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 нейромережевого розпізнавання зображень тварин»

Виконав: студент 2-го курсу, групи 2КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Максименко В. О.
(прізвище та ініціали)

Керівник: к.т.н., проф каф. КН
Месюра В.І.
(прізвище та ініціали)
«15» 12 2022 р.

Опонент: к.т.н., професор каф. КСУ
Биков М. М.
(прізвище та ініціали)
«15» 12 2022 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«16» 12 2022 р.

Вінниця ВНТУ - 2022 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.
15.09. **2022 року**

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Максименку Владиславу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія нейромережевого розпізнавання зображень тварин

керівник роботи к.т.н., проф. кафедри КН Месюра В. І.
затверджені наказом вищого навчального закладу від "11" 09 2022 року №203
2. Строк подання студентом роботи 18 листопада 2022 року
3. Вихідні дані до роботи:
Вхідні дані – формат вхідних зображень - jpg, кількість згорткових шарів – не менше 3 , розмір зображення на вході нейромережі – 50х50, обсяг навчальної вибірки – не менше 500, обсяг тестової вибірки – не менше 100, середовище програмування – об'єктно-орієнтоване.

4. Зміст текстової частини:
Вступ, аналіз предметної області розпізнавання зображень, розробка інформаційної технології розпізнавання зображень тварин, програмна реалізація інформаційної технології розпізнавання зображень тварин, тестування та аналіз результатів роботи програми розпізнавання зображень тварин, економічна частина, висновки, список використаних джерел, додатки

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
Алгоритм роботи програми розпізнавання зображень тварин, структура нейронної мережі, структура інформаційної технології нейромережевого розпізнавання зображень тварин, структура програми розпізнавання зображень тварин, робочі вікна програми розпізнавання зображень тварин, результати роботи програми розпізнавання зображень тварин.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконав прийняв
1-4	Месюра В.І., к.т.н., проф. каф. КН	14.09.22	14.09.22
5	Бурєннікова Н. В., д. е. н., проф. каф. ЕПВМ	19.09.22.	19.09.22

7. Дата видачі завдання 14.09. 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз сучасного рівня інформаційних технологій розпізнавання зображень тварин. Постановка задач дослідження	14.09.22 р. – 01.10.22 р.
2	Побудова моделей розпізнавання зображень тварин на основі нейронної мережі та функціонування нейронної мережі	02.10.22 р. – 16.10.22 р.
3	Практичне застосування та оцінка ефективності розроблених моделей	17.10.22 р. – 07.11.22 р.
4	Підготовка економічної частини	08.11.22 р. – 21.11.22 р.
5	Апробація та/або впровадження результатів дослідження	23.11.22 р. – 01.12.22 р.
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	02.12.22 р. – 14.12.22 р.

Студент _____ (підпис) Максименко В. О.

Керівник роботи _____ (підпис) Месюра В.І.

АНОТАЦІЯ

УДК 004.8

Максименко В. О. Інформаційна технологія нейромережевого розпізнавання зображень тварин. Магістерська кваліфікаційна робота зі спеціальності 122 – комп'ютерні науки, освітня програма - комп'ютерні науки. Вінниця: ВНТУ, 2022. 101 с.

На укр. мові. Бібліогр.: 21 назв; рис.: 21; табл. 14.

Дана магістерська кваліфікаційна робота присвячена розробці інформаційної технології та програмного забезпечення для нейромережевого розпізнавання зображень тварин. Було обґрунтовано вибір згорткової нейромережі. Була розроблена структура та проаналізована математична модель згорткової нейромережі, розроблено структуру інформаційної технології нейромережевого розпізнавання зображень тварин. Програмна реалізація здійснена у середовищі Visual Studio Code на мові C# з використанням бібліотеки ConvNetSharp. Тестування нейронної мережі проходило на вибірці з 100 зображень, з яких 50 – були зображення котів, а інші 50 – зображеннями інших тварин. Розроблена згорткова нейромережа ConvNetSharp Network має достовірність розпізнавання 94%, а глибинна мережа переконань Accord.Net Network (аналог) має достовірність розпізнавання 83%, тобто мета роботи досягнута – достовірність розпізнавання підвищена.

Графічна частина складається з 6 плакатів.

У економічному розділі розраховано рівень комерційного потенціалу розробки, який становить 42,0 бала, спрогнозовано орієнтовану величину витрат по кожній з статей витрат, термін окупності витрат для виробника становить 0,60 роки.

Ключові слова: розпізнавання, математична формула, нейронна мережа, багатошаровий персептрон, зворотнє поширення помилки.

ABSTRACT

Maksymenko V. O. Information technology of mathematical formulas recognition on the basis of a neural network. Master's thesis in the specialty 122 - computer sciences, educational program - computer science. Vinnytsia: VNTU, 2022. 101 p.

In Ukrainian language. Bibliographer: 21 titles; fig .: 21; table 14.

This Master's thesis is devoted to the development of information technology and software for neural network recognition of animal images. The choice of a convolutional neural network was justified. The structure and analysis of the mathematical model of the convolutional neural network was developed, and the structure of the information technology of neural network recognition of animal images was developed. The software implementation was carried out in the Visual Studio Code environment in the C# language using the ConvNetSharp library. The neural network was tested on a sample of 100 images, of which 50 were images of cats, and the other 50 were images of other animals. The developed convolutional neural network ConvNetSharp Network has a recognition accuracy of 94%, and the deep belief network Accord.Net Network (similar) has an accuracy of recognition of 83%, that is, the goal of the work has been achieved - the recognition accuracy is increased..

The graphic part consists of 6 posters.

In the economic section, the level of the commercial potential of the development is calculated, which is 42.0 points, the estimated amount of costs for each of the cost items is predicted, the payback period for the manufacturer is 0.60 years.

Keywords: recognition, mathematical formula, neural network, multilayer perceptron, inverse error propagation

ЗМІСТ

ВСТУП 6
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ 10
1.1 Постановка задачі розпізнавання зображень тварин 10
1.2 Огляд відомих методів розпізнавання зображень 10
1.3 Аналіз програмних засобів для нейромережевого розпізнавання 13
1.3.1 Neuroph 15
1.3.2 Encog 16
1.4 Висновок до розділу 1 17
2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН НА ОСНОВІ CONVNETSHARP NETWORK 18
2.1 Обґрунтування вибору згорткової нейронної мережі для розпізнавання зображень тварин 18
2.1.1 Персептрон 18
2.1.2 Багатошаровий персептрон 20
2.1.3 Згорткові нейронні мереж 20
2.1.4 Рекурентна нейронна мережа 24
2.2 Архітектура згорткової нейронної мережі 25
2.3 Математична модель згорткової нейронної мережі 27
2.4 Навчання згорткової нейронної мережі 29
2.5 Розробка структури згорткової нейромережі 33
2.6 Структура інформаційної технології нейромережевого розпізнавання зображень тварин 35
2.7 Висновок до розділу 2 37
3 ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН НА ОСНОВІ CONVNETSHARP 38
3.1 Обґрунтування вибору середовища програмування 38
3.2 Обґрунтування вибору мови програмування 39
3.3 Використання бібліотеки ConvNetSharp 40
3.4 Структура та алгоритм роботи програмного забезпечення 41

3.5 Реалізація згорткової нейронної мережі для розпізнавання зображень за допомогою ConvNetSharp 43
3.6 Висновок до розділу 3 46
4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН НА ОСНОВІ CONVNETSHARP .. 47
4.1 Тестування програми розпізнавання зображень тварин 47
4.2 Аналіз результатів роботи програми розпізнавання зображень тварин ... 52
4.3 Висновок до розділу 4 53
5 ЕКОНОМІЧНА ЧАСТИНА 54
5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки 54
5.2 Розрахунок узагальненого коефіцієнта якості розробки 58
5.3 Розрахунок витрат на проведення науково-дослідної роботи 60
5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором 71
5.5 Висновок до розділу 5 76
ВИСНОВКИ 77
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 79
Додаток А (обов’язковий) Результат перевірки на плагіат в онлайн-системі UNICHECK 83
Додаток Б (обов’язковий) Лістинг програми 84
Додаток В (обов’язковий) ІЛЮСТРАТИВНА ЧАСТИНА 91
Додаток Г (довідниковий) Інструкція користувача 98

ВСТУП

Актуальність. Дана робота призначена розпізнаванню образів тварин. На сьогоднішній день створення штучних систем розпізнавання образів залишається досить складною теоретичною та технічною проблемою. Розпізнавання образів широко використовується в самих різних областях – від військової справи й систем безпеки до оцифрування різних аналогових сигналів. Розпізнавання образів вирішує задачу виділення істотних ознак та їх віднесення до певного класу, що характеризують даний образ, із загальної маси даних.

Існують два класи задач розпізнавання / класифікації:

- розпізнавання з учителем;
- розпізнавання без вчителя (наприклад, кластеризація).

Потрібно зауважити, що завдання розпізнавання вважається завданням класифікації, в якій класи або задаються дизайнером системи (в розпізнаванні з учителем), або будуються на підставі схожості образів (в розпізнаванні без вчителя).

Сучасним підходом до розпізнавання зображень є згорткові нейронні мережі. Вони мають велику кількість шарів, в порівнянні з класичним багатошаровим персептроном. За рахунок спільних ваг, які використовуються відразу декількома нейронами в кожному шарі, вдається знизити загальну кількість параметрів мережі і прискорити навчання. Також, на відміну від багатошарового персептрона, згорткові мережі сприйнятливі до топології вхідного зображення.

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 зображень тварин шляхом використання згорткової нейронної мережі.

Для досягнення мети розробки необхідно виконати такі задачі:

- провести аналіз предметної області розпізнавання зображень тварин;
- розглянути існуючі методи розпізнавання зображень тварин та обрати й обґрунтувати вибір методу, який задовольняє мету даної магістерської кваліфікаційної роботи;
- проаналізувати архітектуру та математичну модель згорткової нейронної мережі для розпізнавання зображень тварин;
- сформулювати стадії інформаційної технології розпізнавання зображень тварин, розробити структуру та алгоритм роботи програмного засобу розпізнавання зображень тварин;
- виконати програмну реалізацію запропонованої інформаційної технології розпізнавання зображень тварин;
- провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об’єкт дослідження – процес розпізнавання зображень тварин із застосуванням інтелектуальних технологій.

Предмет дослідження – інформаційна технологія та програмні засоби розпізнавання зображень тварин комп’ютерними засобами з використанням нейронних мереж та достовірність їх роботи.

Методи дослідження. У роботі використані наступні методи наукових досліджень: системного аналізу, теорії штучних нейронних мереж для реалізації інформаційної технології, методи математичної статистики для розробки процесу розв'язання задачі нейромережевого розпізнавання зображень

тварин та обрахунків результатів експериментів із програмним засобом, об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів.

1. Набула подальшого розвитку інформаційна технологія нейромережевого розпізнавання зображень тварин, яка відрізняється використанням згорткової нейронної мережі ConvNetSharp Network, що дозволило підвищити достовірність програмних засобів розпізнавання зображень тварин.

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

Запропонована інформаційна технологія сприяє підвищенню достовірності програмних засобів розпізнавання зображень тварин, зокрема:

- розроблено алгоритм роботи програмного забезпечення розпізнавання зображень тварин;
- розроблено програмні засоби для розпізнавання зображень тварин.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю використання математичного апарату методів дослідження, експериментальними дослідженнями тестування програмної реалізації інформаційної технології розпізнавання зображень тварин. 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 зображень тварин та методів підвищення достовірності програмних засобів розпізнавання зображень тварин [1].

Апробація результатів роботи. Результати роботи були апробовані на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)», Вінниця, 15 листопада 2022 року - 12 травня 2023 року.

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

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

1.1 Постановка задачі розпізнавання зображень тварин

Задача якісного розпізнавання об’єктів [2] різної природи є однією із найрозповсюдженіших у сучасні сфері інформаційних технологій. Аналіз та розпізнавання образів є доволі актуальнимми у багатьох галузях. Наприклад, розпізнавання зображень застосовується у промисловості для контролю якості, у медицині – для діагностування, в біологічних та інших наукових дослідженнях, для охорони об’єктів, у біометриці та інших галузях.

Програма, що розробляється, має розпізнавати на вхідному зображення об’єкти у вигляді тварин (у даному випадку котів). Але тварини взяті лише для прикладу, тому що, в принципі, розроблювана програма повинна передбачати можливість навчання на будь-який вид об’єктів розпізнавання (автомобіль, пішохід, велосипед і т.д. і т.п.). Повинна бути передбачена можливість вибору аналізованого зображення із певної папки. Після аналізу зображення за допомогою нейронної мережі на нього має накладатись текстовий напис з назвою класу розпізнаного об’єкта (у нашому випадку «Cat» і «No Cat»).

Ця робота передбачає реалізацію цього завдання за допомогою двох різних типів нейромереж і подальше порівняння їх параметрів (зокрема достовірності розпізнавання та часу навчання) з метою визначення переваг і недоліків кожного із двох типів нейромереж.

1.2 Огляд відомих методів розпізнавання зображень

Є чимало методів для розпізнавання зображень. Основна відмінність між ними полягає в наборі характеристик, що розпізнаються. Передбачено поділ на чотири загальних підходи розпізнавання образів, а саме:

- співставлення шаблонів,

- статистичні методи,

- структурні методи,

- нейронні мережі.

Такі підходи не обов’язково є незалежними чи відокремленими один від одного. Іноді метод в одному підході також можна віднести і до інших.

Операції співставлення шаблонів визначають ступінь подібності між двома векторами (групами пікселів, фігур, вигинів і т.д.) у просторі ознак. Відповідні методи можуть бути згруповані в три класи:

- пряме співставлення,

- пружне співставлення,

- релаксаційне співставлення.

Статистичні методи стосуються статистичних функцій прийняття рішень і набір оптимальних критеріїв, що визначають ймовірність належності картини, що спостерігається, до певного класу. Кілька популярних підходів розпізнавання символів належать до цієї групи:

- Правило k-найближчих сусідів (k-NN) є популярним непараметричним методом розпізнавання, де апостеріорна ймовірність приблизно визначається з частоти найближчих сусідів невідомого шаблону. Такий підхід дає досить хороші результати, хоча слід зазначити, що він вимагає значних обчислювальних потужностей в процесі класифікації.

- Прихована модель Маркова (ПММ) є ще одним популярним способом вирішення задачі. Модель є двічі стохастичним процесом: вона включає неспостережуваний процес (звідси "прихована"), що, однак, може спостерігатися через інший випадковий процес, який продукує послідовність спостережень. ПММ широко визнана одним з найпотужніших інструментів для моделювання мови та широкого кола інших реальних сигналів. Ці ймовірнісні моделі мають багато властивостей, бажаних для моделювання символів або слів. Однією з найбільш важливих властивостей є наявність ефективних

алгоритмів для автоматичного навчання моделі без необхідності маркування попередньо сегментованих даних. ПММ широко застосовуються для розпізнавання рукописних слів в тому числі в поєднанні з іншими підходами, такими як стохастичні граматики і нейронні мережі. Є два основних підходи, які можна виділити в зазначених роботах: модель-дискримінантні ПММ і шлях-дискримінантні ПММ. У першому випадку модель будується для кожного класу (слова, літери, або одиниці сегментації) у фазі підготовки, в другому одна ПММ будується для всієї мови або контексту. Продуктивність обох в основному порівняна.

- Метод опорних векторів (Support Vector Machine – SVM) заснований на статистичній теорії навчання і квадратичній оптимізації. SVM – це, по суті, двійковий класифікатор, і кілька SVM можуть бути об'єднані, щоб сформувати систему для класифікації декількох класів. В останні роки SVM отримав більшу вагу в середовищі машинного навчання через його відмінну продуктивність щодо узагальнення..

Існують також декілька інших статистичних підходів, такі як байєсівский або поліноміальний дискримінантний класифікатори, та вони менш популярні для задачі, що розглядається, і в більшості випадків дають гірші результати.

У структурних методах зображення представлені у вигляді об’єднань структурних зображень-примітивів, піддаються кількісній оцінці, і можна знайти взаємозв'язок між ними. Структурні методи можна розділити на два класи: граматичні та графічні методи [2].

Найбільш універсальний підхід до задачі – нейромережевий. Нейронна мережа – це обчислювальна структура, яка складається з штучних нейронів – абстракції нервових клітин людини. Створені при спробах імітувати людський мозок, ці структури широко використовуються в розпізнаванні образів, обробці даних та задачах апроксимації функцій. Основні переваги нейронних мереж полягають в здатності навчатися автоматично на основі вибірок, бути продуктивною на зашумлених даних, можливістю паралельної реалізації та

бути ефективними інструментами для обробки великих баз даних. Нейронні мережі широко використовуються в області, що розглядається, і були досягнуті перспективні результати, особливо в розпізнаванні рукописних символів. У цьому підході є багато різних методів. Найбільш успішними можна назвати нечіткі нейронні мережі, мережу Хеммінга, мережу Хопфілда, самоорганізувальні карти Кохонена та ін [3,4].

1.3 Аналіз програмних засобів для нейромережевого розпізнавання

На сьогоднішній день існує багато програмних засобів розпізнавання зображень, які містять модулі для розпізнавання об’єктів та знаходження схожості на зображеннях та у потоці відео-даних. Нижче наведено деякі відомості про сучасні програмні засоби розпізнавання зображень , а також обробки зображень і відео.

CCV(Community Core Vision) — бібліотека для розпізнавання образів та об’єктів, розроблена і підтримується групою під назвою NUI.

CCV - крос-платформенне рішення з відкритим вихідним кодом для відстеження об'єктів за допомогою так званого комп'ютерного зору. Він приймає відео-дані на вхідний потік і виводить результати спостереження за ним (наприклад, координати і розмір об’єкту, що шуканий) та події (наприклад, переміщення пальцю вниз, переміщення та відпускання), які використовуються у створенні мульти-тач додатків. CCV може взаємодіяти з різними веб-камерами та іншими відео-пристроями, а також підключатися до різних програм з підтримкою TUIO/OSC/XML і підтримує безліч методів мульти-тач освітлення в тому числі: FTIR, DI, DSI та LLP із запланованим розширенням для майбутніх додатків (призначені для користувача модулі та фільтри). Цей проект був розроблений і підтримується групою під назвою NUI.

BoofCV — Java-бібліотека з відкритим вихідним кодом для розпізнавання об’єктів у реальному часі. Написана з нуля, простота для використання і має

високу продуктивність. Її функціональність охоплює широке коло завдань, в тому числі, оптимізованих процедур обробки зображень низького рівня, калібрування камери, функція виявлення / стеження, розпізнавання руху і символів. BoofCV була випущена під ліцензією Apache 2.0 як для академічного та комерційного використання.

BoofCV організована в кілька пакетів: обробка зображень, характеристик, геометричного бачення, калібрування, розпізнавання та візуалізації. Блок обробки зображень містить часто використовувані функції обробки зображень, які працюють безпосередньо з пікселями. Блок ознак містить алгоритми вилучення ознак для використання в операціях більш високого рівня.

Блок калібрування містить підпрограми для визначення внутрішніх і зовнішніх параметрів камери. Блок розпізнавання для розпізнавання і відстеження складних візуальних об'єктів. Блок геометричного бачення складається з підпрограм для обробки визначених ознак зображення з використанням 2D і 3D геометрії. Блок візуалізації містить підпрограми для візуалізації і відображення виділених ознак.

Computer Vision Sandbox — призначена, як пакет програмного забезпечення, метою якого є вирішення різних завдань, пов'язаних з комп'ютерним баченням в таких областях, як, наприклад, відеоспостереження, автоматизації на основі розбору різних видів зображень, обробки відео і тому подібного. На даний момент програмне забезпечення знаходиться на ранній початковій фазі з великою кількістю задач, які ще потрібно реалізувати, але досі не реалізовані. Однак розробники рухаються по плану, розвивають бібліотеку, в результаті чого з’явиться ще багато нових можливостей, будуть досягнуті цілі, що були поставлені на початку розробки.

Отже, у світі є багато систем розпізнавання зображень, які мають як свої переваги, так і недоліки. Найбільшим недоліком усіх систем є точність розпізнавання. Тому у даній роботі буде створюватися система, яка орієнтуватиметься на збільшення достовірності розпізнавання зображень.

На даний момент існує велика кількість алгоритмів розпізнавання та програмних засобів для побудови нейромережевих (і не тільки) розпізнавачів. Розглянемо деякі з них.

1.3.1 Neuroph

Neuroph - система для нейромережевого розпізнавання, написана на мові Java (рис. 1.1). Складається з Java API, що включає в себе основні класи, класи-утиліти і реалізацію конкретних типів нейронних мереж. Також включає в себе середовище Neuroph Studio, реалізовану на платформі NetBeans.

