Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Удосконалена система нейромережевого розпізнавання зображень»

Виконав: студент 2 курсу, групи 2КІ—21м
напряму підготовки (спеціальності)
123 — «Комп'ютерна інженерія»
_____Кучевський О.В.

Керівник: д.т.н., зав. каф.ОТ
_____ Азаров О.Д.
«19» 12 2022 р.

Опонент: д.т.н., проф. каф. МБІС.,
_____ Яремчук Ю.Є.
«21» 12 2022 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«22» 12 2022 р.

Вінниця ВНТУ – 2022

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо—кваліфікаційний рівень: магістр
Спеціальність 123 — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри
обчислювальної техніки
_____ проф., д.т.н. О.Д. Азаров
«15» 09 2022 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

Кучевському Олексію Володимировичу

1 Тема роботи «Удосконалена система нейромережевого розпізнавання зображень» керівник Азаров Олексій Дмитрович д.т.н., зав. каф. ОТ, затверджені наказом №205 вищого навчального закладу від 15.09.2022 р.

2 Строк подання студентом роботи 18 листопада 2022 року.

3 Вихідні дані до роботи :

Вхідні дані — формат вхідних зображень — jpg, кількість основних нейронів в мережі — не менше 1000, кількість прихованих шарів — не менше 2 , розмір зображення на вході нейромережі — 50x50, обсяг навчальної вибірки — не менше 500, обсяг тестової вибірки — не менше 100, середовище програмування — об'єктно—орієнтоване.

2 Зміст розрахунково—пояснювальної записки (перелік питань, які потрібно розробити):

Вступ, постановка задачі, аналіз предметної області розпізнавання зображень тварин, обґрунтування вибору типу нейронної мережі для розв'язання задачі розпізнавання зображень тварин, проектування основного компонента

програми розпізнавання зображень тварин, програмна реалізація модуля для розпізнавання зображень тварин, тестування та аналіз результатів роботи програми розпізнавання зображень тварин, висновки, перелік використаних джерел, додатки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

схема алгоритму роботи програми, структура нейронної мережі, результати роботи програми.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1—3	Азаров О.Д., доц. каф. КН		
4	Небава Микола Іванович, к.е.н., професор каф. ЕПВМ		

7 Дата видачі завдання 07.09.2022 р.

8 Календарний наведенио в таблиці 2.

Таблиця 2 — Календарний план

№	Назва етапів МКР	Строк виконання	Підпис
1	Постановка задачі	01.09.22	
2	Пошук літературних джерел по темі магістерської роботи.	09.09—18.09.22	
3	Пошук та підбір актуальних нейромереж.	19.09—01.10.22	
4	Вивчення способів реалізації нейромереж.	02.10—18.10.22	
6	Розробка структури згорткової нейромережі.	17.11—30.11.22	
10	Оформлення пояснювальної записки та ілюстративного матеріалу	01.12—05.12.22	
11	Аналіз виконання роботи, висновки, додатки	05.12—14.12.22	
12	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	17.12.22	

Студент _____ Кучевський О. В.

Керівник роботи _____ Азаров О. Д.

АНОТАЦІЯ

УДК 004.42

Кучевський О.В. Удосконалена система нейромережевого розпізнавання зображень. Магістерська кваліфікаційна робота за спеціальністю 123 — комп'ютерна інженерія, освітня програма—комп'ютерна інженерія. Вінниця: вид—у ВНТУ, 2022. 99с.

На укр.мові. Бібліогр.: 45 назв; рис.: 22; табл. 09.

У результаті виконання магістерська кваліфікаційної роботи було сформульовано постановку задачі розпізнавання зображень, під час якої досліджено різновиди нейронних мереж та обрано для вирішення поставленої задачі глибинну мережу переконань, а також зроблено огляд існуючих аналогів розроблюваної системи. Після цього було проаналізовано предметну область та досліджено математичну модель обраної нейронної мережі.

