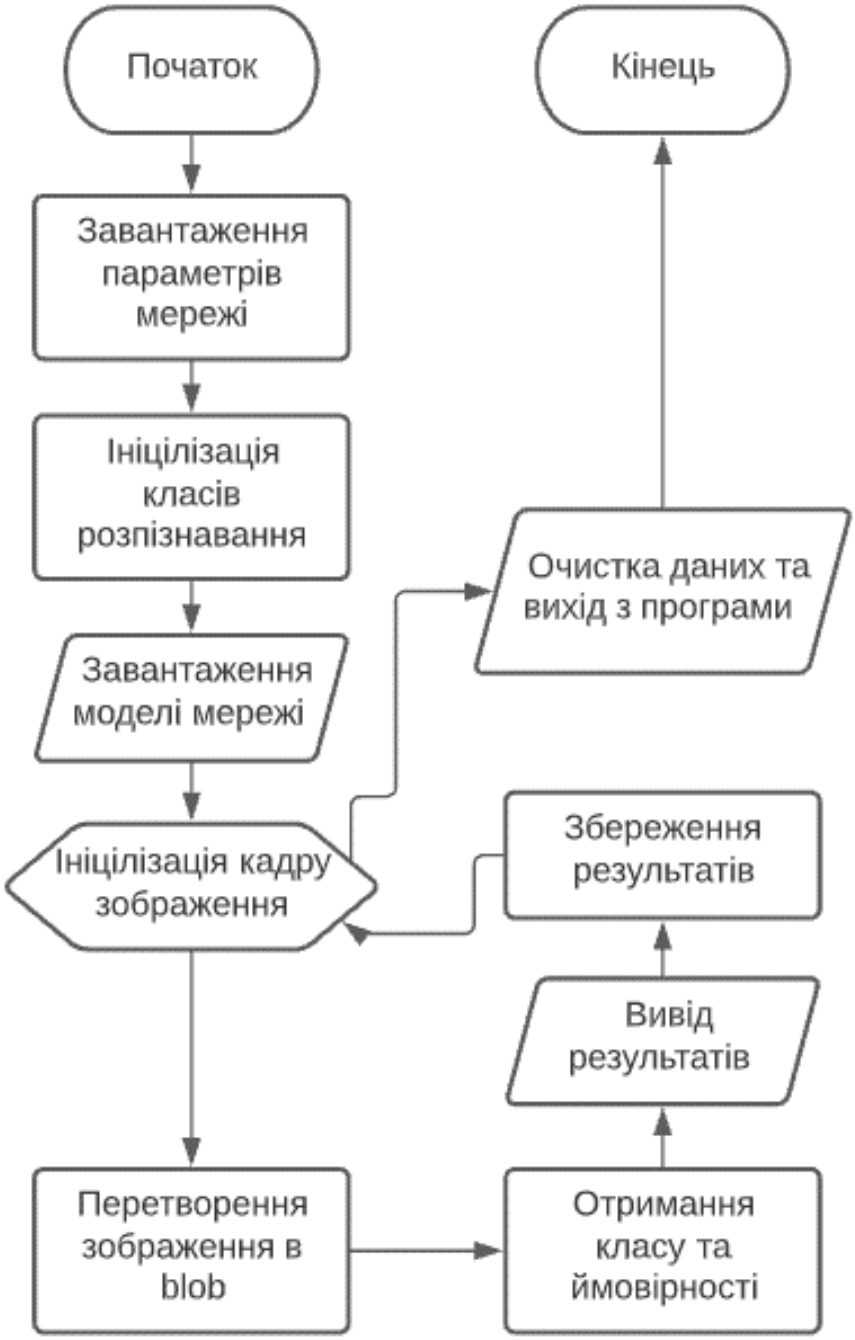

Рисунок В.1 –Загальний алгоритм роботи програми розпізнавання порід котів

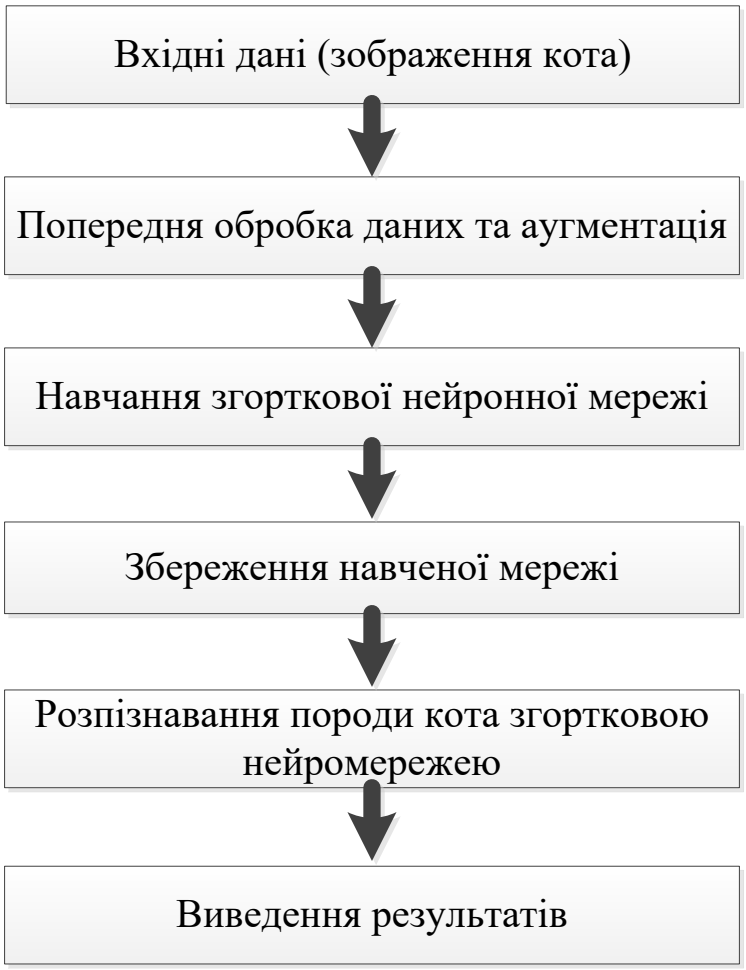

Рисунок В.2 – Структура інформаційної технології розпізнавання порід котів на основі згорткової нейронної мережі