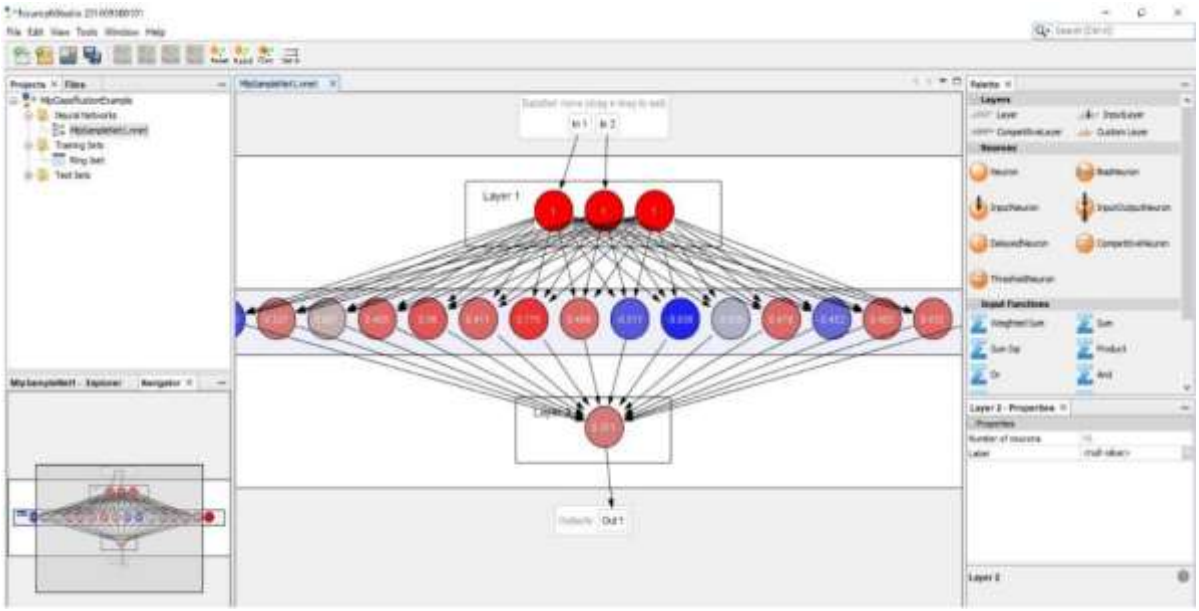


Рисунок 1.1 - Інтерфейс програми Neuroph

У середовищі реалізовані наступні нейромережеві архітектури:

- Адалайн
- Персептрон
- Багатошаровий персептрон з алгоритмом зворотного поширення помилки, моментом
- мережа Хопфілда

- Двостороння асоціативна пам'ять
- мережа Кохонена
- мережа Хебба
- RBF нейронна мережа

У систему включені приклади використання мереж для прогнозування цін на фінансових ринках, приклади класифікації тварин і інші. У систему включена підтримка розпізнавання зображень за допомогою багатошарового перцептрона з алгоритмом зворотного поширення помилки. У системі немає реалізації згорткових нейронних мереж.

1.3.2 Encog

Система Encog є найбільш повною з розглянутих (рис.1.2). Вона включає в себе велику кількість ефективно написаних Java класів. На відміну від інших систем, в ній реалізовані не тільки нейромережеві алгоритми розпізнавання, але і кластеризація, генетичні алгоритми, приховані марковскі моделі, байєсовські мережі та ін.

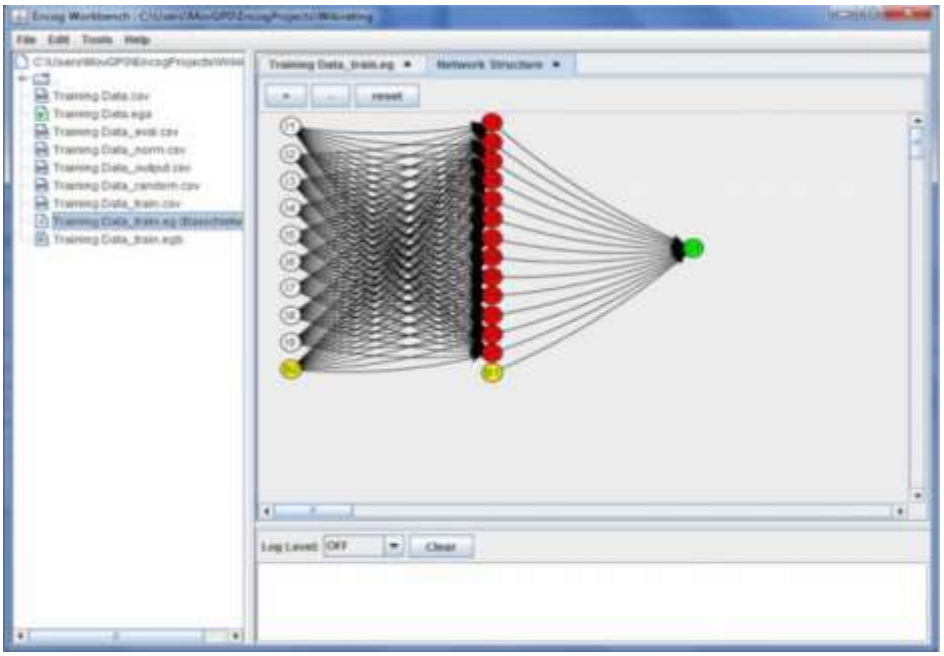


Рисунок 1.2 - Інтерфейс системи Encog

Серед підтримуваних архітектур нейронних мереж є такі, як адалайн, машина Больцмана, нейронні мережі зворотного поширення, рекурентні мережі Ельмана, рекурентні мережі Джордана, самоорганізовані карти Кохонена і т.д. Доступні такі функції активації нейронів: біполярна, змагальна, функція Еліотта, функція Гаусса, гіперболічний тангенс, лінійна, синусоїдальна, сигмоїда.

У порівнянні з іншими система, Encog написана дуже ефективно, підтримується паралельне функціонування мереж на машинах з декількома процесорами. Але Encog так само не підтримує розпізнавання зображень - немає реалізації згорткових нейронних мереж.

Розглянувши велику кількість систем для розпізнавання образів, був зроблений висновок, що всі вони не підходять для реалізації поставлених завдань, тому було вирішено створити власну систему.

1.4 Висновок до розділу 1

У першому розділі зроблено огляд методів розпізнавання зображень на сьогоднішній день. Проведено також огляд відомих програмних засобів для роботи з інтелектуальними обчисленнями у сфері комп’ютерного зору. В ході аналізу предметної області розглянуто основні методи розпізнавання зображень та як найбільш перспективний, було обрано нейромережевий метод.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН НА ОСНОВІ CONVNETSHARP NETWORK

2.1 Обгрунтування вибору згорткової нейронної мережі для розпізнавання зображень тварин

Штучна нейронна мережа (ШНМ) - це взаємозв'язана мережа вузлів, уподібнена до великої мережі нейронів у головному мозку [5].

Штучні нейронні мережі імітують поведінку мозку у простішому вигляді. Вони можуть бути навчені контрольованим та неконтрольованим шляхами. У контрольованій ШНМ, мережа навчається шляхом передавання відповідної вхідної інформації та прикладів вихідної інформації. Такий вид навчання додає ваги зв'язкам ШНМ, але це буде обговорено пізніше. Неконтрольоване навчання у ШНМ намагається "змусити" ШНМ "зрозуміти" структуру переданої вхідної інформації "самостійно".

Біологічний нейрон імітується у ШНМ через активаційну функцію. У задачах класифікації (наприклад класифікація зображень) активаційна функція повинна мати характеристику "вмикача". Іншими словами, якщо вхід більше, ніж деяке значення, то вихід повинен змінювати стан, наприклад з 0 на 1 або з -1 на 1. Це імітує "включення" біологічного нейрону [4,5].

За весь час існування ШНМ було створено досить багато видів штучних нейронних мереж, кожна з яких має певні, свої, особливості. Розглянемо ті з них, що більше всього підходять для вирішення поставленої задачі.

2.1.1 Персептрон

Розенблатт сконструював перший у 1958 р. нейронний комп'ютер, персептрон, намагаючись імітувати навчання людини. Дендрити нейрона моделюються за вагами, що множать на вхідні значення. Крім того, додається

значення зміщення для моделювання необхідного потенціалу активації нейрона. Потім множинні значення та зміщення підсумовуються в тілі клітини і пропускаються через функцію активації для отримання виходу, що представляє швидкість випалу в нейроні. Описана архітектура показана на малюнку нижче.

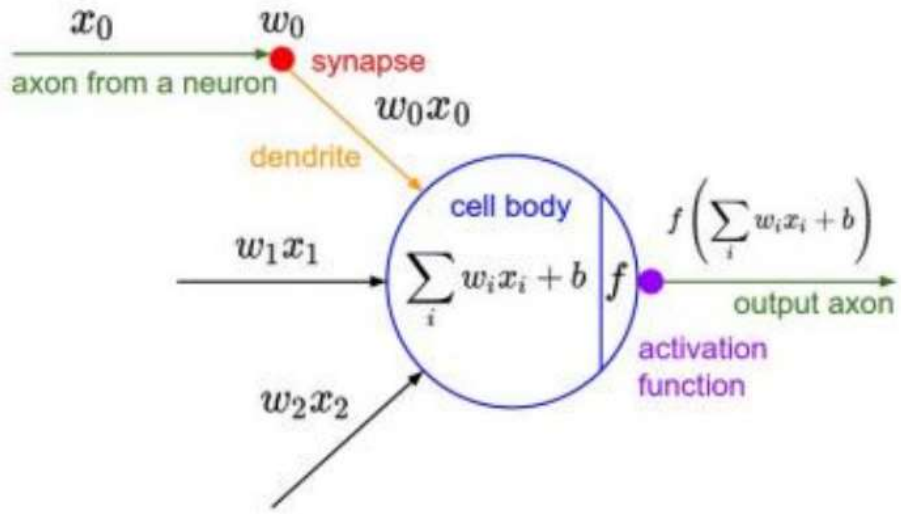


Рисунок 2.1 - Модель персептрона з вагами w_i, входами x_i та зсувом b

Персептрон - це лінійний класифікатор, визначений вагами w_i, зміщенням b і функцією активації f. Можна об'єднати зміщення з вагами за допомогою однорідних координат, тобто покласти зміщення в нижній частині вагового вектора і додати постійну 1 до вхідного вектора x. Можуть використовуватися різні функції активації, але оригінальний персептрон використовує функцію «жорсткої» сходинки, що призводить до двійкового виходу. Тому персептрон є лінійним класифікатором, і його ваги визначають гіперплощину як лінійну межу рішення між класами. Таким чином, його представницька сила обмежена; неможливо моделювати, наприклад, функцію XOR з використанням персептрона, оскільки жодна лінія не може бути проведена для розділення класів [6].

2.1.2 Багатошаровий персептрон

Багатошаровий персептрон має три або більше шарів. Він використовується для класифікації даних, які неможливо розділити лінійно. Це тип штучної нейронної мережі, яка повністю пов'язана. Це відбувається тому, що кожен окремий вузол у шарі з'єднаний з кожним вузлом у наступному шарі.

Багатошаровий персептрон використовує нелінійну функцію активації (головним чином гіперболічний тангенс або логістичну функцію). Ось як виглядає багатошаровий персептрон (рис. 2.2).

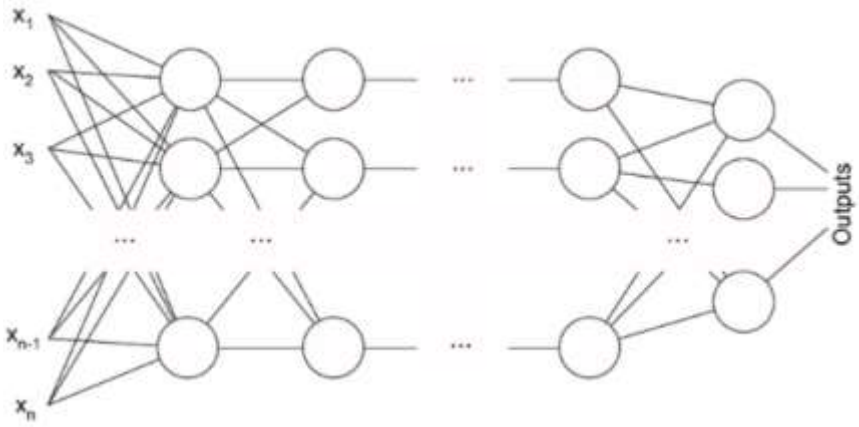


Рисунок 2.2 – Багатошаровий персептрон

Цей тип нейронної мережі широко застосовується в технологіях розпізнавання мовлення та технологіях машинного перекладу [6].

2.1.3 Згорткові нейронні мереж

Згорткові нейронні мережі (CNN) - це підгрупа алгоритмів глибокого навчання, які зосереджуються на завданнях комп'ютерного зору та обробки зображень. Системні мережі CNN надихнуті тим, як мозок обробляє візуальну інформацію. Hubel і Wiesel були першими, хто запропонував глибокі моделі, схожі на структуру зорової котячої кори. Ці моделі ототожнювали прості

клітини з локальними сприйнятливими полями, схожими на фільтри або ядра, і складні комірки, схожі на об'єднання шарів. Перший CNN був введений в неокогнітроні Фукушіми. Пізніше CNN були покращені. У LeNet-5 Yann LeCun застосував стохастичні градієнти на основі градієнта для розпізнавання документів і був дуже успішним для розпізнавання рукописних завдань. Основним недоліком у дослідженні та розробці CNN в 1990-х і 2000- х роках була необхідна обчислювальна потужність для їх широкого застосування до зображень з високою роздільною здатністю. Однак це змінилося з 2010 році. Існує три причини, завдяки чому глибокі мережі стали успішними:

– більша обчислювальна здатність завдяки закону Мура, особливо це стосується сьогоднішніх графічних процесорів;

– більше навчальних даних;

– нові та кращі алгоритми.

Зі збільшенням обчислювальної потужності дослідники описали нові способи більш ефективного тренування згорткових нейронних мереж, що дозволяло створювати більш глибокі мережі.

Останнім часом продуктивність значно збільшилась завдяки безлічі баз даних зображень, наближаючись або навіть перемагаючи продуктивність людини, наприклад щодо розпізнавання цифр (<0.25 відсотка) та багатьох інших завдань з розпізнавання шаблонів, найбільш важливих проблем візуальної класифікації та інших.

Загалом, CNN дуже добре класифікують такі об'єкти, як конкретні породи собак та котів на основі дрібнозернистих деталей, тоді як люди мають проблеми з цим. Недоліком глибокого навчання є те, що для розпізнавання об'єктів у реалістичних умовах потрібні дуже великі набори даних для навчання. В даний час найкращі CNN борються з невеликими або плоскими об'єктами, або спотвореними цифровими фільтрами. Недавнє дослідження виявило відмінності того, як глибокі нейронні мережі та людина розпізнають об'єкти. У цьому дослідженні зображення кодуються таким чином, який

людина не в змозі розпізнати, але глибокі нейронні мережі виявляли правильний клас об'єкта майже зі 100% точністю.

Для тренування дуже складних згорткових нейромереж на великих наборах даних, останнім часом, з’явилася тенденція до збільшення кількості шарів та розмірів шарів при використанні відсіювання для вирішення проблеми надмірного оснащення. Krizhevsky та Szegedy, творці двох найбільш відомих мереж, що використовуються в дослідженнях, а саме AlexNet та GoogLeNet, підкреслюють, наскільки важливо використовувати високий коефіцієнт випадання під час тренувань.

Ще одна тенденція - зробити мережі дуже глибокими, тобто мережі мають багато шарів. Zeiler експериментував із глибиною CNN і дійшов висновку, що продуктивність мережі сильно залежить від кількості шарів. Виміряна продуктивність значно знижується, навіть якщо видаляється один згортковий шар. Він зазначає, що глибина мережі важливіша, ніж будь-яка інша архітектурна складова мережі.

Отже, вибір базової архітектури є ключем до ефективності мережі. Це також можна побачити на GoogLeNet, який був опублікований у 2014 році та має глибину 22 шари, не рахуючи шарів об’єднання. У 2012 році AlexNet розпочав ажіотаж навколо прогресу в глибокому навчанні, зокрема, з мережею, що складається з 8 шарів. Хоча вона досягла найсучасніших результатів, коли вона була опублікована, відтепер багато інших мереж перевершили її результативність.

Обмежуючим фактором розміру CNN є об'єм пам'яті, наявної в поточних системах обробки графічних процесорів (GPU), і переносний час на навчання. До того як розпочали широко використовувати GPU для навчання глибоких мереж, загальний час навчання процесів був більшим у 9 разів. Бібліотека cuDNN від Nvidia, яка оптимізована для швидкої обробки зображень та ефективних операцій згортання в графічних процесорах, додатково прискорює час навчання майже в 9 разів. Таким чином, сучасні графічні процесори з

найновішими встановленими бібліотеками CUDA та cuDNN прискорюють час обчислення в 17 разів порівняно з сьогоднішніми процесорами. Залежно від кількості параметрів мережі та розміру набору даних, навчання згорткової нейронної мережі на графічному процесорі з випадково ініційованими вагами може іноді займати кілька днів або навіть і тижнів.

Згорткова нейронна мережа (CNN) використовує варіацію багатошарових перцептронів. CNN містить один або декілька згорткових шарів. Ці шари можуть бути повністю пов'язані між собою або об'єднані.

Перш ніж передати результат наступному шару, згортковий шар використовує згорткову операцію на вході.

Завдяки цій згортковій роботі мережа може бути набагато глибшою, але зі значно меншими параметрами. Завдяки цій здатності конвертовані нейронні мережі показують дуже ефективні результати в розпізнаванні зображень і відео, природній обробці мови та системах рекомендування.

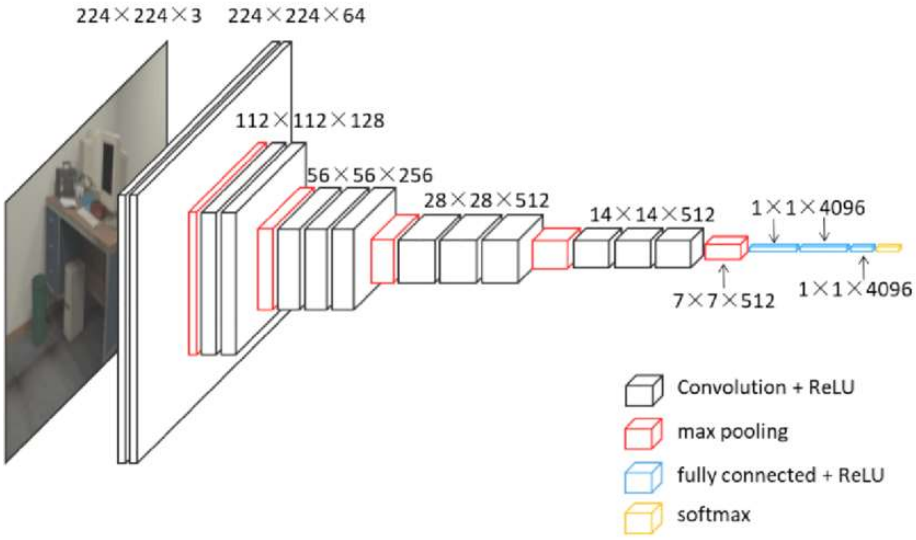


Рисунок 2.3 – Архітектура згорткової нейронної мережі

Згорткові нейронні мережі показують чудові результати в семантичному розборі та виявленні парафрази. Вони також застосовуються при обробці сигналів та класифікації зображень, де відмінно себе проявляють [7].

2.1.4 Рекурентна нейронна мережа

Рекурентна нейронна мережа - це тип штучної нейронної мережі, в якій вихід певного шару зберігається і подається назад на вхід. Це допомагає передбачити результат шару.

Перший шар формується так само, як і в мережі подачі (див. рис. 2.4). Тобто з добутком суми ваг та ознак. Однак у наступних шарах починається повторюваний процес нейронної мережі.

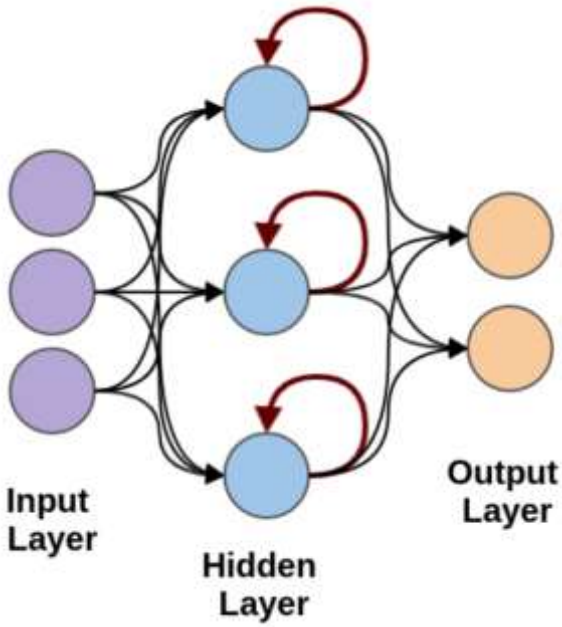


Рисунок 2.4 - Рекурентна нейронна мережа

Від кожного часового кроку до наступного кожен вузол запам'ятовуватиме інформацію, яку він мав у попередньому часовому кроці. Іншими словами, кожен вузол виконує функцію комірки пам'яті під час обчислення та виконання операцій. Нейронна мережа починається з прямого розповсюдження, як зазвичай, але запам'ятовує інформацію, яку, можливо, потрібно буде використовувати згодом.

Якщо прогноз неправильний, система самостійно навчається та працює над правильним прогнозуванням під час розмноження. Цей тип нейронної мережі дуже ефективний в технології перетворення тексту в мовлення.

Для розробки програмного продукту було обрано згорткові нейронні мережі, оскільки вони успішно застосовуються до аналізу візуальних зображень.

2.2 Архітектура згорткової нейронної мережі

Згорткові нейронні мережі (CNN) в останні роки повністю домінували в просторі машинного зору. CNN складається з вхідного шару, вихідного шару, а також безлічі прихованих шарів. Приховані шари CNN зазвичай складаються зі згорткових шарів, шарів об'єднання, повнозв'язних шарів і шарів нормалізації (ReLu). Для складніших моделей можна використовувати додаткові шари. Приклад типової CNN можна побачити на рис.2.5.