Згідно розробленої моделі, схема алгоритму програмно реалізована в додатку. Програмне забезпечення розроблено на мові програмування C# в середовищі Visual Studio з використанням глибинної мережі переконань Accord.Net Network. Розроблене додаток було протестовано.

Ключові слова: розпізнавання, нейромережа, розпізнавання об’єктів.

ABSTRACT

UDC 004.42

Kuchevsky O.V. An improved neural network image recognition system. Master's qualification work on specialty 123 — computer engineering, educational program — computer engineering. Vinnytsia: VNTU edition, 2022. 99p.

In the Ukrainian language. Bibliography: 45 titles; Fig.: 22; table 09.

As a result of completing the master's qualification work, the formulation of the problem of image recognition was formulated, during which the varieties of neural networks were investigated and a deep network of beliefs was chosen to solve the problem, as well as an overview of existing analogues of the developed system was made. After that, the subject area was analyzed and the mathematical model of the selected neural network was investigated.

According to the developed model, the scheme of the algorithm is programmatically implemented in the application. The software is developed in the C# programming language in the Visual Studio environment using the Accord.Net Network deep belief network. The developed application was tested.

Keywords: recognition, neural network, object recognition.

ЗМІСТ

					08-23.МКР.023.00.000 ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розробив		Кучевський О.В			Удосконалена система нейромережевого розпізнавання зображень Пояснювальна записка	Літ.	Аркуш 6 / Аркушів 84
Керівник		Азаров О.Д					
Рецензент		Яремчук Ю . Є				ВНТУ, гр. 2КІ-21м	
Н.контр.		Швець С.І.					
Затвердж.		Азаров О.Д					

ВСТУП

Актуальність теми полягає в тому, що в наш час такі речі як класифікація і машинне розпізнавання, групування образів, їх опис — важливі завдання у великій кількості інженерних і наукових областей, таких як біологія, фізіологія, медицина, маркетинг, комп'ютерний зір, штучний інтелект. Введемо поняття образу. Образ це протилежність хаосу; це певна сутність, якої може бути дано ім'я.

Однією з найважчих завдань розпізнавання образів є задача розпізнавання (класифікації) зображень. Це завдання виникає в таких областях як розпізнавання рукописного тексту, дорожніх знаків, номерів автомобілів, стерео і мультизору. Відмінною особливістю даного завдання є величезна розмірність вхідного простору — що веде до ускладнення розпізнавальних і обчислювальних процесів. Більшість підходів до розпізнавання зображень пропускають вхідні дані через фільтр, що проектує вхідний вектор на простір істотно меншої розмірності, після чого розбивають проміжні вектори на класи за допомогою стандартних систем розпізнавання.

Завдання розпізнавання зображень тварин відноситься до завдань комп'ютерного зору. Серед завдань комп'ютерного зору можна виділити наступні:

— класифікація найбільш відома проблема комп'ютерного зору. Вона полягає у віднесенні зображення до однієї з множини категорій.

— локалізація визначає місце розташування одного об'єкта на зображенні. Вона може комбінуватися з класифікацією для визначення місця розташування об'єкта та класифікації його.

— виявлення об'єкта включає в себе завдання класифікації і локалізації декількох об'єктів на зображенні водночас.

Завдання розпізнавання зображень тварин представляє собою завдання виявлення об'єкта, так як необхідно як виявити об'єкт, так і правильно його

класифікувати як об'єкт, при тому, що шуканих об'єктів на зображенні може бути кілька.

Магістерська робота присвячена розпізнаванню образів, а саме розпізнавання зображень тварин. Розпізнавання образів вирішує задачу виділення істотних ознак та їх віднесення до певного класу, що характеризують даний образ, із загальної маси даних. У даній роботі нейронна мережа використовуються для побудови програмних засобів для розпізнавання зображень тварин.

Мета і завдання досліджень полягає в підвищенні достовірності розпізнавання зображень тварин програмними засобами за рахунок застосування згорткових нейронних мереж.