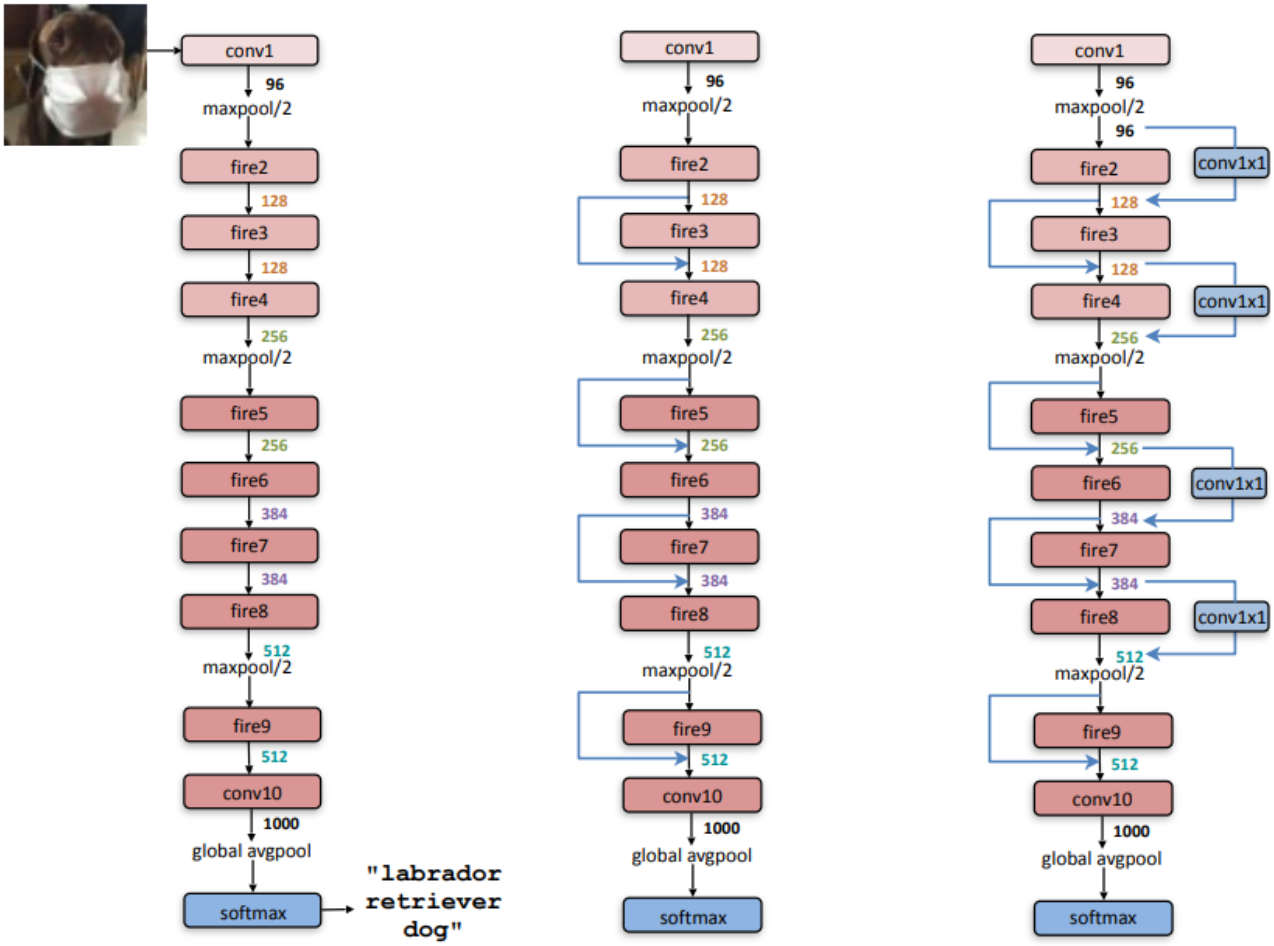

Рисунок В.3 – Структура згорткової нейронної мережі SqueezeNet. Ліворуч: звичайна SqueezeNet, посередині: SqueezeNet з простим обходом, праворуч: SqueezeNet зі складним байпасом