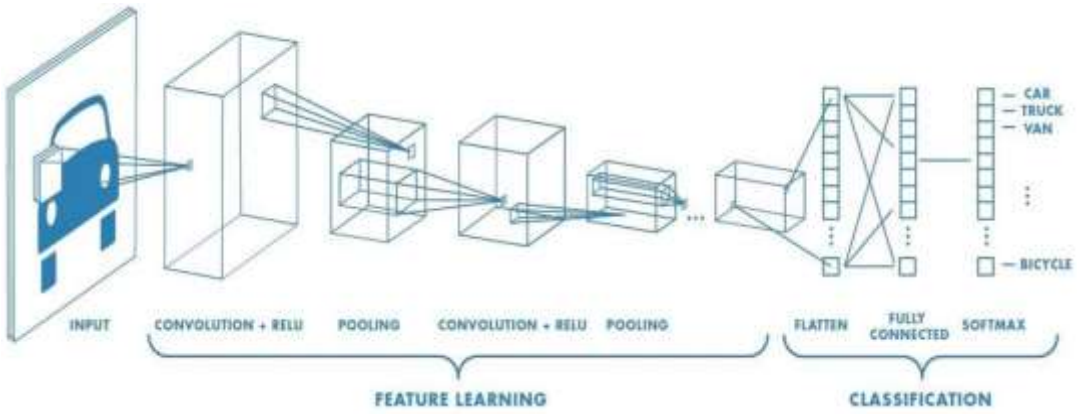


Рисунок 2.5 - Приклад архітектури шарів згорткової нейронної мережі

Архітектура CNN показала відмінні результати у багатьох проблемах із комп’ютерним зором та машинним навчанням. CNN тренує та прогнозує на абстрактному рівні з деталями, залишеними для наступних розділів. Ця модель CNN широко використовується в сучасних програмах машинного навчання завдяки ефективності. Лінійна алгебра є основою для роботи цих CNN.

Множення матриці на вектор лежить в основі представлення даних і ваг. Кожен з шарів містить різний набір характеристик для набору зображень. Наприклад, якщо зображення обличчя є входом до CNN, мережа вивчить деякі основні характеристики, такі як краї, яскраві плями, темні плями, фігури тощо, у своїх початкових шарах. Наступний набір шарів буде складатися з фігур і предметів, пов'язаних із зображенням, які можна розпізнати, такі як: очі, ніс і рот. Наступний шар складається з аспектів, схожих на фактичні обличчя, іншими словами, форми та об'єкти, які мережа може використовувати для визначення людського обличчя. CNN відповідає деталям, а не цілому зображенню, тому розбиває процес класифікації зображення на більш дрібні частини (характеристики). Сітка 3x3 визначена для відображення функції вилучення ознак CNN для оцінки. Наступний процес, відомий як фільтрування, включає в себе підшивку функції патчем зображення. Один за одним кожен піксель множиться на відповідний піксель функції, і після завершення всі значення підсумовуються та діляться на загальну кількість пікселів у просторі функцій. Кінцеве значення для функції потім розміщується в патчі функції. Цей процес повторюється для інших патчів функцій з подальшим випробуванням усіх можливих повторних застосувань цього фільтра, який відомий як згортка.

Наступний шар CNN називається "максимальним об'єднанням", що передбачає зменшення розміру зображення. Щоб об'єднати зображення, потрібно визначити розмір вікна (наприклад, зазвичай 2x2 / 3x3 пікселі), також слід визначити крок (наприклад, 2 пікселі). Потім вікно фільтрується через зображення кроками, при цьому максимальне значення записується для кожного вікна. Максимальне об'єднання зменшує розмірність кожної карти зображень, зберігаючи найважливішу інформацію. Нормалізаційний шар CNN, який також називають процесом випрямленого лінійного блоку (ReLu), включає зміну всіх негативних значень у відфільтрованому зображенні на 0. Цей крок повторюється на всіх відфільтрованих зображеннях, шар ReLu збільшує нелінійні властивості моделі.

Наступним кроком CNN є складання шарів (згортка, об'єднання, ReLu), так що вихід одного шару стає входом наступного. Шари можна повторити, що призведе до «глибокого укладання». Заключний шар в архітектурі CNN називається повнозв'язним шаром, також відомим як класифікатор. У цьому шарі кожне значення отримує голос за визначення класифікації зображення. Повнозв'язні шари часто складаються між собою, при цьому кожен проміжний шар «голосує» за фантомними «прихованими» категоріями. Насправді кожен додатковий шар дозволяє мережі вивчити ще більш складні комбінації функцій для кращого прийняття рішень. Значення, використовувані для шару згортки, а також ваги для повністю з'єднаних шарів отримують за допомогою зворотного розповсюдження, що здійснюється глибокою нейронною мережею. Зворотне розповсюдження полягає в тому, що нейронна мережа використовує помилку в остаточній відповіді, щоб визначити, наскільки мережа коригується та змінюється.

2.3 Математична модель згорткової нейронної мережі

Найчастіше при побудові архітектури CNN складають кілька шарів згортки ректифікації (ReLu), за ними слідують шари пулінгу; цей шаблон продовжує працювати, доки знімок не буде об'єднано до малого розміру. В деякий момент здійснюється перехід до повнозв'язних шарів. Останній повнозв'язний шар містить вихідну інформацію, наприклад, оцінки класу. Іншими словами, найбільш поширена архітектура CNN відповідає схемі:

$$INPUT \rightarrow \left[[CONV \rightarrow RELU] * N \rightarrow POOL?\right] * M \rightarrow [FC \rightarrow RELU] * K \rightarrow FC, \quad (2.1)$$

де «$*$» означає повторення, а «$POOL?$» - вказує на додатковий шар пулінгу. Крім того, $0 \leq N \leq 3$, $M \geq 3$, $0 \leq K < 3$. Далі представлені архітектури CNN, організовані за таким зразком:

$$INPUT \rightarrow FC, \tag{2.2}$$

що реалізують лінійний класифікатор. Тут $N = M = K = 0$.

$$INPUT \rightarrow CONV \rightarrow RELU \rightarrow FC, \tag{2.3}$$

$$INPUT \rightarrow [CONV \rightarrow RELU \rightarrow FC] * 2 \rightarrow FC \rightarrow RELU \rightarrow FC \tag{2.4}$$

реалізують один згортковий шар між кожним шаром пулінгу.

$$INPUT \rightarrow [CONV \rightarrow RELU \rightarrow CONV \rightarrow RELU] * 3 \rightarrow [FC \rightarrow RELU] * 2 \rightarrow FC \tag{2.5}$$

реалізують 2 шари згортки, що укладаються перед кожним пулінговим шаром.

Це хороша реалізація для великих і більш глибоких мереж, оскільки кілька складених шарів згортки можуть розвивати більш складні властивості вхідних даних перед деструктивною операцією пулінгу. Як правило, віддають перевагу стеку шарів з невеликими фільтрами, як одному великому рецептивному полю згорткового шару. Припустимо, ви, дотримуючись нелінійності між шарами, викладаєте 3 шари згортки розміру 3 × 3 один на одного. При такому розташуванні, кожен нейрон на першому згортковому шарі має видом 3 × 3 вхідного обсягу. Нейрони на другому шарі мають видом 3 × 3 першого шару і 5 × 5 вхідного обсягу. Точно також нейрони на третьому шарі мають видом 3 × 3 другого шару і 7 × 7 вхідного обсягу. Припустимо замість цих трьох згорткових шарів, ми хочемо використовувати тільки один згорткових шар з рецептивних полями розміру 7 × 7. Нейрони такого шару матимуть рецептивне поле вхідного обсягу, аналогічне просторової протяжності 7 * 7, але з деякими недоліками. По-перше, нейрони будуть обчислювати лінійну функцію вхідного обсягу, в той час як стек з 3 шарів

згортки містить нелінійності, що підкреслюють властивості шарів. По-друге, якщо ми припустимо, що весь вхідний обсяг має C каналів, тоді один згортковий шар розміру 7 * 7 міститиме $3 * (7 * 7 * C) = 49C2$ параметрів, в той час як стек з 3 шарів містить лише $3 * (C * (3 * 3 * C)) = 27C2$ параметрів.

Очевидно, що використання декількох згорткових шарів з невеликими фільтрами дозволяє підкреслити більш потужні особливості вхідної інформації з меншою кількістю параметрів. Недоліком є більший обсяг використовуваної пам'яті для збереження всіх проміжних результатів.

Для багатьох застосувань тренувальних даних доступно мало. А згорткові нейронні мережі зазвичай вимагають великої кількості тренувальних даних, щоби запобігати перенавчанню. Поширеною методикою є тренувати мережу на ширшому наборі даних з пов'язаної області визначення. Щойно параметри мережі зійшлися, виконується додатковий етап тренування із застосуванням даних з області визначення для тонкого налаштування ваг мережі. Це дозволяє згортковим мережам застосовуватися до задач з невеликими тренувальними наборами [7].

2.4 Навчання згорткової нейронної мережі

На початковому етапі нейронна мережа є ненавченою (ненастроєною). У загальному змісті під навчанням розуміють подальше попереднє створення образу на вході нейромережі, з навчального набору, потім отримана відповідь порівнюється з бажаним виходом. Тому цю детальну помилку необхідно поширювати на всі пов'язані нейронні мережі.

Таким чином навчання нейронної мережі зводиться до мінімізації функції помилки, шляхом коригування вагових коефіцієнтів синаптичних зв'язків між нейронами. Під функцією помилки розуміється різниця між отриманою відповіддю і бажаною відповіддю. Наприклад, на вхід був поданий образ особи,

припустимо, що вихід нейромережі був 0.73, а бажаний результат 1 (тому що образ особи), отримаємо, що помилка мережі є різницею, тобто 0.27. Потім ваги вихідного шару нейронів коригуються відповідно до помилки. Для нейронів вихідного шару відомі їхні фактичні та бажані значення виходів. Тому налаштування ваг зв'язків для таких нейронів є відносно простим. Однак для нейронів попередніх шарів налаштування не настільки очевидне. Довгий час не було відомо алгоритму поширення помилки по прихованим шарам.

Для навчання згорткової нейронної мережі використовують алгоритм зворотного поширення помилки. Цей метод навчання багатошарової нейронної мережі називається узагальненим дельта-правилом. Для вихідного шару коригування ваг інтуїтивно зрозуміло, але для прихованих шарів довгий час не було відомо алгоритму. Ваги прихованого нейрона повинні змінюватися прямо пропорційно помилці тих нейронів, з якими даний нейрон пов'язаний. Ось чому зворотнє поширення цих помилок через мережу дозволяє коректно налаштовувати ваги зв'язків між усіма шарами. У цьому випадку величина функції помилки зменшується і мережа навчається.

Основні співвідношення методу зворотного поширення помилки отримані при наступних позначеннях:

E_p – величина функції похибки для образа p;

t_{pj} –бажаний вихід нейрону j для образа p;

y_{pj} – активований вихід нейрону j для образа p;

s_{pj} – зважена сума виходів зв’язаних нейронів попереднього шару на усі зв’язки, також визначається як неактивований стан нейрону j для образа p;

w_{pj}– вага зв’язку між i та j нейронами;

Величина похибки визначається за формулою середньоквадратичної похибки:

$$E_p = \frac{1}{2}\sum_j (t_{pj} - y_{pj})^2, \tag{2.6}$$

Неактивований стан нейрона j для образу p визначається за наступною формулою:

$$s_{pj} = \sum_j w_{pj} y_{pj}. \tag{2.7}$$

Вихід кожного нейрона j є значенням функції активації f_j, яка переводить нейрон в активований стан. В якості функції активації може використовуватися будь-яка монотонна функція, що безперервно диференціюється. Активований стан нейрона визначається за наступною формулою:

$$y_{pj} = f_j(s_{pj}). \tag{2.8}$$

Як метод мінімізації помилки використовується метод градієнтного спуску, суть цього методу зводиться до пошуку мінімуму (або максимуму) функції за рахунок руху вздовж вектору градієнта. Для пошуку мінімуму рух має здійснюватися в напрямку антиградієнта.

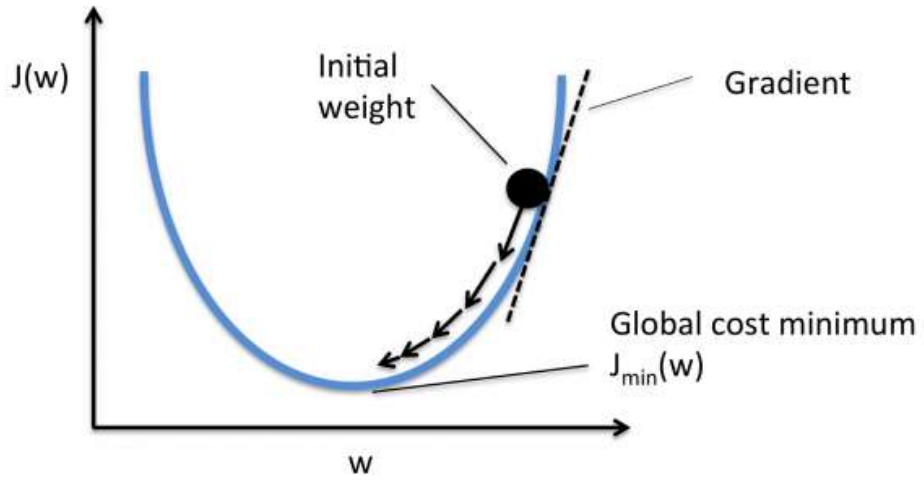


Рисунок 2.6 - Метод градієнтного спуску

Градієнт функції втрат являє собою вектор часткових похідних, що визначаються за формулою:

$$\nabla \mathrm{E}(\mathrm{W}) = [\frac{dE}{dw_1}, \dots, \frac{dE}{dw_n}], \tag{2.9}$$

де $\nabla \mathrm{E}(\mathrm{W})$ – градієнт функції втрат від матриці ваг, $\frac{dE}{dw}$ – часткова похідна функції похибки по вазі нейрону, n – загальна кількість ваг в мережі.

Похідну функції похибки по конкретному образу можна записати за формулою:

$$\frac{dE}{dw_{ij}} = \frac{dE}{dy_j} * \frac{dy_j}{ds_j} * \frac{ds_j}{dw_{ij}}, \tag{2.10}$$

де $\frac{dE}{dw_{ij}}$ – значення похідної похибки по вазі dw_{ij} між i та j нейронами, $\frac{dE}{dy_j}$ – похибка нейрону j, $\frac{dy_j}{ds_j}$ – значення похідної функції активації по її аргументу для нейрону j, $\frac{ds_j}{dw_{ij}}$ – вихід i нейрону попереднього шару (по відношенню до нейрону j)

Похибка нейрона $\frac{dE}{dy_j}$ зазвичай записується у вигляді символу d. Для вихідного шару похибка визначена в явному вигляді, якщо взяти похідну від формули 2.6, то отримаємо t мінус y , тобто різницю між бажаним і отриманим виходом. Для визначення похибки для прихованих шарів був придуманий алгоритм зворотного поширення похибки. Суть його полягає в послідовному обчисленні похибок прихованих шарів за допомогою значень похибки вихідного шару, тобто значення похибки поширюються по мережі в зворотному напрямку від виходу до входу.

Похибка d для прихованого шару розраховується за формулою:

$$d_i = \frac{dy_i}{ds_i} * \sum_j d_j * w_{ij}, \tag{2.11}$$

де $\frac{dy_i}{ds_i}$ – значення похідної функції активації по її аргументу для нейрону j, d_i – похибка нейрону i прихованого шару, d_j – похибка нейрону j наступного шару, w_{ij} – усі зв’язки між нейроном i прихованого шару і нейроном j вихідного шару.

Алгоритм поширення помилки зводиться до наступних етапів:

– пряме поширення сигналу по мережі, обчислення стану нейронів;

– обчислення значення помилки d для вихідного шару;

– зворотне поширення: послідовно від кінця до початку для всіх прихованих шарів обчислюємо d за формулою 2.10;

– оновлення ваг мережі на розрахункову раніше d помилки [8,9].

2.5 Розробка структури згорткової нейромережі

Згорткові нейронні мережі - це потужний інструмент обробки зображень, в якому ядро використовується для перетворення зображення шляхом його ітерації та виконання обчислень за допомогою цього ядра. Ядра різних типів можуть бути використані для створення виявлення країв, заривання та багатьох інших ефектів. Кожна згорткова нейронна мережа зазвичай складається із згорткового шару, за яким випрямлений лінійний активаційний шар, а потім шар об'єднання. Набори цих трьох шарів можна використовувати послідовно з різними параметрами для створення нейронної мережі, яка добре підходить для задач класифікації зображень. Вихід згорткових шарів потім надходить у повнозв`язну мережу, яка, як правило, буде набагато меншою, ніж мережі глибоких переконань та подібні типи мереж. Оскільки згортка зображення за

допомогою відповідного ядра витягує певні риси із зображення, а потім випрямлена функція лінійної активації усуває негативні значення, а потім, нарешті, шар об'єднання зменшує розміри вихідних даних, зберігаючи важливі особливості, цілком підходить для вилучення належних ознак, класифікації, якою б абстрактною вона не була, як різниця між собакою та котом.

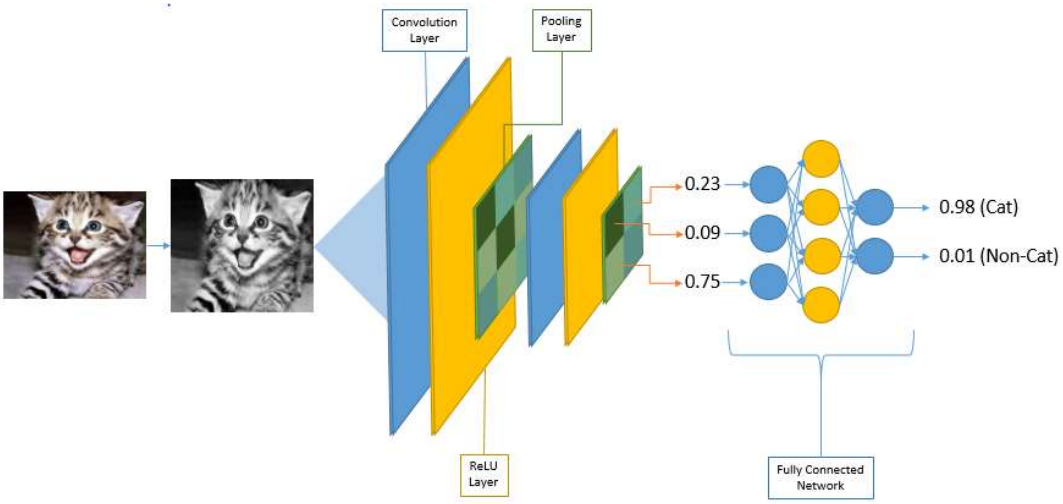


Рисунок 2.7 – Схема згорткової мережі

Згорткові шари, розміщені перед повнозв`язною, в основному є дуже простим, але потужним етапом попередньої обробки для зменшення розмірів вхідних даних, зберігаючи важливі особливості. Зрештою, повнозв`язна мережа видає такий самий вид класифікації, як і будь-яка звичайна нейронна мережа [10].

Для створюваної нейромережевої системи було обрано такі параметри:

– розмір зображення – 50x50;
– максимальна кількість ітерацій на сторінці – 500;
– помилка для завершення тренування на сторінці – 0,02;
– помилка для завершення ітераційного тренувального процесу – 1;
– кількість шарів – 12;
– кількість вхідних шарів – 1;
– кількість згорткових шарів – 3;

– кількість шарів ReLu – 2;
– кількість з’єднувальних шарів – 2;
– кількість шарів на повнозв’язному шарі – 3;
– кількість нейронів SoftMax шару – 2;

Згідно з параметрами було створено структуру розроблюваної нейромережевої системи (рис.2.8):

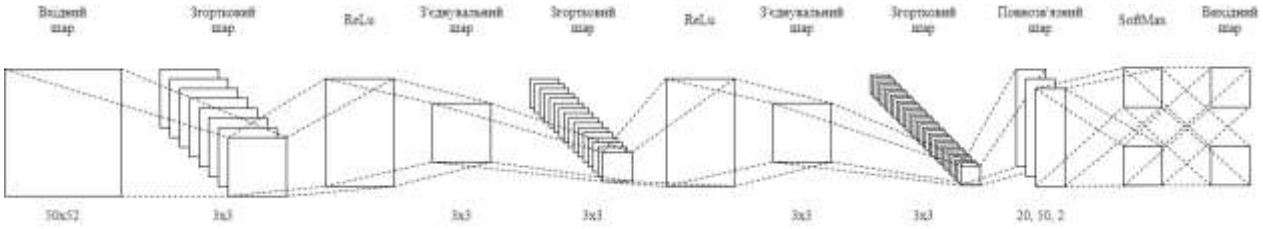


Рисунок 2.8 – Структура нейромережевої системи

2.6 Структура інформаційної технології нейромережевого розпізнавання зображень тварин

Структура інформаційної технології нейромережевого розпізнавання зображень тварин, зображена на рис. 2.9.

В основі цієї інформаційної технології лежить згорткова нейронна мережа CONVNETSHARP Network. Оскільки кожна нейронна мережа може працювати у двох режимах: режимі навчання та режимі функціонування, то структура інформаційної технології буде мати дві частини. Ліва частина структури ІТ на рис. 2.9 віддображає режим навчання згорткової нейронної мережі, а права частина структури на рис. 2.9 віддображає режим функціонування. Для роботи інформаційної технології нейромережевого розпізнавання зображень тварин в режимі навчання вхідною є інформація у вигляді множини зображень для навчання і тестування. У цій лівій частині спочатку здійснюється процес створення (генерації) згорткової нейромережі відповідно до обраної архітектури CONVNETSHARP Network. Потім відбувається процес початкової

ініціалізація ваг мережі невеликими випадковими значеннями у певному діапазоні. Потім здійснюється процес навчання нейронної мережі на основі навчальної множини зображень. Після цього згорткова нейронна мережа підготовлена до розпізнавання зображень тварин.