Для досягнення мети розробки необхідно виконати такі задачі:

— проаналізувати відомі методи розпізнавання зображень тварин та обрати напрямок досліджень;

— обгрунтувати вибір архітектури нейромережі;

— проаналізувати модель розпізнавання зображень тварин на основі глибинної нейронної мережі переконань;

— розробити структуру програмного модуля;

— розробити алгоритм роботи програмного модуля розпізнавання зображень тварин,

— розробити програмне забезпечення розпізнавання зображень тварин на основі Accord.Net;

— провести тестування програмного модуля розпізнавання зображень тварин на основі Accord.Net.

Об’єктом дослідження є процес розпізнавання зображень тварин з використанням згорткових нейронних мереж.

Предмет дослідження — інформаційна технологія та програмні засоби розпізнавання зображень тварин з використанням згорткових нейронних мереж та достовірність їх роботи.

Для досягнення поставленої в роботі мети використовуються такі **методи**

дослідження:

— системний аналіз;

— розпізнавання образів;

— теорії штучних нейронних мереж,

— методи математичної статистики

— об’єктно—орієнтоване програмування.

Наукова новизна полягпє в подальшоику розвитку інформаційної технологія розпізнавання зображень тварин, яка відрізняється використанням глибиної мережі переконань, що дозволило підвищити достовірність 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 процесу розпізнавання зображень тварин, зокрема:

— розроблено алгоритм роботи програмного забезпечення розпізнавання зображень тварин на основі згорткової штучної нейронної мережі;

— розроблено програмні засоби для розпізнавання зображень тварин на основі згорткової штучної нейронної мережі;

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.

Апробація результатів роботи наведені у магістерській кваліфікаційній роботі, отримані самостійно. У працях, написаних у співавторстві, здобувачу належать: аналіз процесу розпізнавання зображень тварин на основі згорткової штучної нейронної мережі та методів підвищення достовірності [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН

1.1 Класифікація існуючих методів розпізнавання зображень тварин

Основна ідея в завданні розпізнавання зображень тварин — виділення ознак, відповідних шуканому об'єкту. Це дозволяє виявляти об'єкт не цілком, що викликає складнощі у зв'язку з високою мінливістю об'єкта, а виявляти об'єкт за характерними йому ознаками. У комп'ютерному зорі і обробці зображень, ознака — це певна структура в даних зображення, яка може бути представлена по—різному. Наприклад, колір конкретної області на зображенні може бути представлений у вигляді значення середнього кольору в області (три скаляра) або гістограми кольору (три функції).

У задачі розпізнавання об'єктів існують різні підходи до виділення ознак і їх подальше використання. Основні існуючі методи і алгоритми виявлення об'єктів можна класифікувати наступним чином [2]:

— метод локальних бінарних шаблонів;

— гістограма орієнтованих градієнтів;

— згорткові нейронні мережі.

1.1.1 Метод локальних бінарних ознак.

Локальні бінарні шаблони вперше запропоновані в 1996 році для аналізу текстури напівтонових зображень [3]. Локальні бінарні шаблони — це певний вид ознаки, що представляє собою оператор. Локальний бінарний шаблон являє собою опис околиці пікселя зображення в двійковому поданні.

Базовий оператор локального бінарного шаблону, який застосовується до пікселя зображення, використовує вісім пікселів околиці, приймаючи значення інтенсивності центрального пікселя в якості порога. Пікселі зі значенням інтенсивності більшим чи рівним значенню інтенсивності центрального пікселя приймають значення рівні «1», інші приймають значення рівні «0». Таким чином, результатом застосування базового оператора локального бінарного

шаблону до пікселя зображення є восьмирозрядний бінарний код, який описує околицю цього пікселя. Базовий оператор зображений на рисунку 1.1.