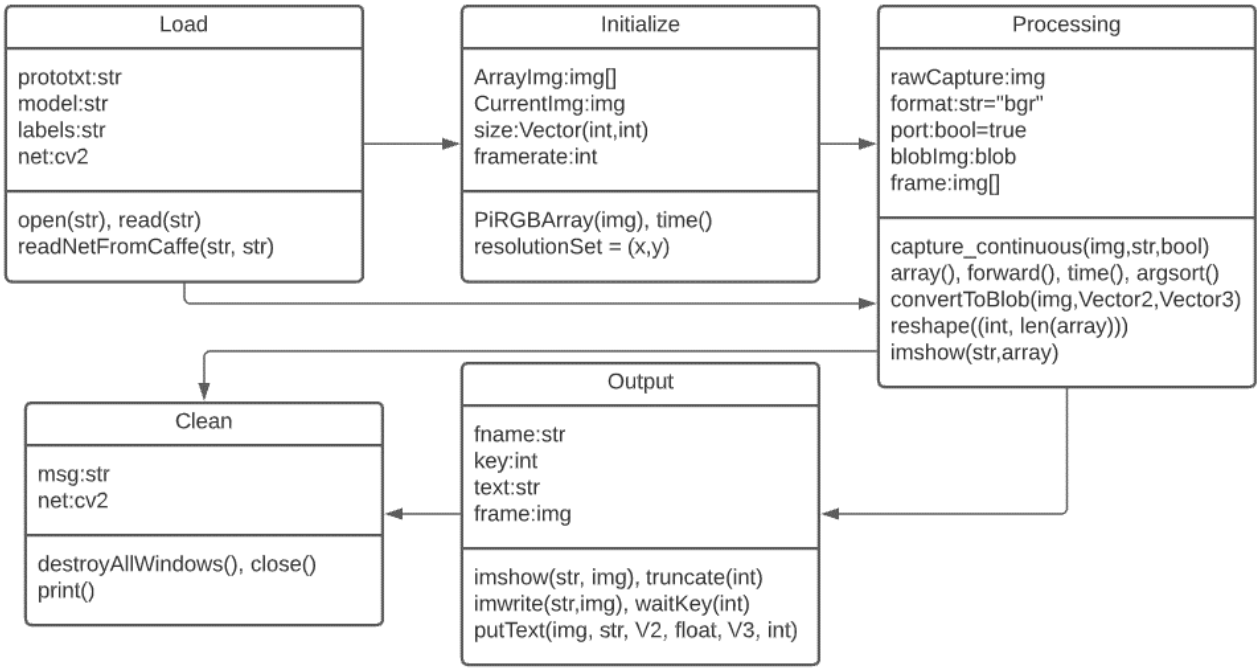

Рисунок В.4 – UML-діаграма класів програми розпізнавання порід котів

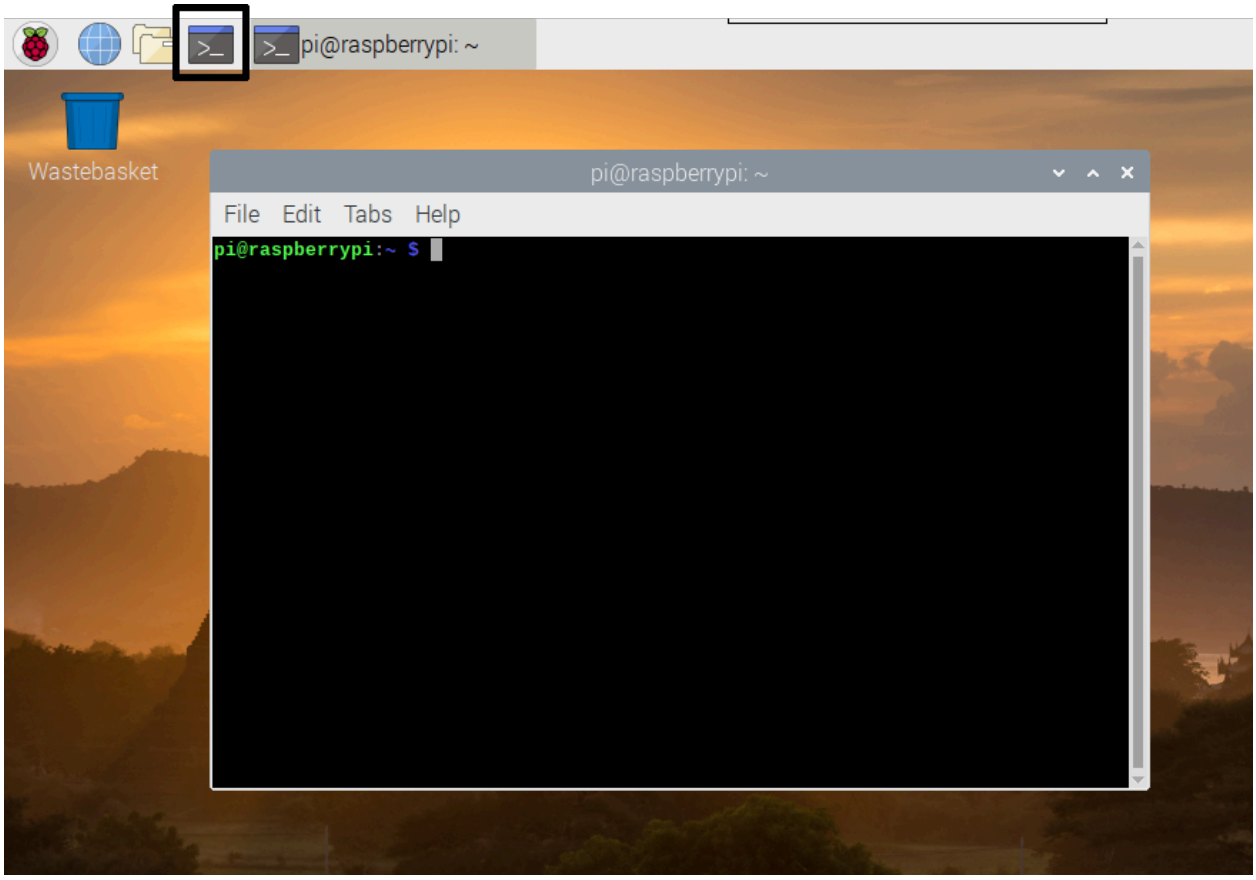

Рисунок В.5 – Зображення консолі у програмному середовищі Raspberry Pi

Рисунок В.6 – Приклад результату роботи програми

Додаток Г (довідниковий)