Рисунок 2.9 - Структура інформаційної технології нейромережевого розпізнавання зображень тварин

Для роботи інформаційної технології розпізнавання зображень тварин на основі згорткової нейронної мережі CONVNETSHARP Network в режимі функціонування вхідною є інформація у вигляді файлу зображення форматів .jpg, .jpeg, .png.). У цій частині спочатку відбувається процес перетворення зображення до розміру 50x50 пікселів, оскільки згорткова нейронна мережа CONVNETSHARP Network приймає на вхід зображення саме такого розміру. Далі здійснюється процес подання зображення на вхід попередньо навченої згорткової нейронної мережі. Після цього відбуваться процес розпізнавання

нейронною мережею зображення тварини з класифікацією на 2 класи – «кіт» і «не кіт». Далі проходить процес виведення інформації про результат розпізнавання зображення тварини.

Таким чином, розроблена структура інформаційної технології розпізнавання зображень тварин з використанням згорткової нейронної мережі може бути використана для подальшої розробки програмних засобів.

2.7 Висновок до розділу 2

У розділі було обґрунтовано вибір типу нейронної мережі для задачі розпізнавання зображень – згорткова нейромережа ConvNetSharp. Була розроблена структура та проаналізована математична модель згорткової нейромережі, обгрунтовано вибір функції активації нейронів. Було розроблено структуру інформаційної технології нейромережевого розпізнавання зображень тварин.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН НА ОСНОВІ CONVNETSHARP

3.1 Обґрунтування вибору середовища програмування

Зараз існує безліч середовищ розробки, які відрізняються дизайном, функціями, можливостями, мовам програмування, які вони підтримують і т.п. Кожен програміст обирає середовище для себе, те, яке йому більш всього імпонує, підходить за функціоналом та його мови розробки. Проведемо порівняння 4 популярних середовищ розробки [11,12] програмного забезпечення (табл. 3.1).

Таблиця 3.1 – Порівняння середовищ розробки програмного забезпечення

Назва	Visual Studio Code	Eclipse	Notepad++	Sharp Develop
Підсвітка синтаксису	+	+	+	+
Автодоповнення	+	+		+
Звертання блоків тексту	+	+	+	+
Список доступних функцій	+	+		+
Компілятор	+	+		+
Довідка	+	+		
Портативність				+

Для використання було обрано Visual Studio Code [13], оскільки воно є звичним та має більшість наведених в таблиці параметрів. Visual Studio Code включає в себе редактор коду, який має технологію IntelliSense, що полегшує роботу програміста.

3.2 Обґрунтування вибору мови програмування

Мови програмування – це формальні комп'ютерні мови. Звичайно, в реальному житті ними ніхто не розмовляє. Вони потрібні для "переговорів" з машинами. На різних мовах програмування пишуться різні програми, які висловлюють певні алгоритми або контролюють поведінку комп'ютера.

Програмний код, написаний на тій чи іншій мові програмування, оточує нас всюди. Інтернет-платформи, мережа інтернету речей, будь-які дії вбудованої техніки і домашніх гаджетів, керування автомобілем і навіть перемикання сигналів світлофора – все це пов'язано з програмуванням. Мов придумано вже більше восьми тисяч. Одні затребувані у всьому світі, інші потрібні тільки декільком програмістам-творцям. Існуючі мови постійно розвиваються, а нові – придумуються, щоб в якийсь момент доповнити або замінити застарілі [14].

Порівняємо 5 найпопулярніших мов [15] (табл. 3.2).

Таблиця 3.2 – Порівняння мов програмування

Мова	C#	Java	JavaScript	PHP	Python
Імперативне	+	+	+	+	+
Об'єктно-орієнтоване	+	+	+	+	+
Функціональне	+		+		+
Процедурне				+	
Узагальнене	+	+			
Рефлективне	+	+	+	+	+
Подійно-орієнтоване	+				

Для розробки програмного забезпечення було обрано мову програмування C# [16], оскільки вона була створена з урахуванням сильних і слабких особливостей інших мов, зокрема Java і C++і широко застосовується для програмування нейронних мереж [17].

3.3 Використання бібліотеки ConvNetSharp

ConvNetSharp – бібліотека, написана на мові C# для навчання та оцінки згорткових нейронних мереж, з можливістю використання як CPU так і CUDA обчислень на GPU. Бібліотека активно розвивається та раз в декілька місяців випускаються нові версії бібліотеки.

ConvNetSharp надає 3 способи створення нейронної мережі: [18]:

Таблиця 3.3 – 3 способи створення нейронної мережі

Основні шари	Потокові шари	Графік обчислення
Відсутній графік обчислень	Шари створюють граф обчислень за кадром	Чистий потік
Мережа, організована за допомогою укладання шарів	Мережа, організована за допомогою укладання шарів	«OPS» пов'язані між собою. Може реалізувати більш складні мережі.

Для створення нейронної мережі у коді програми можна використовувати як звичайні класи, що будуть містити в собі параметри мережі, такі як кількість та типи шарів, модель тренування та інші, так і Fluent API.

Fluent API (Вільний інтерфейс) – спосіб конструювання об’єктно орієнтованого API, в якому читабельність коду є близькою до звичайного прозового тексту. Він зазвичай реалізується за допомогою ланцюжка методів для реалізації каскадного методу, конкретно шляхом повернення кожному методу об’єкта, до якого він приєднаний. Якщо говорити більш абстрактно, вільний інтерфейс передає контекст інструкцій наступного виклику в ланцюгу методів, де існує контекст.

Вільний інтерфейс також може бути використаний для ланцюжка набору методів, який працює / використовує один і той же об’єкт. Замість того, щоб створювати клас замовника, ми можемо створити контекст даних [19].

За допомогою ConvNetSharp можна створити будь-яку згорткову нейронну мережу, використовуючи різний порядок шарів різного типу.

3.4 Структура та алгоритм роботи програмного забезпечення

На етапі проектування програмного продукту було визначено основні модулі програмного продукту, а саме: інтерфейс, модуль вибору типу мережі, модуль завантаження моделі мережі, модуль тренування мережі, модуль виведення графіку помилки навчання, модуль завантаження зображення, модуль класифікації зображення, модуль виведення результату. Усі модулі та зв’язки між ними зображені на створеній структурній схемі (див. рис. 3.1).

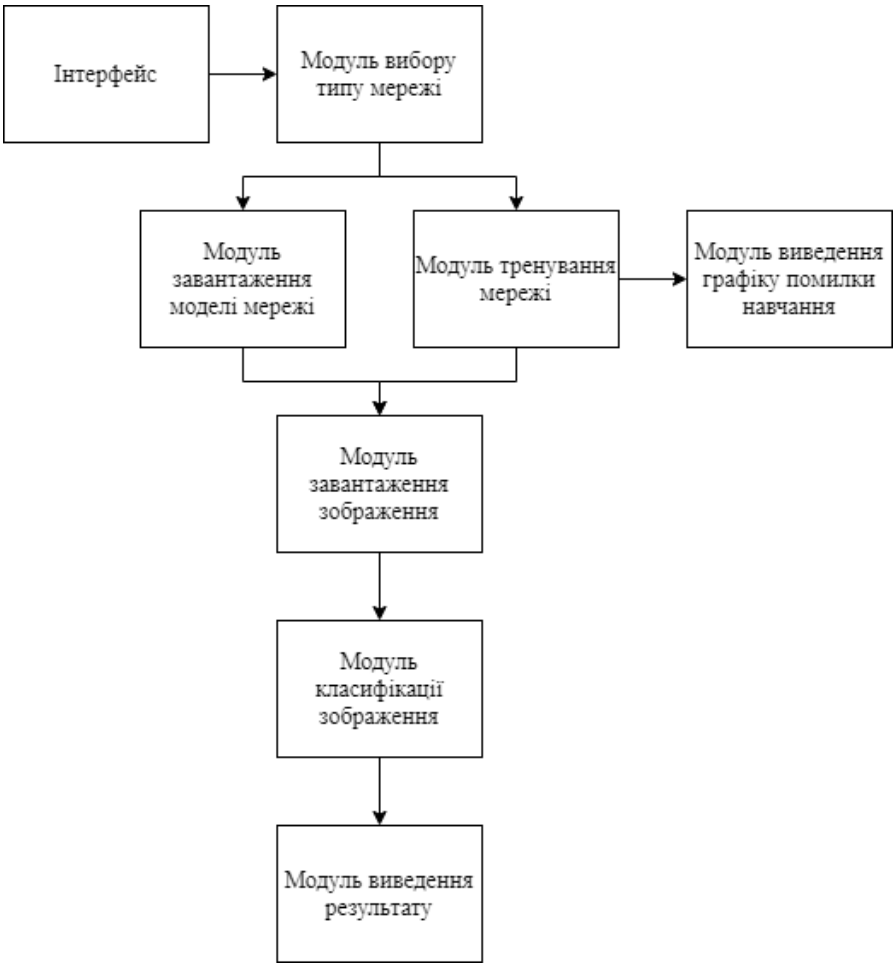


Рисунок 3.1 – Структурна схема програми

Опишемо деякі визначені модулі.

Модуль тренування мережі тісно пов’язаний з модулем завантаження зображень та модулем виведення графіку помилки навчання. Основною задачею модуля є тренування мережі за завантаженими зображеннями, а також збереження натренованої моделі у файл для подальшого використанні при класифікації зображень.

Модуль класифікації зображення включає у себе нейронну мережу, яка проводить процедуру класифікації. Результати передаються на модуль виведення результату.

Також було спроектовано загальний алгоритм роботи програми, який представлено на рисунку 3.2:

Рисунок 3.2 – Загальний алгоритм роботи програми

Під час етапу проектування було розроблено структуру та алгоритм розроблюваного програмного забезпечення.

3.5 Реалізація згорткової нейронної мережі для розпізнавання зображень за допомогою ConvNetSharp

Попередня обробка вхідного зображення.

Вхідне зображення спочатку підлягає попередній обробці, яка полягає у перетворенні кольорового зображення у «сіре», а потім зменшенні вхідного зображення до 50x50 пікселів, що різко зменшує кількість вхідних нейронів до 2500. Це також зменшить складність та особливості, які нейромережі потрібно аналізувати. Бібліотека EmguCV, яка є .NET-обгорткою OpenCV, дуже зручна у виконанні цих завдань.

Нормалізація значень яскравостей пікселів.

Оскільки значення яскравості пікселя кодується одним байтом (8 біт), то воно має 256 градацій сірого. Нам потрібно перетворити значення яскравості у діапазон від 0 до 1, щоб його можна було подати у нейронну мережу. EmguCV зображення має властивість Bytes, що дає нам масив значень пікселів.

Створення згорткової нейронної мережі.

У бібліотеці ConvNetSharp згорткові шари, об’єднальні шари та повнозв’язні шари доступні на вибір для побудови нейронної мережі, специфічної для класифікації зображень. У коді нижче параметри та послідовність шарів визначаються конструктором.

```
public ConvNetSharpNetwork()
{
    network = new Net<double>();

    network.AddLayer(new InputLayer(50, 52, 1));
    network.AddLayer(new ConvLayer(3, 3, 8) { Stride = 1, Pad = 2
});
```

```
            network.AddLayer(new ReluLayer());
            network.AddLayer(new PoolLayer(3, 3) { Stride = 2 });
            network.AddLayer(new ConvLayer(3, 3, 16) { Stride = 1, Pad = 2
});
            network.AddLayer(new ReluLayer());
            network.AddLayer(new PoolLayer(3, 3) { Stride = 2 });
            network.AddLayer(new ConvLayer(3, 3, 32) { Stride = 1, Pad = 2
});
            network.AddLayer(new FullyConnLayer(20));
            network.AddLayer(new FullyConnLayer(50));
            network.AddLayer(new FullyConnLayer(2));
            network.AddLayer(new SoftmaxLayer(2));

            trainer = GetTrainerForNetwork(network);
        }
```

Вищевказана мережа містить три згорткові шари та повнозв’язну нейронну мережу на виході для класифікації. Ця мережа навчається за методом стохастичного градієнтного спуску - клас SgdTrainer, наданий тією ж бібліотекою.

```
        private TrainerBase<double> GetTrainerForNetwork(Net<double> net)
        {
            return new SgdTrainer<double>(net)
            {
                LearningRate = 0.003
            };
        }
```

Навчання мережі.

У цій згортковій мережі вхідні та вихідні дані повинні бути перетворені у обсяги тієї ж форми, що і розміри вхідного та вихідного шарів. У ConvNetSharp є допоміжні класи, які дозволяють це зробити легко. Спочатку зображення комбінуються у пакети, а потім перетворюються у томи, щоб пришвидшити

навчальний процес. Пакети розміром 50 чудово працюють для цієї мережі, балансуючи час на сходимість та точність.

```
    public void BatchTrain(double[][] batchInputs, double[][] batchOutputs,
                int iterations, Action<double, int, string> progressCallback)
    {
        trainer.BatchSize = batchInputs.Length;
        foreach (int currentIteration in Enumerable.Range(1, iterations))
        {
            Randomizer.Shuffle(batchInputs, batchOutputs);
            (Volume<double> inputs, Volume<double> outputs) =
                    GetVolumeDataSetsFromArrays(batchInputs, batchOutputs);
            trainer.Train(inputs, outputs);
            var error = network.GetCostLoss(inputs, outputs);
            if (progressCallback != null)
            {
                progressCallback(error, currentIteration, "Supervised");
            }
            inputs.Dispose();
            outputs.Dispose();
            if(this.ShouldStopTraning)
            {
                this.ShouldStopTraning = false;
                break;
            }
        }
    }
```

Вхідні дані у вигляді 2D масиву спочатку перетворюються у сумісну форму та том, подаються в мережу для навчання, а вихід цієї мережі є томом, який перетворюється у 2D масив та повертається.

3.6 Висновок до розділу 3

У третьому розділі було обґрунтовано вибір середовища розробки, мови програмування та фреймворків, які будуть використовуватися при розробці програми та наведено їх основні переваги. У результаті аналізу було обрано середовище розробки Visual Studio Code, мову програмування C# та бібліотеку ConvNetSharp. Було розроблено алгоритм роботи програми та структуру програмного забезпечення розпізнавання зображень тварин. Було проведено реалізацію програмних модулів системи, включно з модулем тренування мережі, виведення графіку помилки навчання та класифікації зображень.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН НА ОСНОВІ CONVNETSHARP

4.1 Тестування програми розпізнавання зображень тварин

Проведемо тестування розробленого програмного забезпечення. Перш за все було проведено навчання нейронної мережі. Для цього потрібно обрати тип мережі ConvNetSharp на початковому вікні, яку будемо навчати (рис.4.1).

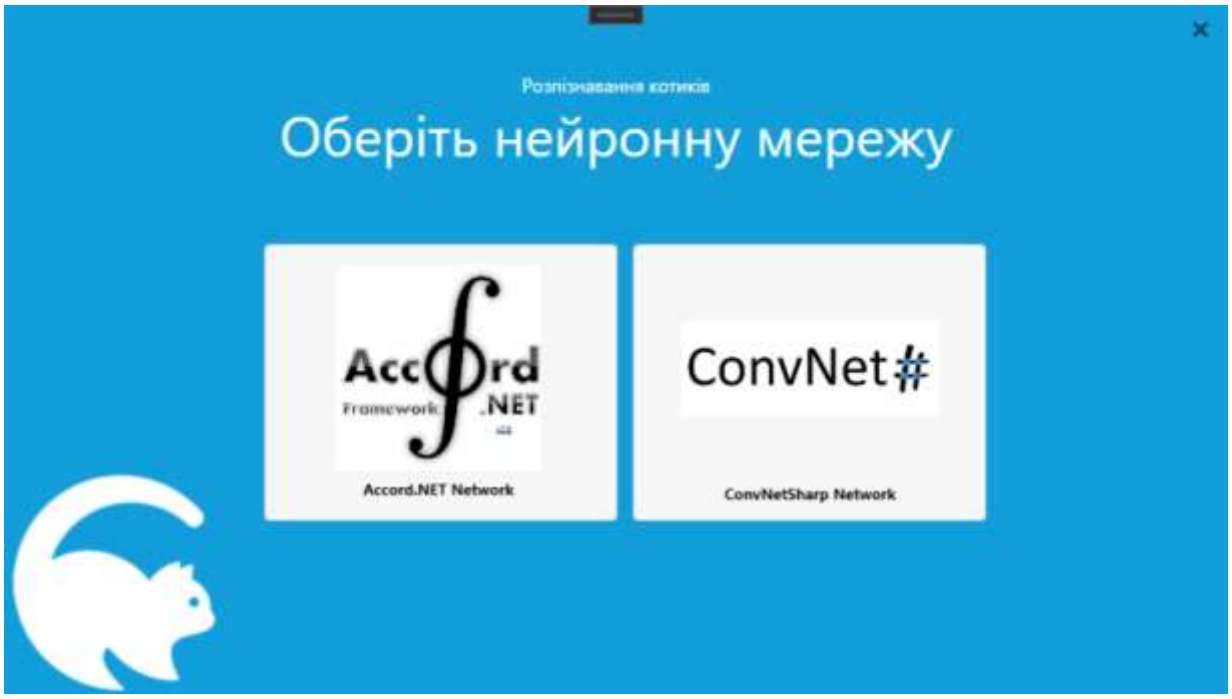


Рисунок 4.1 – Вікно вибору типу мережі

Після того, як обрано тип мережі, переходимо до самого вікна тренування. Щоб відкрити вікно тренування потрібно натиснути на кнопку «Тренування і тестування» на відкритому вікні (рис.4.2). Далі буде відкрито вікно тренування та тестування нейронної мережі (рис.4.3) куди можна завантажити зображення, вказуючи, чи зображено кота на ньому.

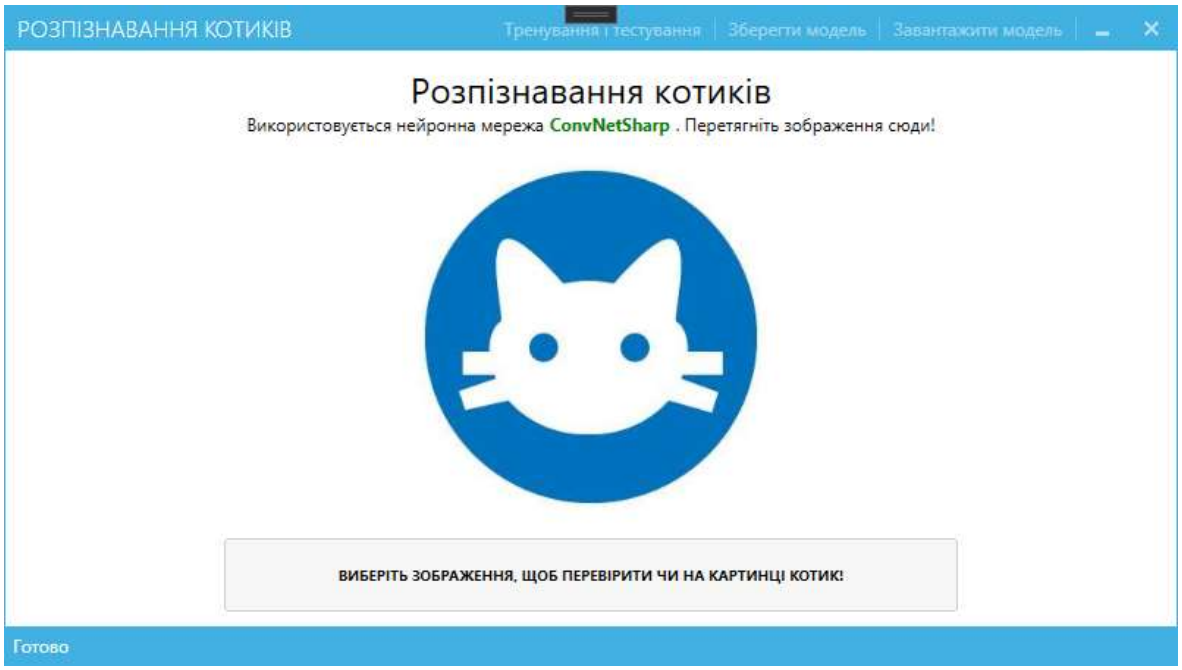


Рисунок 4.2 – Вікно розпізнавання

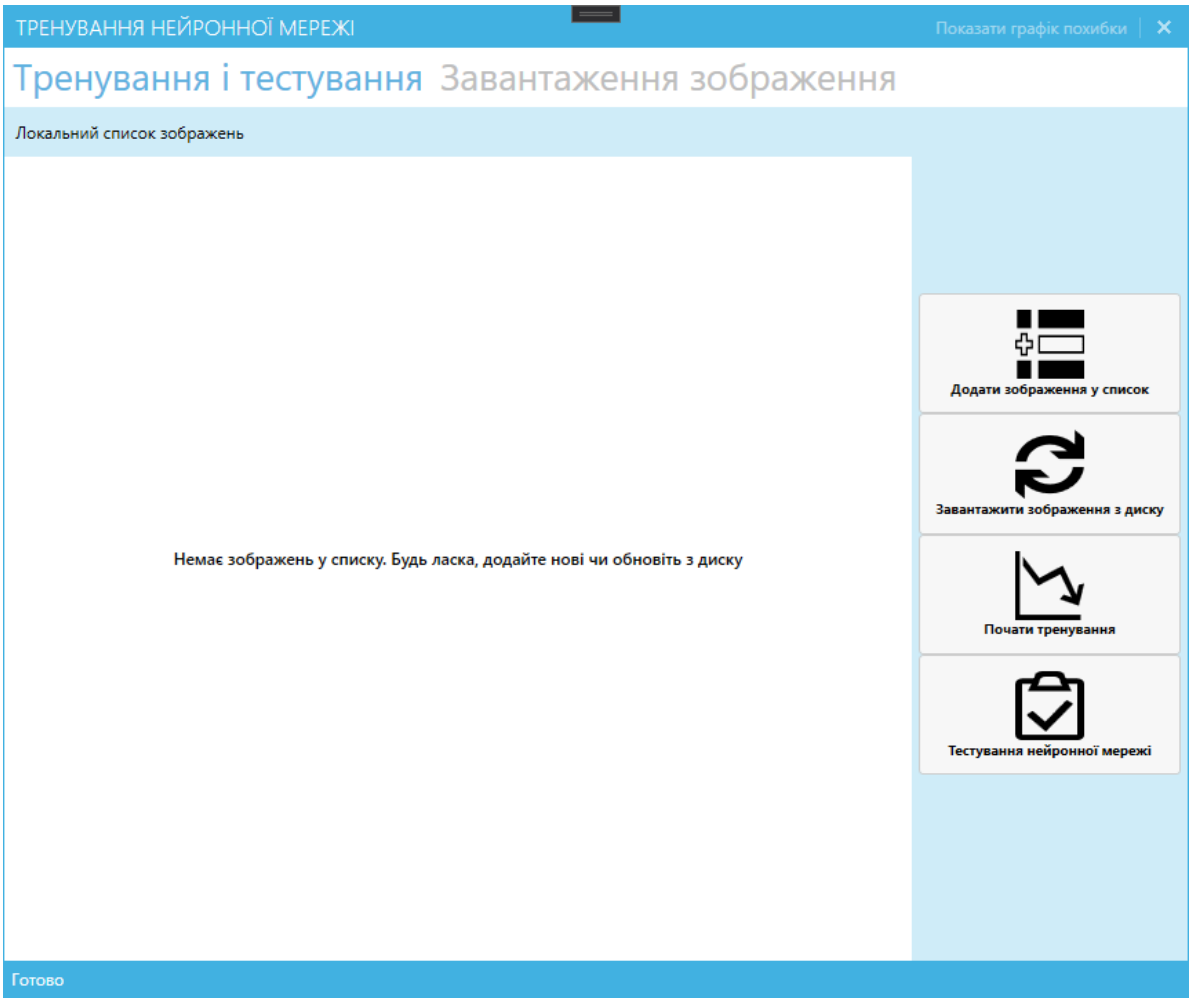


Рисунок 4.3 – Вікно «Тренування і тестування»

Після завантаження тренувальної бази в програму можна розпочати тренування (рис.4.4). При натисканні на кнопку «Показати графік похибки» буде відображено нове вікно, де динамічно відображається теперішнє значення похибки (рис.4.5). Динамічне оновлення графіку дозволяє стежити за процесом навчання мережі.

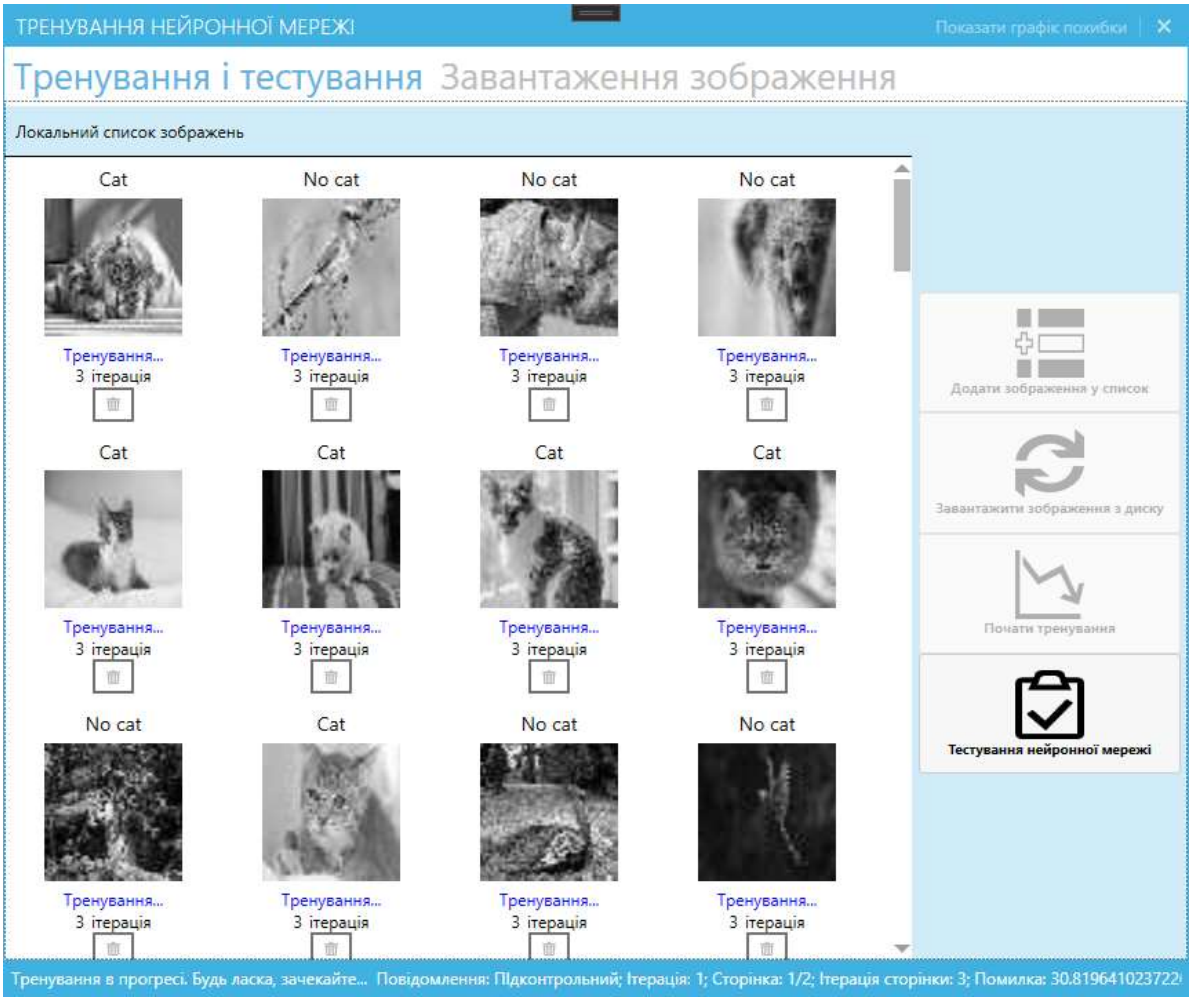


Рисунок 4.4 - Вікно «Тренування і тестування» із завантаженими та початком тренування нейронної мережі

Тренування нашої нейронної мережі проходило на вибірці з 500 зображень, з яких 250 – були зображення котів, а інші 250 – зображеннями інших тварин. Коли процес навчання завершено, потрібно натиснути кнопку «Тестування нейронної мережі», що дозволить перевірити чи правильно мережа класифікує зображення (рис.4.6).

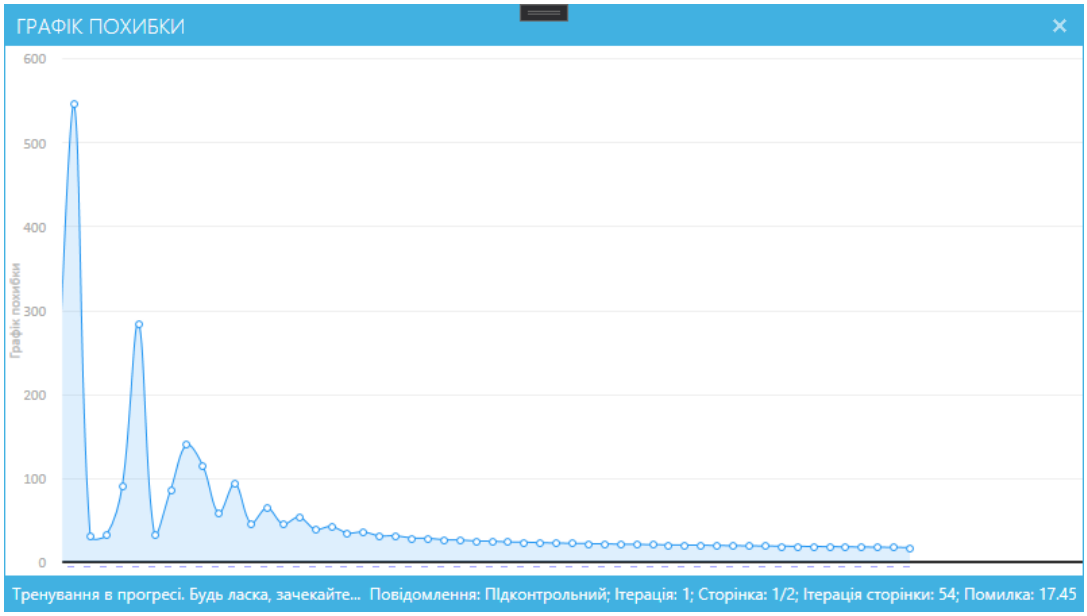


Рисунок 4.5 – Графік зміни похибки навчання

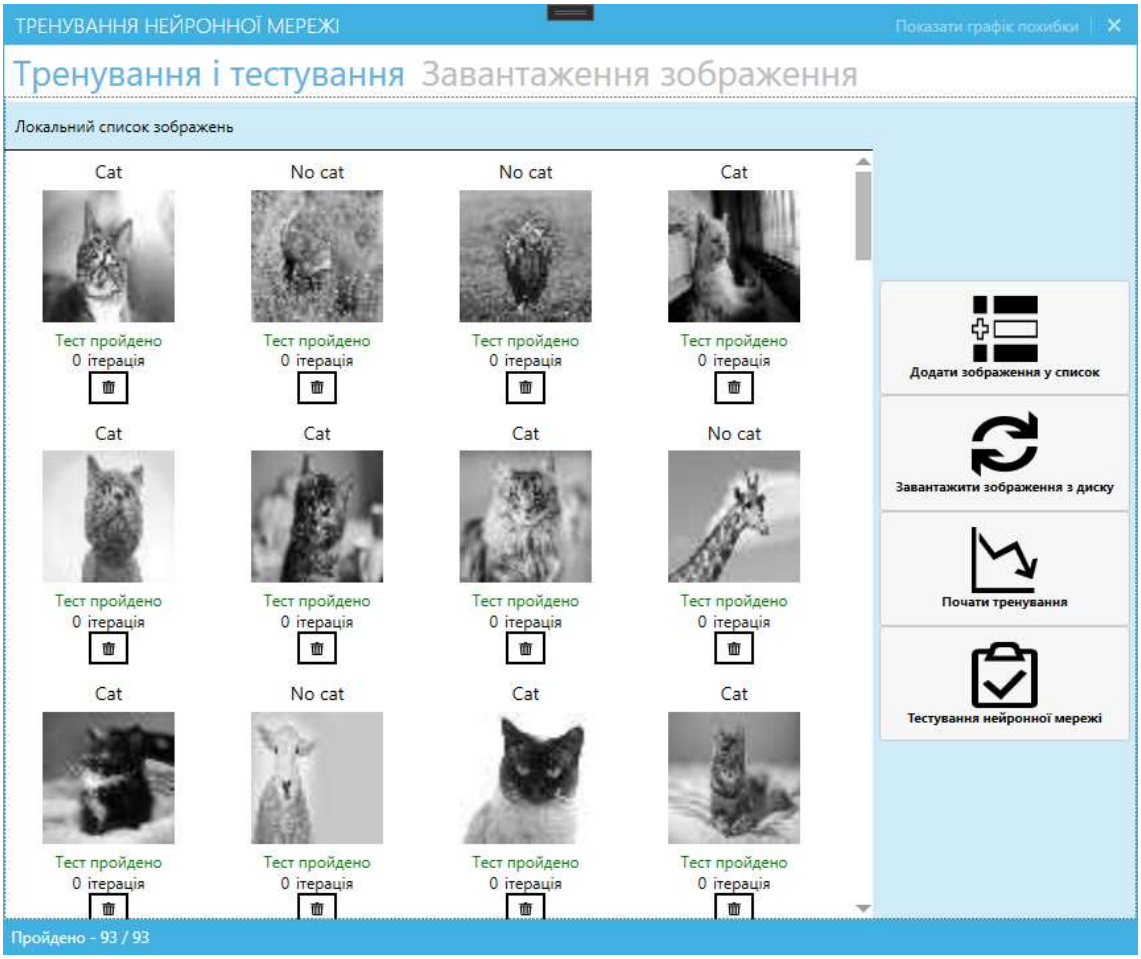


Рисунок 4.6 – Тестування навченої нейронної мережі

Проведемо тестування на зображеннях, які не було включено до тестової вибірки. Для цього завантажимо у вікно (рис.4.2) зображення кота (рис.4.7) та іншої тварини (рис.4.8) та перевіримо правильність класифікації створеної мережі.

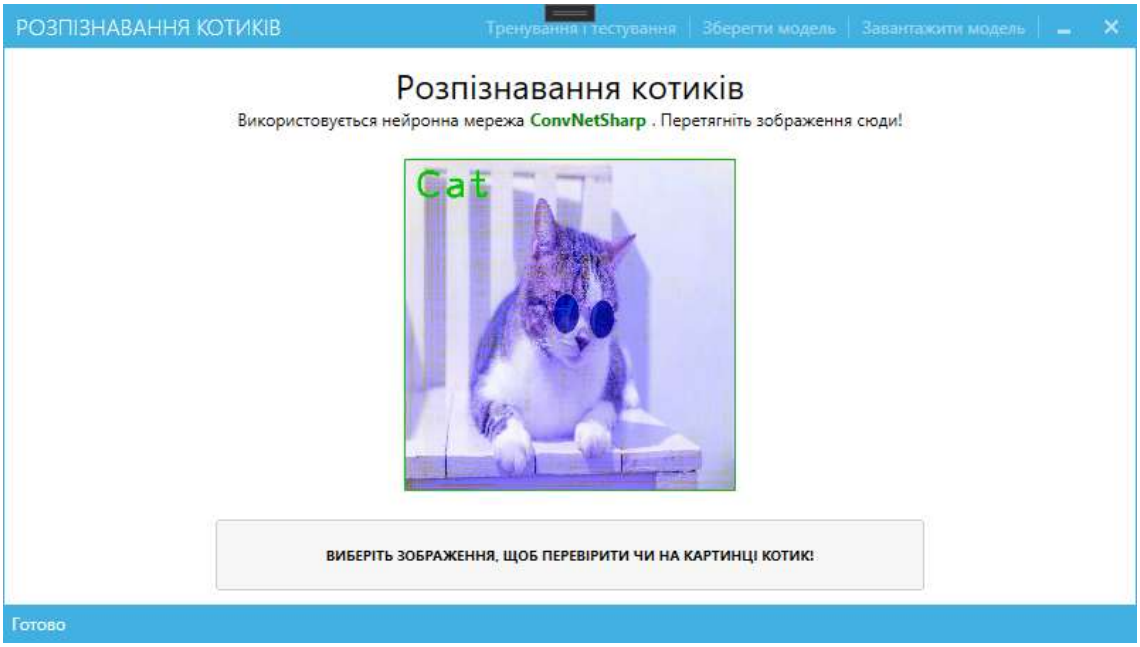


Рисунок 4.7 – Зображення кота у вікні розпізнавання

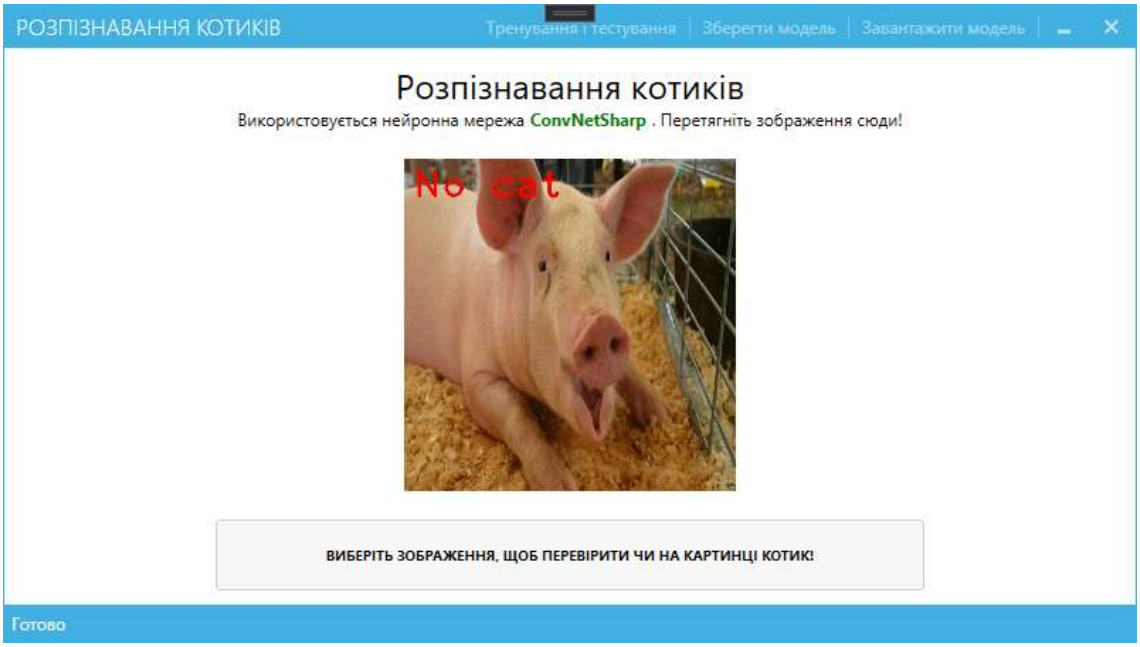


Рисунок 4.8 – Зображення іншої тварини у вікні розпізнавання

4.2 Аналіз результатів роботи програми розпізнавання зображень тварин

Перевірка достовірності розпізнавання нашої нейронної мережі проходило на вибірці з 100 зображень, з яких 50 – були зображення котів, а інші 50 – зображеннями інших тварин.

Тестування показує, що навчена нейронна мережа правильно класифікує 94 зображень із 100, які були у тестовій вибірці. У відсотковому співвідношенні результат становить 94%, тобто помилка – 6%, що означає, що мета роботи та параметри, вказані в індивідуальному завданні, досягнуті.

Результати порівняння параметрів розробленої згорткової нейронної мережі ConvNetSharp Network із глибинною мережею переконань Accord.Net Network, наведено у табл.4.1.

Таблиця 4.1 – Порівняння параметрів розробленої згорткової нейронної мережі ConvNetSharp Network із мережею Accord.Net Network

	Кількість зображень у тестовій вибірці	Кількість вірно розпізнанних зображень	Достовірність розпізнавання	Час навчання мережі на вибірці у 500 зображень
Accord.Net Network	100	83	83 %	8 годин
ConvNetSharp Network	100	94	94%	24 години

Із табл. 4.1 видно, що згорткова нейронна мережа ConvNetSharp Network досягла похибки в 6% (достовірність 94%) та тренувалась біля 24 години. А глибинна мережа переконань Accord.Net Network досягла похибки аж у 17% (достовірність 83%), але потребує набагато менше часу на навчання – 8 годин.

Таким чином, можна зробити висновок, що згорткова нейронна мережа ConvNetSharp Network має порівняно з глибинною мережею переконань Accord.Net Network збільшену на 11% (17-6=11) достовірність розпізнавання.

Тобто мета роботи досягнута – достовірність розпізнавання підвищена. Але ця перевага досягається за рахунок збільшення у 3 рази часу навчання. Значить згорткову нейронну мережу ConvNetSharp Network варто використовувати для задач, де потрібна висока достовірність розпізнавання, але не є критичним невисока швидкість навчання.

4.3 Висновок до розділу 4

У четвертому розділі в результаті тестування програми було доведено її повну працездатність та відповідність поставленому завданню. Тестування нейронної мережі проходило на вибірці з 100 зображень, з яких 50 – були зображення котів, а інші 50 – зображеннями інших тварин. Розроблена згорткова нейронна мережа ConvNetSharp Network досягла похибки в 6% (достовірність 94%) та тренувалась біля 24 години. А глибинна мережа переконань Accord.Net Network (аналог), досягла похибки у 17% (достовірність 83%), але потребує набагато менше часу на навчання – 8 годин. Таким чином, можна зробити висновок, що згорткова нейронна мережа ConvNetSharp Network має порівняно з глибинною мережею переконань Accord.Net Network збільшену на 11% (17-6=11) достовірність розпізнавання. Тобто мета роботи досягнута – достовірність розпізнавання підвищена.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота з розробки та дослідження «Інформаційна технологія нейромережевого розпізнавання зображень тварин» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія нейромережевого розпізнавання зображень

тварин» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи 5оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [20].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 5.1

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	5	5
2. Ринкові переваги (наявність аналогів)	4	4	3
3. Ринкові переваги (ціна продукту)	3	3	3
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	1	1	1
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	2	3
8. Практична здійсненність (наявність фахівців)	4	4	3
9. Практична здійсненність (наявність фінансів)	3	4	3
10. Практична здійсненність (необхідність нових матеріалів)	5	5	4
11. Практична здійсненність (термін реалізації)	5	4	4
12. Практична здійсненність (розробка документів)	5	4	5
Сума балів	44	42	40
Середньоарифметична сума балів $СБ_c$	42,0		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [20].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія нейромережевого розпізнавання зображень тварин» становить 42,0 бала, що, відповідно до таблиці 5.3, свідчить про

комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

5.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості (Вн) для нового технічного рішення розрахуємо за формулою [21]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \quad , \tag{5.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу і-го технічного показника в загальній якості розробки. Коефіцієнт αі визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення і-го технічного показника якості нової розробки.

Відносні значення β_i для різних випадків розраховуємо за такими формулами:

- для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{нi}}{I_{ai}} \quad , \tag{5.2}$$

де $I_{ні}$ та $I_{на}$ – чисельні значення конкретного і-го технічного показника якості відповідно для нової розробки та аналога;

- для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{ні}}; \tag{5.3}$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 5.4.