Інструкція користувача

1. Увімкнути Raspberry Pi;
2. Відкрити консольне меню Raspberry Linux (див. рис. Г1);

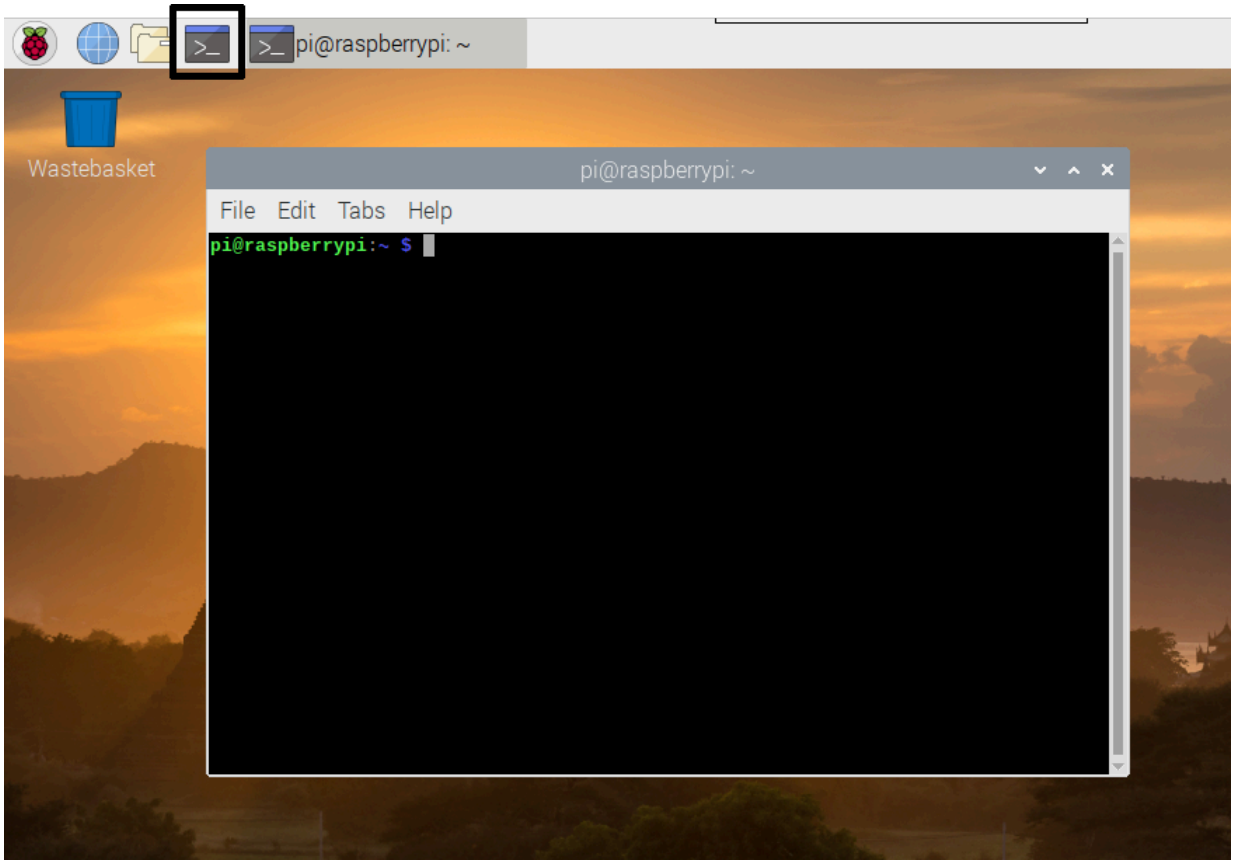

Рисунок Г.1 – Консольне меню Raspberry Linux

3. Завантажуємо програмні компоненти та ініціалізуємо їх для даного програмного середовища. Для цього послідовно вводимо наступні команди: “apt-get install screen”, “screen bash”, “pip init index” (див. рис. Г.2).

```
root@raspberrypi:~# apt-get install screen
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  screen
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 616 kB of archives.
After this operation, 1,024 kB of additional disk space will be used.
Get:1 http://ftp.uk.debian.org/debian/ squeeze/main screen armel 4.0.3-1
6 kB]
Fetched 616 kB in 0s (1,194 kB/s)
Selecting previously deselected package screen.
(Reading database ... 49685 files and directories currently installed.)
Unpacking screen (from .../screen_4.0.3-14+b1_armel.deb) ...
Processing triggers for man-db ...
Processing triggers for install-info ...
Setting up screen (4.0.3-14+b1) ...
root@raspberrypi:~# screen bash
```