Таблиця 5.4 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Вимоги до оперативної пам'яті	GB	4	1	4	0,2
Рекомендоване місце на диску	GB	8	4	2	0,1
Точність розпізнавання	%	75	90	1,2	0,25
Вимоги до CPU	бал	6	7	0,85	0,25
Захищеність БД	бал	7	7	1	0,2

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 4 \cdot 0{,}2 + 2 \cdot 0{,}1 + 1{,}2 \cdot 0{,}25 + 0{,}85 \cdot 0{,}25 + 1 \cdot 0{,}2 = 1{,}71.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,71 рази.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія нейромережевого розпізнавання зображень тварин», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.3.1 Витрати на оплату праці

До статті «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.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [20]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (5.4)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=24 дні.

$$З_о = 15100{,}00 \cdot 36 / 24 = 22650{,}00 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
1. Керівник НДР	15100,00	629,17	36	22650,00
2. Ст. науковий співробітник у сфері розпізнавання зображень	14800,00	616,67	24	14800,00
3. Інженер-програміст	14600,00	608,33	24	14600,00
4. Фахівець-консультант	14500,00	604,17	8	4833,33
5. Технік	7520,00	313,33	10	3133,33
Всього				60016,67

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія нейромережевого розпізнавання зображень тварин» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \qquad (5.5)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \qquad (5.6)$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=6700,00 грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [20];

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середнє число робочих днів в місяці, приблизно $Т_р$ = 24 дн;

$t_{зм}$ – тривалість зміни, год.

$С_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}7 / (24 \cdot 8) = 65{,}26$ *грн.*

$З_{р1} = 65{,}26 \cdot 16{,}00 = 1044{,}08$ *грн.*

Таблиця 5.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
1. Встановлення допоміжного обладнання	16,00	2	1,10	65,26	1044,08
2. Інсталяція програмного забезпечення	12,00	4	1,50	88,98	1067,81
3. Встановлення модулів розпізнавання	16,00	5	1,70	100,85	1613,58
Всього					3725,48

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (60016,67 + 3725,48) · 11 / 100% = 7011,64 грн.

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (60016,67 + 3725,48 + 7011,64) · 22 / 100% = 15565,83 грн.

5.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційна технологія нейромережевого розпізнавання зображень тварин».

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (5.9)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j = 1,1 \ldots 1,15$);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$М_1 = 3,0 \cdot 292,00 \cdot 1,1 - 0,000 \cdot 0,00 = 963,60$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір офісний А4 білий (80%)	292,00	3,0	0,000	0,00	963,60
Диск оптичний (CD-R)	136,00	3,0	0,000	0,00	448,80
Органайзер офісний ОФГХ-145-А	198,00	4,0	0,000	0,00	871,20
Канцелярське приладдя	185,00	4,0	0,000	0,00	814,00
Тонер HP/Canon-26A-JL (для заправки картриджа)	5630,00	0,0	0,000	0,00	123,86
FLASH-пам’ять (32 Gb)	450,00	1,0	0,000	0,00	495,00
Всього					3716,46

5.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія нейромережевого розпізнавання зображень тварин» відсутні.

5.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення. Витрати на виготовлення та придбання спецустаткування відсутні.

5.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.10)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 8410,00 \cdot 1 \cdot 1,11 = 9335,10$ грн.

Отримані результати зведемо до таблиці:

Таблиця 5.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows	1	8410,00	9335,10
Прикладний пакет Microsoft Office	1	7770,00	8624,70
Середовище розробки Visual Studio Code	1	12310,00	13664,10
Бібліотека ConvNetSharp	1	13400,00	14874,00
Програмне забезпечення програмування мовою C#	1	5050,00	5605,50
Всього			52103,40

5.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.11)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (28300{,}00 \cdot 2) / (2 \cdot 12) = 2358{,}33$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер розробника програмного забезпечення	28300,00	2	2	2358,33
Робоче місце інженера-програміста	9520,00	6	2	264,44
Пристрої виведення інформації	8500,00	5	2	283,33
Оргтехніка	7850,00	5	2	261,67
Приміщення лабораторії розробки інформаційних систем	285000,00	25	2	1900,00
Електронно-обчислювальний комплекс обробки зображень	54223,00	3	2	3012,39
Всього				8080,17

5.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (5.12)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е = 6,20$ грн;

$К_{вп і}$ – коефіцієнт, що враховує використання потужності, $К_{вп і} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$В_е = 0,45 \cdot 240,0 \cdot 6,20 \cdot 0,95 / 0,97 = 669,60$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер розробника програмного забезпечення	0,45	240,0	669,60
Робоче місце інженера-програміста	0,12	220,0	163,68
Пристрої виведення інформації	0,08	8,0	3,97
Оргтехніка	0,65	7,5	30,23
Електронно-обчислювальний комплекс обробки зображень	0,75	200,0	930,00
Всього			1797,47

5.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Інформаційна технологія нейромережевого розпізнавання зображень тварин» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.13)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 22%.

$B_{св}$ = (60016,67 + 3725,48) · 22 / 100% = 14023,27 грн.

5.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.14)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 33%.

$B_{сп}$ = (60016,67 + 3725,48) · 33 / 100% = 21034,91 грн.

5.3.11 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» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.15)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 55%.

$I_в$ = (60016,67 + 3725,48) · 55 / 100% = 35058,18 грн.

5.3.12 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» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.16)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 130%.

$В_{нзв}$ = (60016,67 + 3725,48) · 130 / 100% = 82864,79 грн.

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія нейромережевого розпізнавання зображень тварин» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.17)$$

$B_{заг}$= *60016,67 +3725,48 +7011,64 +15565,83201 +3716,46 +0,00 +0,00 + 52103,40 + 8080,17 +1797,47 +14023,27 +21034,91 +35058,18 +82864,79 = 304998,26 грн.*

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

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.18)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 304998,26 / 0,9 = 338886,96 грн.

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій 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.

Результати дослідження проведені за темою «Інформаційна технологія нейромережевого розпізнавання зображень тварин» передбачають комерціалізацію протягом 4-х років реалізації на ринку і можуть бути класифіковані як розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	1500	2500	2500	1800

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 9500 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 8200,00 грн;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 842,50 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [20]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.19)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2022 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =35%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2022 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (842{,}50 \cdot 9500{,}00 + 9042{,}50 \cdot 1500) \cdot 0{,}83 \cdot 0{,}35 \cdot (1\text{-}0{,}18/100\%) = 5137594{,}18$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (842{,}50 \cdot 9500{,}00 + 9042{,}50 \cdot 4000) \cdot 0{,}83 \cdot 0{,}35 \cdot (1\text{-}0{,}18/100\%) = 10522628{,}99$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (842{,}50 \cdot 9500{,}00 + 9042{,}50 \cdot 6500) \cdot 0{,}83 \cdot 0{,}35 \cdot (1\text{-}0{,}18/100\%) = 15907663{,}80$ грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 = (842{,}50 \cdot 9500{,}00 + 9042{,}50 \cdot 8300) \cdot 0{,}83 \cdot 0{,}35 \cdot (1\text{-}0{,}18/100\%) = 19784888{,}87$ грн.

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

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.20)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}15$;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП = 5137594{,}18/(1+0{,}15)^1 + 10522628{,}99/(1+0{,}15)^2 + 15907663{,}80/(1+0{,}15)^3 +$

$+ 19784888{,}87/(1+0{,}15)^4 = 4467473{,}20 + 7956619{,}27 + 10459547{,}17 + 11312074{,}42 = 34195714{,}06$ грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{5.21}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 338886,96 грн.

$PV = k_{інв} \cdot ЗВ$ = 2 · 338886,96 = 677773,92 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.22}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 34195714,06 грн;

PV – теперішня вартість початкових інвестицій, 677773,92 грн.

$E_{абс} = ПП - PV$ = 34195714,06 - 677773,92 = 33517940,14 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.23}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 33517940,14 грн;

PV – теперішня вартість початкових інвестицій, 677773,92 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_{в} = \sqrt[T_{ж}]{1 + \frac{E_{абс}}{PV}} - 1 = (1+33517940{,}14/677773{,}92)^{1/4} = 1{,}67.$$

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

$$\tau_{мін} = d + f, \tag{5.24}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d =0,12;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,45.

$\tau_{мін}$ = 0,12+0,45 = 0,57 < 1,67 свідчить про те, що внутрішня економічна дохідність інвестицій $E_{в}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія нейромережевого розпізнавання зображень тварин» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_{в}}, \tag{5.25}$$

де $E_{в}$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 1,67 = 0,60 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.5 Висновок до розділу 5

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія нейромережевого розпізнавання зображень тварин» становить 42,0 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий). При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,71 рази. Також термін окупності становить 0,60 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок. Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія нейромережевого розпізнавання зображень тварин».

ВИСНОВКИ

При виконанні магістерської кваліфікаційної роботи розв’язано задачу розробки інформаційної технології та програмного забезпечення розпізннання зображень тварин на основі згорткової нейронної мережі ConvNetSharp Network.

У першому розділі магістерської роботи було зроблено огляд методів розпізнавання зображень на сьогоднішній день. Проведено також огляд відомих програмних засобів для роботи з інтелектуальними обчисленнями у сфері комп’ютерного зору. В ході аналізу предметної області розглянуто основні методи розпізнавання зображень та як найбільш перспективний, було обрано нейромережевий метод.

У другому розділі магістерської кваліфікаційної роботи було обґрунтовано вибір типу нейронної мережі для задачі розпізнавання зображень – згорткова нейромережа ConvNetSharp. Була розроблена структура та проаналізована математична модель згорткової нейромережі, обгрунтовано вибір функції активації нейронів. Було розроблено структуру інформаційної технології нейромережевого розпізнавання зображень тварин.

У третьому розділі було обґрунтовано вибір середовища розробки, мови програмування та фреймворків, які будуть використовуватися при розробці програми та наведено їх основні переваги. У результаті аналізу було обрано середовище розробки Visual Studio Code, мову програмування C# та бібліотеку ConvNetSharp. Було розроблено алгоритм роботи програми та структуру програмного забезпечення розпізнавання зображень тварин. Було проведено реалізацію програмних модулів системи, включно з модулем тренування мережі, виведення графіку помилки навчання та класифікації зображень.

У четвертому розділі в результаті тестування програми було доведено її повну працездатність та відповідність поставленому завданню. Тестування нейронної мережі проходило на вибірці з 100 зображень, з яких 50 – були

зображення котів, а інші 50 – зображеннями інших тварин. Розроблена згорткова нейронна мережа ConvNetSharp Network досягла похибки в 6% (достовірність 94%) та тренувалась біля 24 години. А глибинна мережа переконань Accord.Net Network (аналог), досягла похибки у 17% (достовірність 83%), але потребує набагато менше часу на навчання – 8 годин. Таким чином, можна зробити висновок, що згорткова нейронна мережа ConvNetSharp Network має порівняно з глибинною мережею переконань Accord.Net Network збільшену на 11% (17-6=11) достовірність розпізнавання. Тобто мета роботи досягнута – достовірність розпізнавання підвищена.

У п`ятому розділі визначено, що рівень комерційного потенціалу розробки становить 42,0 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий). Також визначено, що науково-технічна розробка переважає існуючі аналоги приблизно в 1,71 рази. Термін окупності становить 0,60 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. В. О. Максименко, О. К. Колесницький «Інформаційна технологія нейромережевого розпізнавання зображень тварин», 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/mn2023/paper/viewFile/16829/14063.

2. Фисенко В.Т. / Компьютерная обработка и распознавание изображений / В.Т. Фисенко, Т.Ю. Фисенко – учеб. пособие – СПб: СПбГУ ИТМО, 2008. – 192 с.

3. Codeguida Нейронні мережі - шлях до глибинного навчання [Електронний ресурс]. – Режим доступу: https://codeguida.com/post/739 (Дата звернення: 03.05.2021)

4. Хайкин С. Нейронные сети: полный курс / С. Хайкин – Издание 2.: Пер. с англ. – М.: Издательский дом «Вильямс», 2006. - 1104 с.

5. 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-X.

6. О.В.Гороховацький, О.О.Передрій Багатошаровий персептрон як інструмент первинної кластеризації зображень [Електронний ресурс] / О.В.Гороховацький, О.О.Передрій. – Режим доступу: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/131626/04-Gorokhovatskiy.pdf?sequence=1 (Дата звернення: 03.05.2021)

7. Віктор Синєглазов, Олена Чумаченко Структурно-параметричний синтез згорткових нейронних мереж [Електронний ресурс] / Віктор Синєглазов, Олена Чумаченко. – Режим доступу: http://itcm.comp-sc.if.ua/2018/synehlazov.pdf (Дата звернення: 03.05.2021)

8. Habr Навчання згорткової нейронної мережі [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/348028/ (Дата звернення: 04.05.2021)

9. Любунь З. М. Основи теорії нейромереж: текст лекцій / З. М. Любунь. – Львів : Видавничий центр ЛНУ ім. Івана Франка, 2006.

10. NeuroHive Функції активації нейромережі [Електронний ресурс]. – Режим доступу: https://neurohive.io/ru/osnovy-data-science/activation-functions/ (Дата звернення: 10.05.2021)

11. CodeProject Класифікація зображень за допомогою нейронних мереж у .NET [Електронний ресурс]. – Режим доступу: https://www.codeproject.com/Articles/5160467/Image-Classification-Using-Neural-Networks-in-NET (Дата звернення: 10.01.2021)

12. Кулаков Д.М., Баєва Н.В. Інтегроване середовище розробки для функціональної мови Refal (рос) [Електронний ресурс] / Кулаков Д.М., Баєва Н.В. – Режим доступу: https://pt.slideshare.net/msucsai/refal (Дата звернення: 10.01.2021)

13. Visual Studio Лучшие в своем классе средства для разработчиков [Електронний ресурс] // Visual Studio – Режим доступу до ресурсу: https://visualstudio.microsoft.com/en/.

14. IPkey Що таке мови програмування [Електронний ресурс] / IPkey. – Режим доступу: http://ipkey.com.ua/uk/faq/925-programming-languages.html (Дата звернення: 10.01.2021)

15. Бубнов И. Лучшие IDE для разработки на C# Один очевидный вариант и несколько других. [Електронний ресурс] / Илья Бубнов // geekbrains. – 2018. – Режим доступу до ресурсу: https://geekbrains.ru/posts/ c_sharp_ides/.

16. Троелсен Э. C# и платформа .NET. Библиотека программиста. - СПб.: Питер, 2007. - 796 с.

17. НЕЙРОННЫЕ СЕТИ НА C#. [Електронний ресурс] – Режим доступу: http://www.cyberguru.ru/algorithms/algorithms-theory/algorithms-neural-networks-written-on-csharp.html

18. GitHub ConvNetSharp [Електронний ресурс] (англ) / GitHub – Режим доступу: https://github.com/cbovar/ConvNetSharp (Дата звернення: 10.05.2021)

19. Wikipedia Вільний інтерфейс [Електронний ресурс] (англ) / Wikipedia. – Режим доступу: https://en.wikipedia.org/wiki/Fluent_interface (Дата звернення: 10.05.2021)

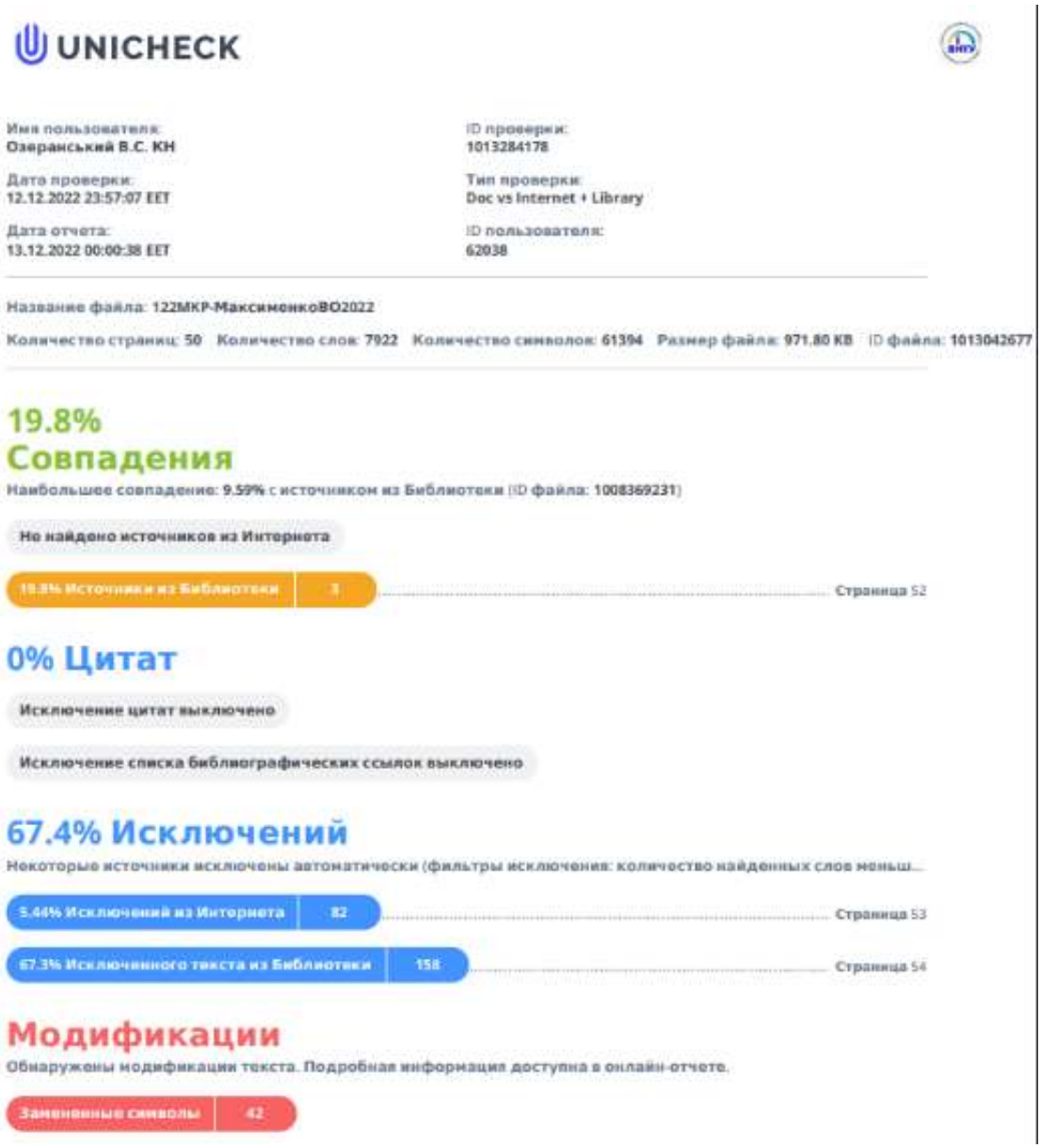
20. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

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

ДОДАТКИ

Додаток А (обов'язковий)

Результат перевірки на плагіат в онлайн-системі UNICHECK



UNICHECK

Имя пользователя:
Озаранський В.С. КН

ID проверки:
1013284178

Дата проверки:
12.12.2022 23:57:07 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
13.12.2022 00:00:38 EET

ID пользователя:
62038

Название файла: 122МКР-МаксименкоВО2022

Количество страниц: 50 Количество слов: 7922 Количество символов: 61394 Размер файла: 971.80 KB ID файла: 1013042677

19.8%
Совпадения

Наибольшее совпадение: 9.59% с источником из Библиотеки (ID файла: 1008369231)

Не найдено источников из Интернета

19.8% Источники из Библиотеки 3 ..... Страница 52

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

67.4% Исключений

Некоторые источники исключены автоматически (фильтры исключения: количество найденных слов меньш...

5.44% Исключений из Интернета 82 ..... Страница 53

67.3% Исключенного текста из Библиотеки 158 ..... Страница 54

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Замененные символы 42

Додаток Б (обов'язковий)

Лістинг програми

ConvNetSharpNetwork.cs

```
using ConvNetSharp.Core;
using ConvNetSharp.Core.Layers;
using ConvNetSharp.Core.Layers.Double;
using ConvNetSharp.Core.Training;
using ConvNetSharp.Volume;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ConvNetSharp.Core.Serialization;
using CatImageRecognizer.Models;
using System.IO;

namespace CatImageRecognizer.NeuralNetworks
{
    public class ConvNetSharpNetwork : INeuralNetwork
    {
        private Net<double> network;
        TrainerBase<double> trainer;

        private int batchSubSize = 50;
        private int maxIterationsOnEachPage = 500;
        private double stopPageTrainingErrorRate = 0.02;
        private double stopIterationTrainingErrorRate = 1;
        public ConvNetSharpNetwork()
        {
            network = new Net<double>();

            network.AddLayer(new InputLayer(50, 52, 1));
            network.AddLayer(new ConvLayer(3, 3, 8) { Stride = 1, Pad = 2 });
            network.AddLayer(new ReLuLayer());
            network.AddLayer(new PoolLayer(3, 3) { Stride = 2 });
            network.AddLayer(new ConvLayer(3, 3, 16) { Stride = 1, Pad = 2 });
            network.AddLayer(new ReLuLayer());
            network.AddLayer(new PoolLayer(3, 3) { Stride = 2 });
            network.AddLayer(new ConvLayer(3, 3, 32) { Stride = 1, Pad = 2 });
            network.AddLayer(new FullyConnLayer(20));
            network.AddLayer(new FullyConnLayer(50));
            network.AddLayer(new FullyConnLayer(2));
            network.AddLayer(new SoftmaxLayer(2));

            trainer = GetTrainerForNetwork(network);
        }

        private TrainerBase<double> GetTrainerForNetwork(Net<double> net)
        {
            return new SgdTrainer<double>(net)
            {
                LearningRate = 0.003
            };
        }

        private double[] Layout2DArray(double[][] inputs, int totalLength)
```

```
            {
                var dataLayoutOutInputs = new double[totalLength];
                foreach (int d in Enumerable.Range(0, inputs.Length))
                {
                    foreach (int i in Enumerable.Range(0, inputs[d].Length))
                    {
                        dataLayoutOutInputs[(inputs[d].Length * d) + i] =
inputs[d][i];
                    }
                }
                return dataLayoutOutInputs;
            }

            public bool ShouldStopTraning { get; set; } = false;

            public void StopBatchTraining()
            {
                this.ShouldStopTraning = true;
            }

            private (Volume<double>, Volume<double>)
GetVolumeDataSetsFromArrays(double[][] batchInputs, double[][] batchOutputs)
            {
                var dataLayedoutInputs = Layout2DArray(batchInputs,
batchInputs.Length * 50 * 52);
                var dataLayedoutOutputs = Layout2DArray(batchOutputs,
batchOutputs.Length * 2);

                trainer.BatchSize = batchInputs.Length;
                Volume<double> inputs =
BuilderInstance<double>.Volume.From(dataLayedoutInputs, new Shape(50, 52, 1,
batchInputs.Length));
                Volume<double> outputs =
BuilderInstance<double>.Volume.From(dataLayedoutOutputs, new Shape(1, 1, 2,
batchOutputs.Length));

                return (inputs, outputs);
            }

            public void BatchTrain(double[][] batchInputs, double[][]
batchOutputs, int iterations, Action<double, int, string> progressCallback)
            {
                trainer.BatchSize = batchInputs.Length;
                foreach (int currentIteration in Enumerable.Range(1, iterations))
                {
                    Randomizer.Shuffle(batchInputs, batchOutputs);
                    (Volume<double> inputs, Volume<double> outputs) =
GetVolumeDataSetsFromArrays(batchInputs, batchOutputs);
                    trainer.Train(inputs, outputs);
                    var error = network.GetCostLoss(inputs, outputs);
                    if (progressCallback != null)
                    {
                        progressCallback(error, currentIteration,
"ПІдконтрольний");
                    }
                    inputs.Dispose();
                    outputs.Dispose();
                    if(this.ShouldStopTraning)
                    {
                        this.ShouldStopTraning = false;
                        break;
```

```
                    }
                }
            }

            public double[] GenerateOutput(double[] inputs)
            {
                Volume<double>                  inputImageData                  =
BuilderInstance<double>.Volume.From(inputs, new Shape(50, 52, 1));
                Volume<double>              calculatedPrediction              =
network.Forward(inputImageData);
                inputImageData.Dispose();
                return calculatedPrediction.ToArray();
            }

            public int GetInputLength()
            {
                return 2600;
            }

            public void LoadNetworkFromFile(string filePath)
            {
                var networkJSON = File.ReadAllText(filePath);
                network = SerializationExtensions.FromJson<double>(networkJSON);
                trainer = GetTrainerForNetwork(network);
            }

            public void SaveNetwork(string filePath)
            {
                var networkJSON = network.ToJson();
                FileHelper.WriteTextAsync(filePath,
Encoding.ASCII.GetBytes(networkJSON)).RunSynchronously();
            }

            public string GetNetworkName()
            {
                return "ConvNetSharp";
            }
            public int GetBatchSubSize()
            {
                return batchSubSize;
            }

            public int GetMaxIterationsOnEachPage()
            {
                return maxIterationsOnEachPage;
            }

            public double GetStopPageTrainingErrorRate()
            {
                return stopPageTrainingErrorRate;
            }

            public double GetStopIterationTrainingErrorRate()
            {
                return stopIterationTrainingErrorRate;
            }
            public string GetModelExtension()
            {
                return "convmodel";
            }
        }
```

```
}
```

Converters.cs

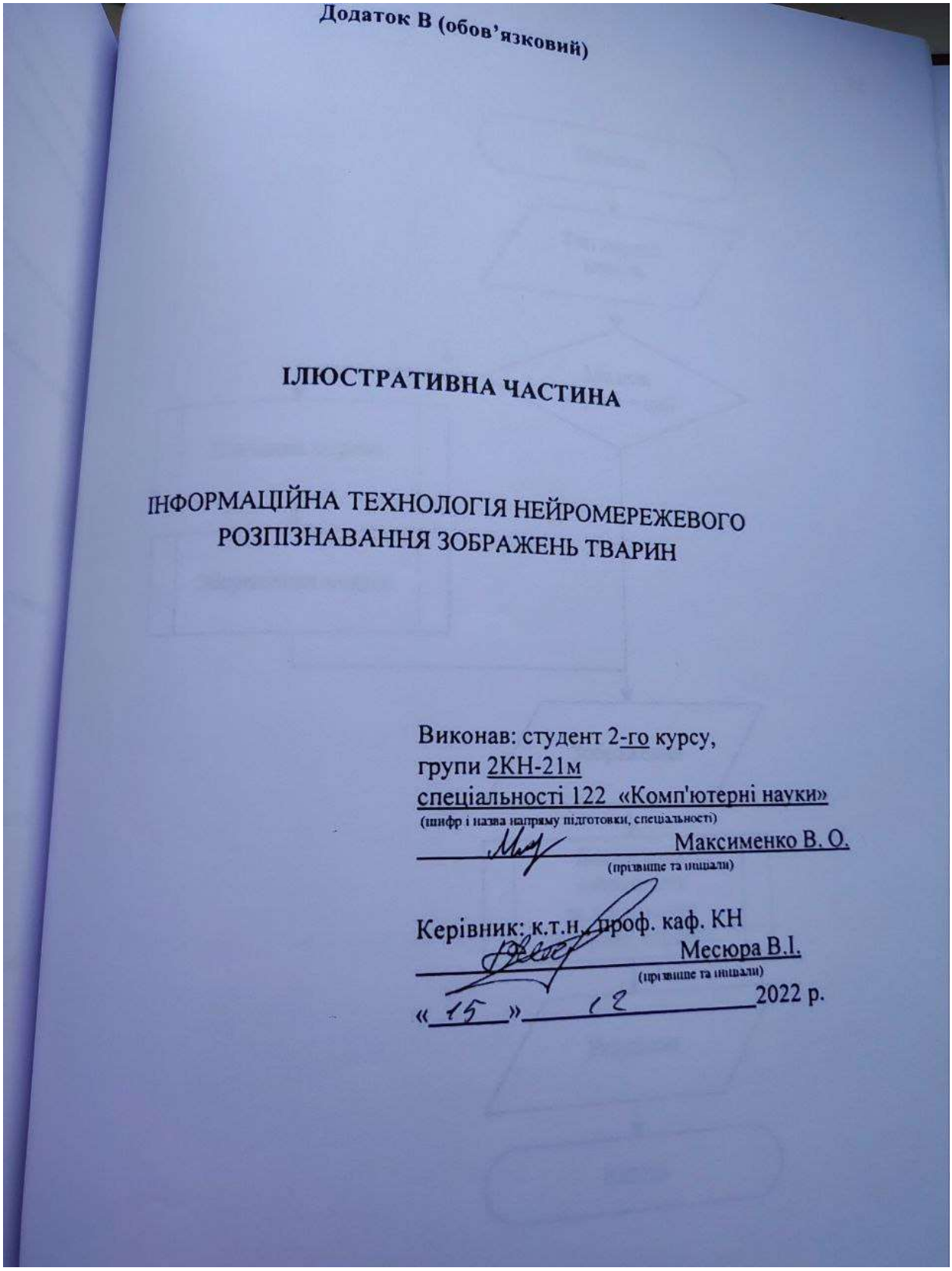
```
using CatImageRecognizer.Models;
using System;
using System.Collections.Generic;
using System.Drawing.Imaging;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Data;
using System.Windows.Media.Imaging;

namespace CatImageRecognizer
{
    public class CatImageTypeToTextConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            var catImageType = (ImageType)value;
            return catImageType == ImageType.CAT ? "Cat" : "No cat";
        }

        public object ConvertBack(object value, Type targetType, object
parameter, CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }

    public class MultipleBooleanANDConverter : IMultiValueConverter
    {
        public object Convert(object[] values, Type targetType, object
parameter, CultureInfo culture)
        {
            bool finalResult = true;
            foreach(bool value in values)
            {
                finalResult = finalResult && value;
            }
            var invert = parameter != null ? (bool)parameter : false;
            return invert ? !finalResult : finalResult;
        }

        public object[] ConvertBack(object value, Type[] targetTypes, object
parameter, CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }

    public class MultipleBooleanORConverter : IMultiValueConverter
    {
        public object Convert(object[] values, Type targetType, object
parameter, CultureInfo culture)
```

```
            {
                bool finalResult = false;
                foreach (bool value in values)
                {
                    finalResult = finalResult || value;
                }
                var invert = parameter != null ? (bool)parameter : false;
                return invert ? !finalResult : finalResult;
            }

            public object[] ConvertBack(object value, Type[] targetTypes, object
parameter, CultureInfo culture)
            {
                throw new NotImplementedException();
            }
        }

        public class BoolToVisibilityConverter : IValueConverter
        {
            public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
            {
                var reverse = false;
                if(parameter != null)
                {
                    reverse = (bool)parameter;
                }

                if((bool)value == true)
                {
                    return reverse ? Visibility.Hidden : Visibility.Visible;
                }
                return reverse ? Visibility.Visible : Visibility.Hidden;
            }

            public  object  ConvertBack(object  value,  Type  targetType,  object
parameter, CultureInfo culture)
            {
                return true;
            }
        }

        public class BoolToVisibilityCollapsedConverter : IValueConverter
        {
            public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
            {
                var reverse = false;
                if (parameter != null)
                {
                    reverse = (bool)parameter;
                }

                if ((bool)value == true)
                {
                    return reverse ? Visibility.Collapsed : Visibility.Visible;
                }
                return reverse ? Visibility.Visible : Visibility.Collapsed;
            }
```

```
            public object ConvertBack(object value, Type targetType, object
parameter, CultureInfo culture)
            {
                return true;
            }
        }

        public class EmptyListToVisibilityConverter : IValueConverter
        {
            public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
            {
                var reverse = false;
                if (parameter != null)
                {
                    reverse = (bool)parameter;
                }

                if ((int)value > 0)
                {
                    return reverse ? Visibility.Hidden : Visibility.Visible;
                }
                return reverse ? Visibility.Visible : Visibility.Hidden;
            }

            public object ConvertBack(object value, Type targetType, object
parameter, CultureInfo culture)
            {
                return true;
            }
        }

        public class EmgCVImageToBitmapImage : IValueConverter
        {
            public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
            {
                var emguCVImage = (Emgu.CV.IImage)value;
                if(emguCVImage == null)
                {
                    return new BitmapImage();
                }
                return                              GetJPEGEncodedImage(emguCVImage,
GetJPEGEncoderParameters());
            }
            private ImageCodecInfo GetEncoder(ImageFormat format)
            {
                ImageCodecInfo[] codecs = ImageCodecInfo.GetImageDecoders();
                foreach (ImageCodecInfo codec in codecs)
                {
                    if (codec.FormatID == format.Guid)
                    {
                        return codec;
                    }
                }
                return null;
            }

            private EncoderParameters GetJPEGEncoderParameters()
            {
```

```
                System.Drawing.Imaging.Encoder qualityEncoder =
System.Drawing.Imaging.Encoder.Quality;
                EncoderParameters jpegEncoderParameters = new
EncoderParameters(1);
                EncoderParameter qualityEncoderParamter = new
EncoderParameter(qualityEncoder, 50L);
                jpegEncoderParameters.Param[0] = qualityEncoderParamter;
                return jpegEncoderParameters;
            }

            private BitmapImage GetJPEGEncodedImage(Emgu.CV.IImage rawImage,
EncoderParameters encoderParameters)
            {
                MemoryStream imageMemoryStream = new MemoryStream();
                ((System.Drawing.Bitmap)rawImage.Bitmap).Save(imageMemoryStream,
GetEncoder(ImageFormat.Jpeg), GetJPEGEncoderParameters());
                BitmapImage image = new BitmapImage();
                image.BeginInit();
                imageMemoryStream.Seek(0, SeekOrigin.Begin);
                image.StreamSource = imageMemoryStream;
                image.EndInit();
                return image;
            }

            public object ConvertBack(object value, Type targetType, object
parameter, CultureInfo culture)
            {
                return null;
            }
        }

        public class BooleanInverterConverter : IValueConverter
        {
            public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
            {
                return !(bool)value;
            }

            public object ConvertBack(object value, Type targetType, object
parameter, CultureInfo culture)
            {
                return !(bool)value;
            }
        }

    }
```

Додаток В (обов'язковий)



Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ НЕЙРОМЕРЕЖЕВОГО РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН

Виконав: студент 2-го курсу,
групи 2КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Максименко В. О.
(прізвище та ініціали)

Керівник: к.т.н., проф. каф. КН
Месюра В.І.
(прізвище та ініціали)
«15» 12 2022 р.

Рисунок В.1 – Граф-схема загального алгоритму роботи програми нейромережевого розпізнавання зображень тварин

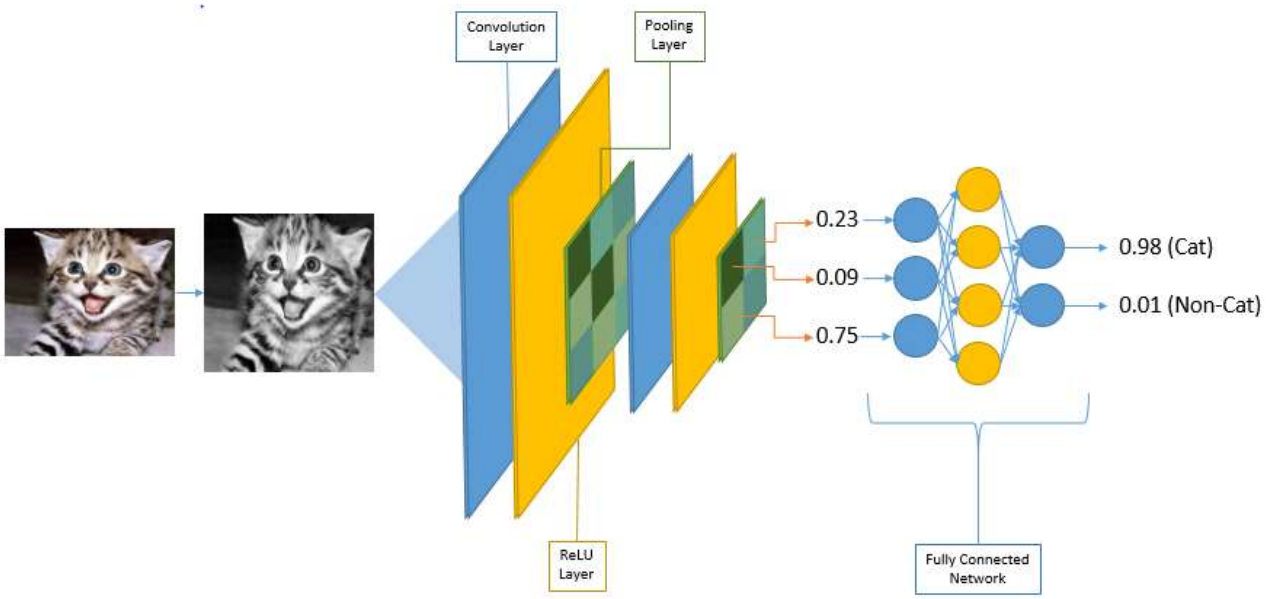


Рисунок В.2 – Структура згорткової нейронної мережі

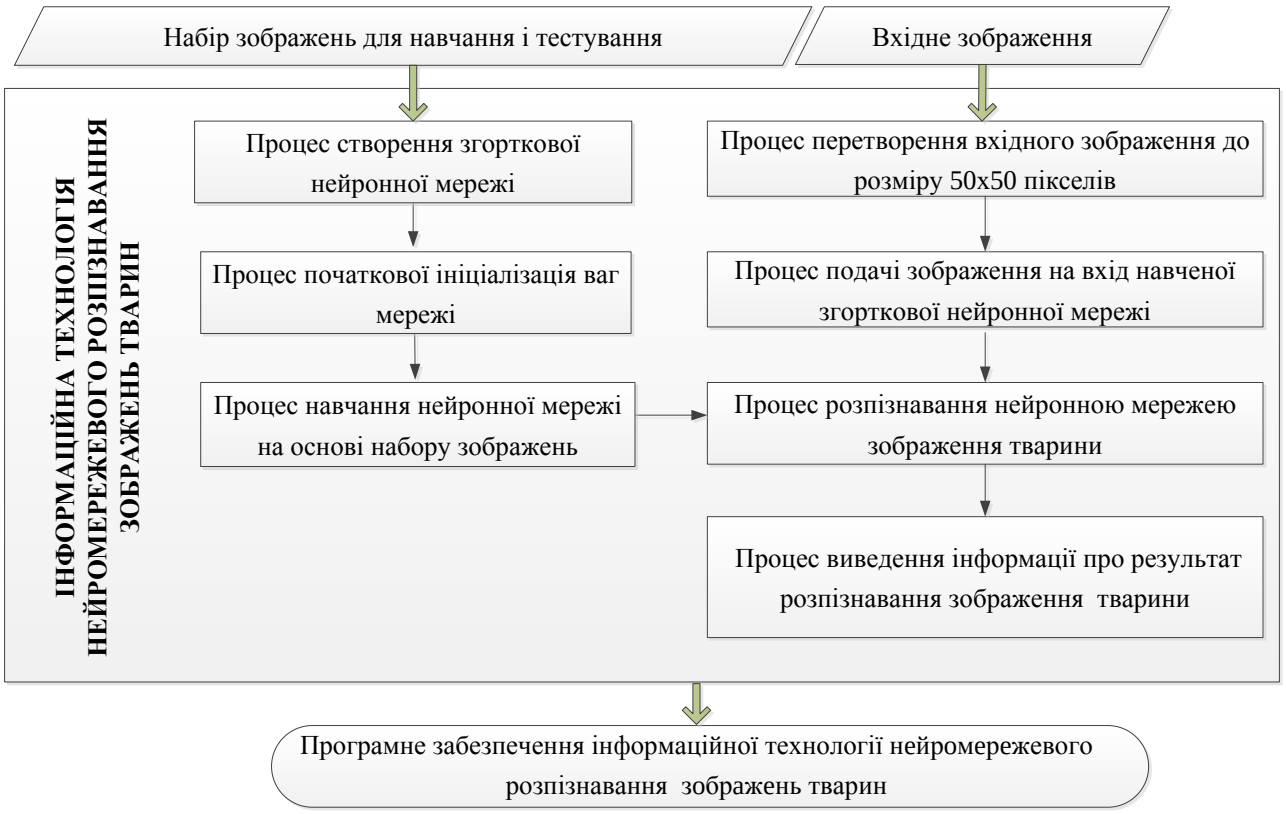


Рисунок В.3 - Структура інформаційної технології нейромережевого розпізнавання зображень тварин

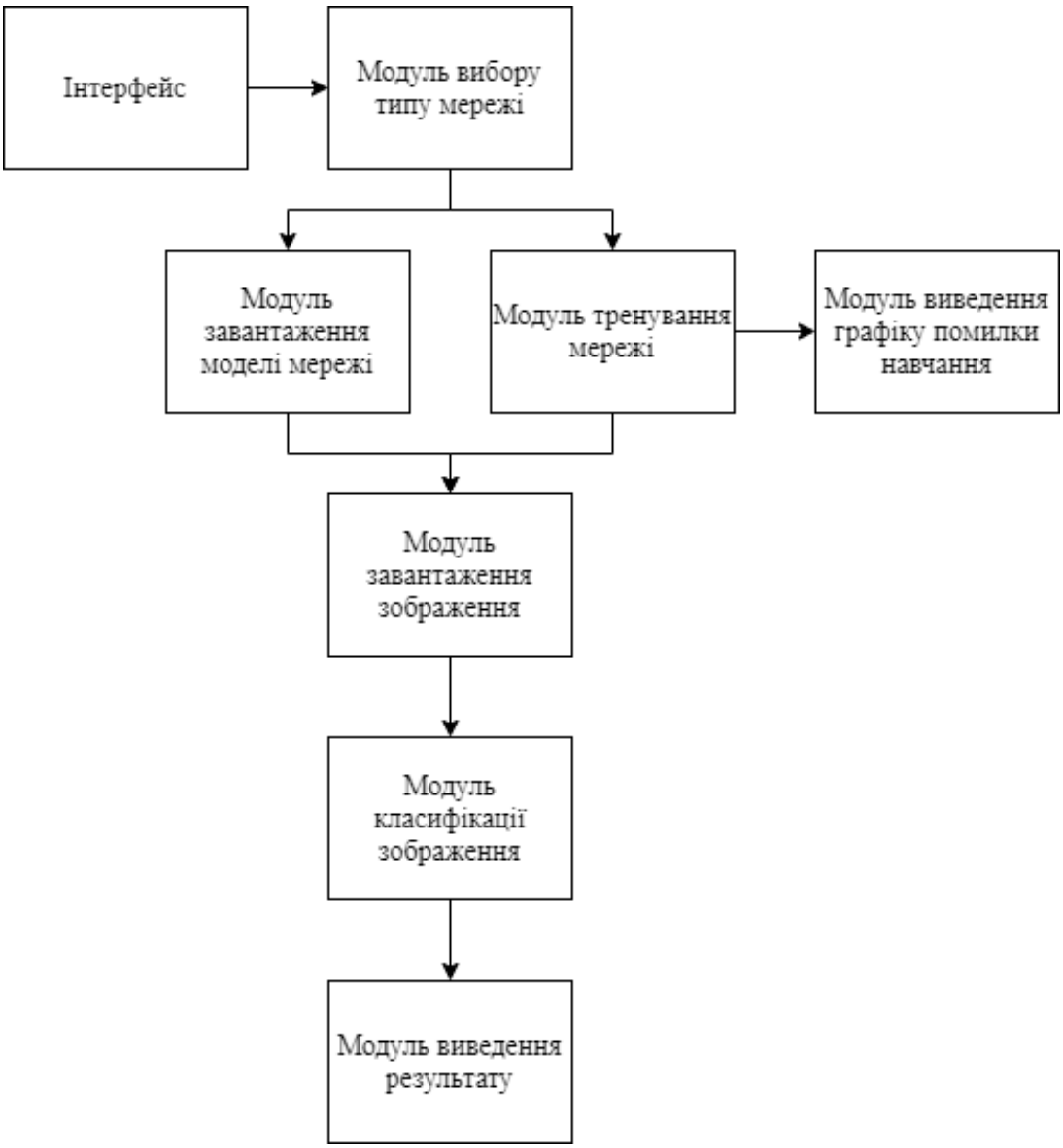


Рисунок В.4 – Структура програмного забезпечення нейромережевого розпізнавання зображень тварин