Рисунок Г.2 – Завантаження та ініціалізація програмних компонентів

4. Підготувати архів зображень або окремий відеопотік. Для створення архіву зображень потрібно вибрати випадкові зображення котів, та за допомогою спеціальної утиліти SONY Vegas Pro, сформувати з даних зображень архівований відеопотік (див. рис. Г.3).

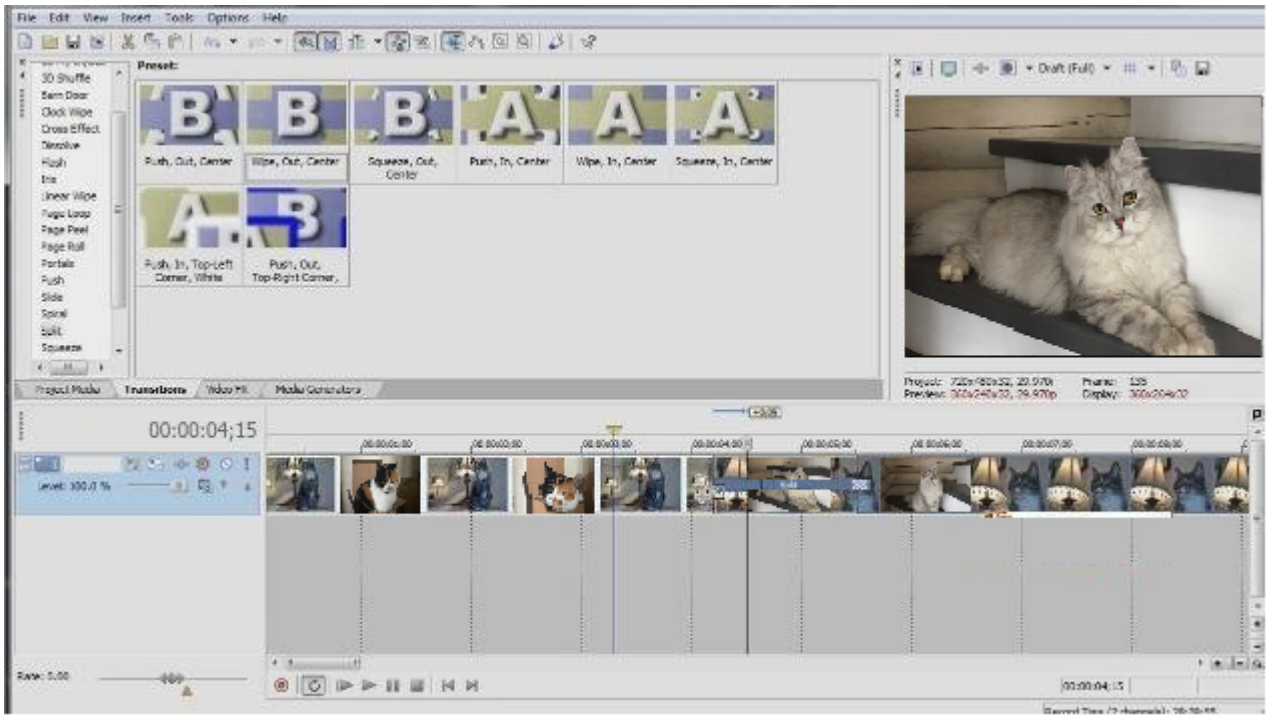

Рисунок Г.3 – Формування архіву відеопотоку

5. Підготовленому архіву зображень змінити назву на «video stream» та запустити програму (див. рис. Г.4 та рис. Г.5).

Рисунок Г.4 – Результат роботи програми

Рисунок Г.5 – Результат роботи програми