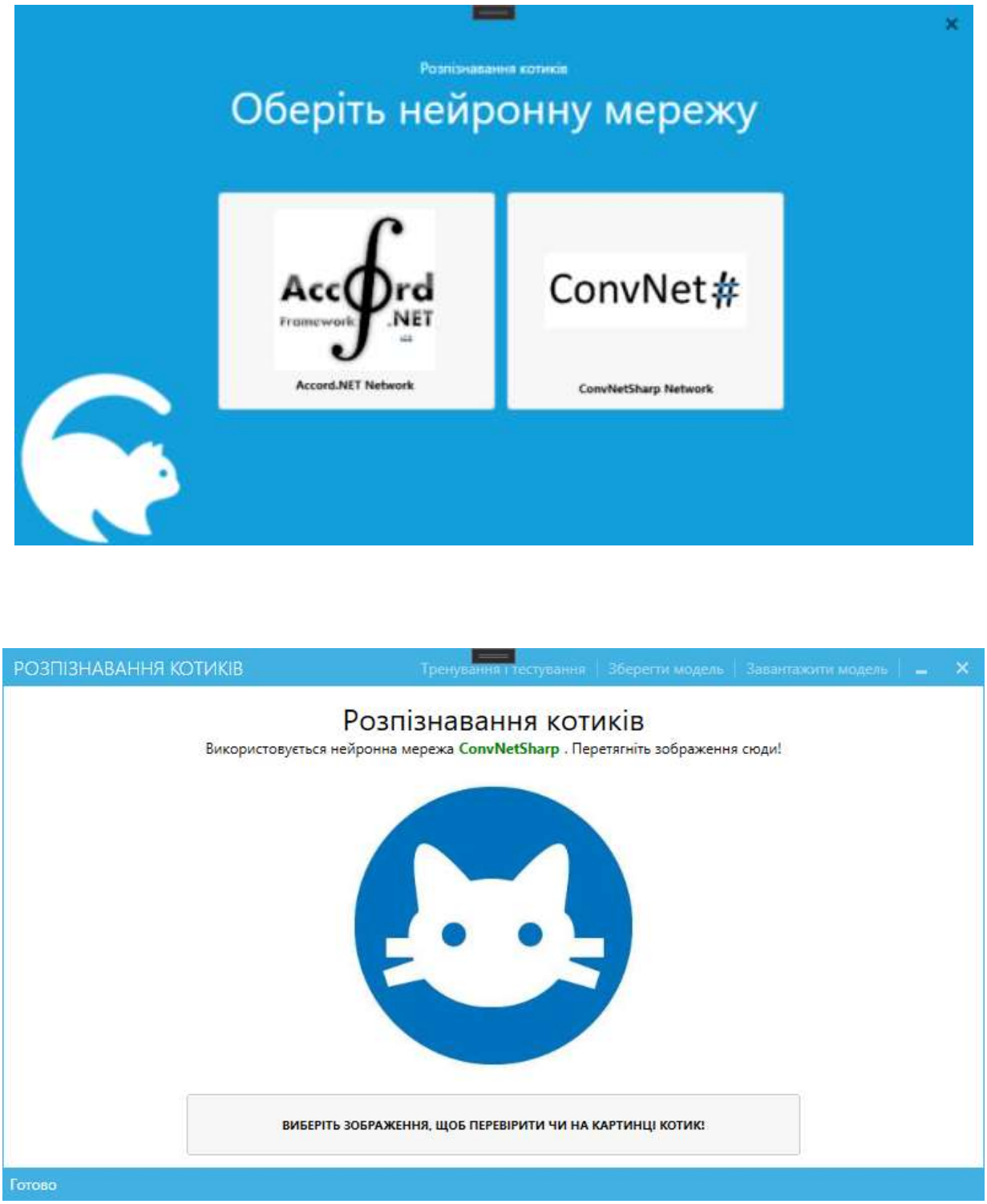


Рисунок В.5 – Стартові вікна програми нейромережевого розпізнавання зображень тварин

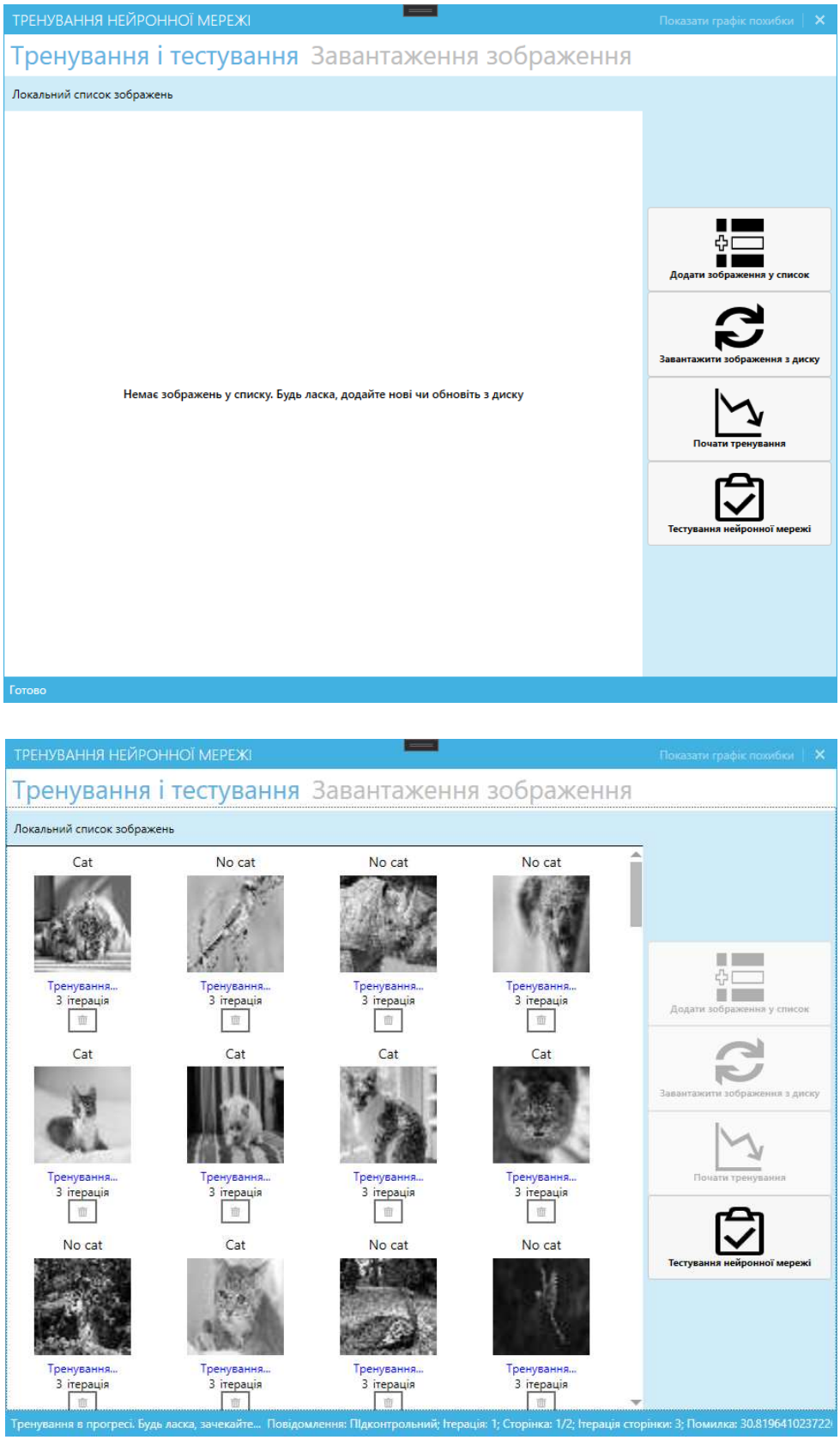


Рисунок В.6 – Результати роботи програми нейромережевого розпізнавання зображень тварин

Додаток Г (довідниковий)

Інструкція користувача

Для початку роботи з програмою, її необхідно запустити. Після запуску відкриється вікно вибору типу мережі (рис.Г.1).

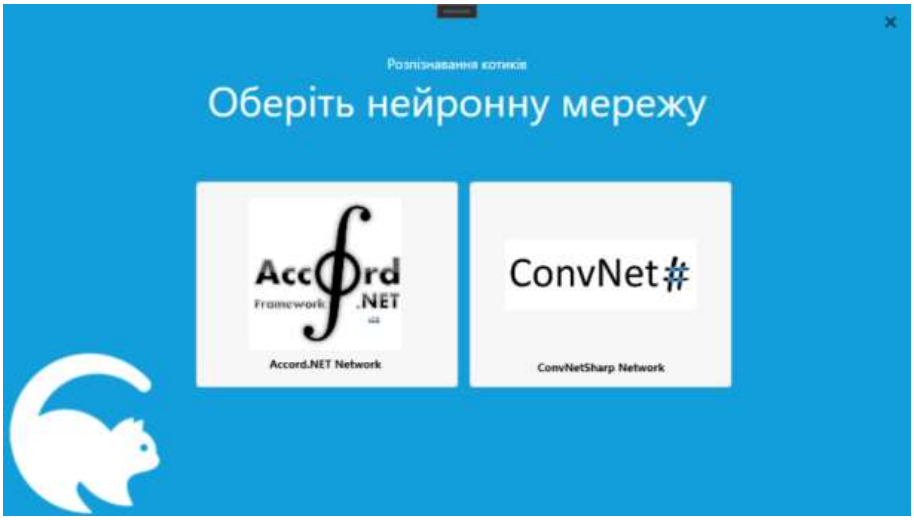


Рисунок Г.1 – Вікно вибору типу мережі

Після вибору типу мережі, відкриється вікно, де можна завантажити навчену модель для розпізнавання (1) або перейти до тренування нової моделі (2) (рис. Г.2).

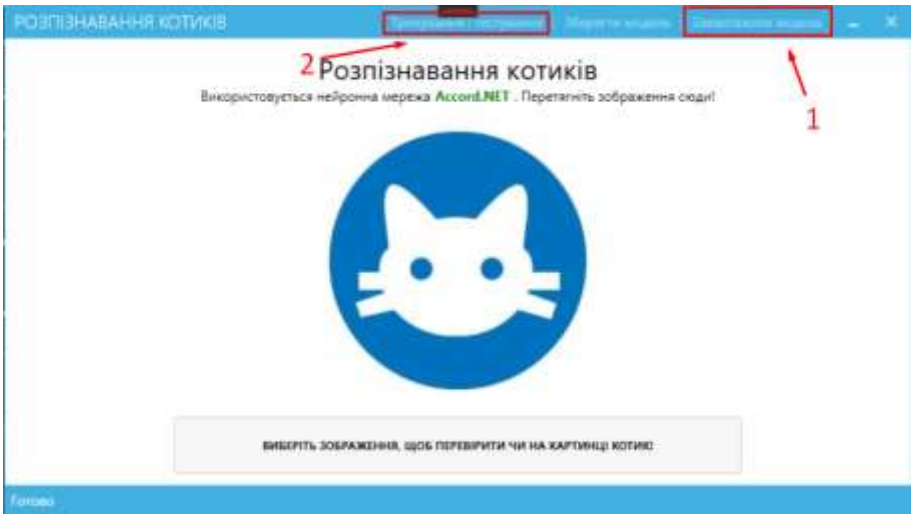


Рисунок Г.2 – Вікно розпізнавання

Якщо у користувача вже є навчена модель, то, після її завантаження, він може перейти до завантаження зображення для початку процесу розпізнавання. Для цього необхідно на вікні розпізнавання натиснути кнопку «Виберіть зображення, щоб перевірити чи на картинці котик!» (рис. Г.3) та обрати зображення, яке бажаєте розпізнати. Після цього зображення з’явиться у вікні розпізнавання та автоматично розпочнеться процес класифікації (рис. Г.4).

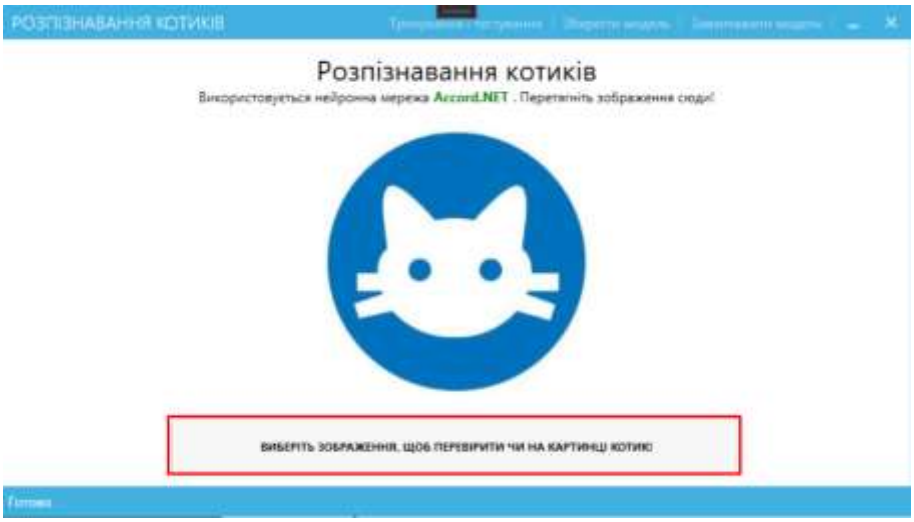


Рисунок Г.3 – Кнопка для завантаження зображення

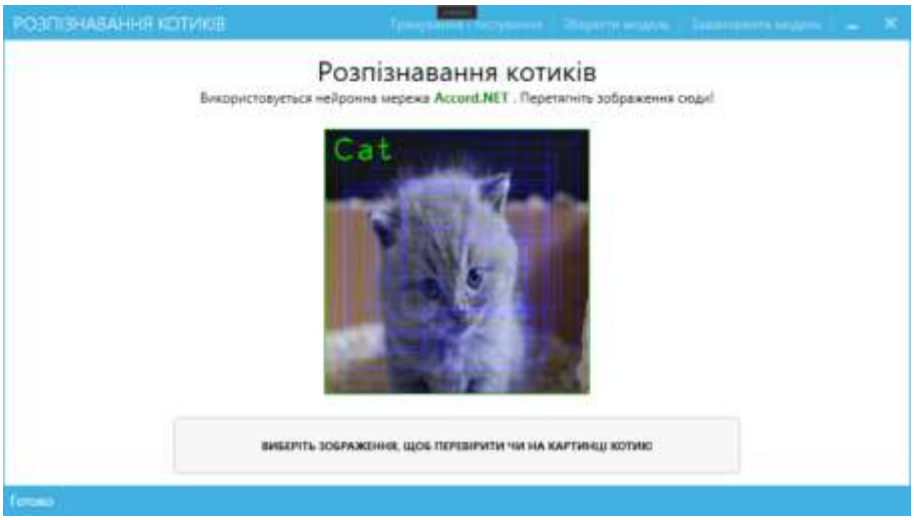


Рисунок Г.4 – Процес розпізнавання зображення

Якщо у користувача немає натренованої моделі, то він може перейти до її тренування, натиснувши кнопку (2), зображену на рисунку Г.2. Після цього

відкриється вікно, в яке необхідно завантажити зображення (1), вказуючи при цьому чи на ньому кіт або завантажити з пам’яті (2) ті зображення, які були раніше і почати тренування мережі (3) (рис. Г.5).

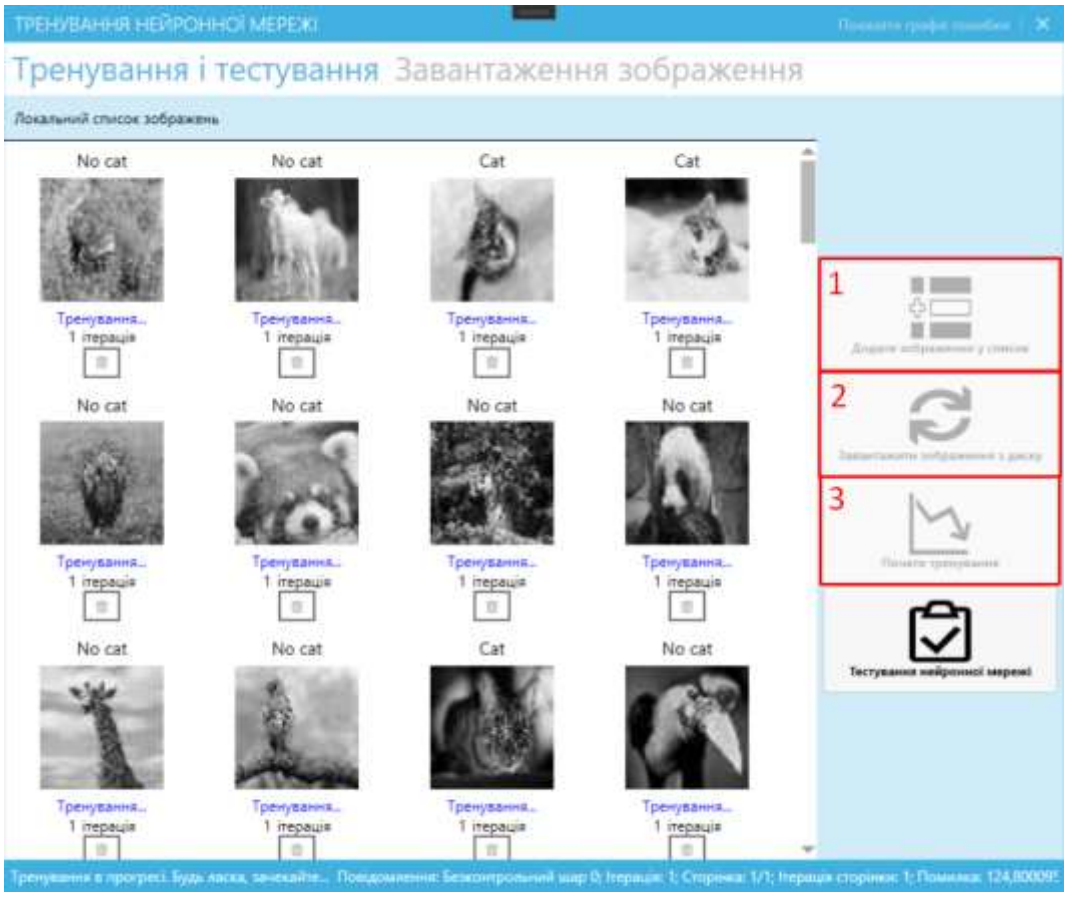


Рисунок Г.5 – Вікно тренування

Коли тренування буде закінчено, необхідно протестувати наскільки модель правильно розпізнає зображення з тестової вибірки. Для цього необхідно натиснути кнопку «Тестування нейронної мережі» та переглянути результат внизу вікна (рис. Г.6).

Рисунок Г.6 – Тестування навченої нейронної мережі

Якщо результати тестування задовольняють користувача, то він може зберегти натреновану модель. Для цього йому потрібно закрити вікно тренування та на вікні розпізнавання натиснути кнопку «Зберегти модель» (рис. Г.7).

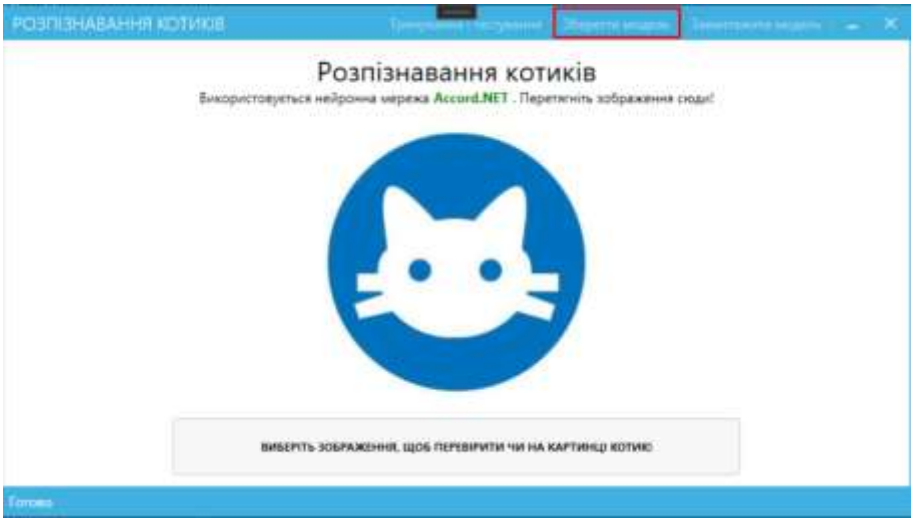


Рисунок Г.7 – Кнопка збереження моделі