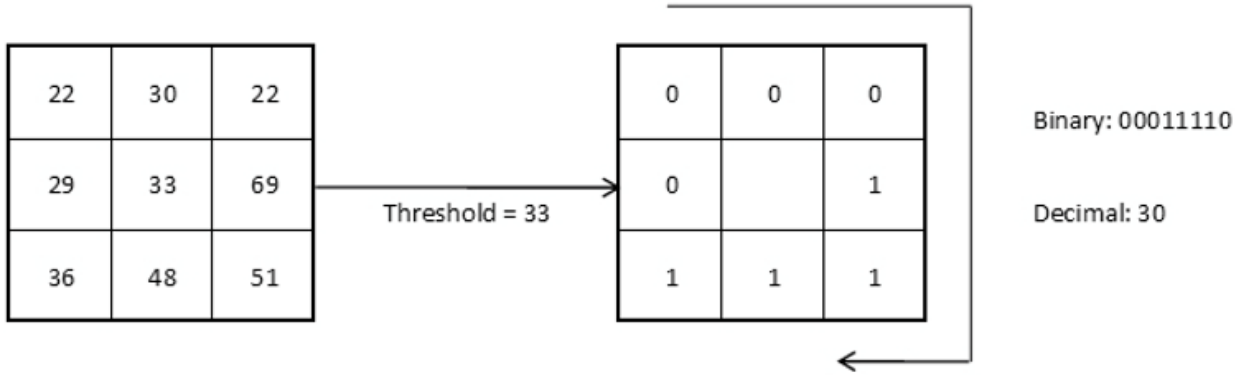

Рисунок 1.1 — Базовий оператор локального бінарного шаблона

Використання кругової околиці і білінійної інтерполяції значень інтенсивностей пікселів дозволяє побудувати локальний бінарний шаблон з будь—якою кількістю точок P і радіусом R , що зображено на рисунку 1.2.

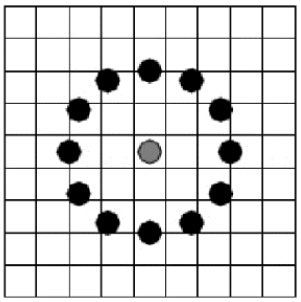

Рисунок 1.2 — Розширений оператор локального бінарного шаблона

Таким чином, бінарні коди несуть в собі інформацію про структуру зображення — кінці ліній, межі, кути, плями. Приклад ознак, що детектуються локальними бінарними шаблонами, зображений на рисунку 1.3

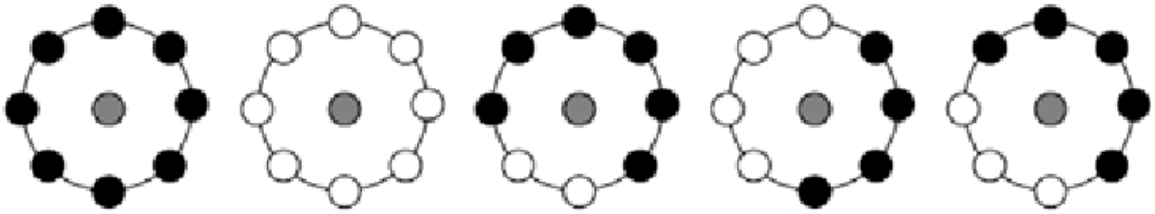

Рисунок 1.3 — Приклад локальних ознак, які детектуються локальними бінарними шаблонами

1.1.2 Метод гістограми орієнтованих алгоритмів.

Подальший розвиток ідея вилучення ознак із зображення отримала в алгоритмі гістограми орієнтованих градієнтів (HOG) [5]. Основна ідея алгоритму гістограми орієнтованих градієнтів — допущення, що зовнішній вид і форма об'єкта на ділянці зображення можуть бути описані розподілом градієнтів інтенсивності або напрямком країв — дескрипторами. Приклад представлення зображення за допомогою дескрипторів зображений на рис. 1.4

Рисунок 1.4 Вихідне зображення та гістограма орієнтованих градієнтів

Реалізація цих дескрипторів може бути проведена шляхом поділу зображення на маленькі пов'язані області (осередку), і розрахунком для кожного осередку гістограми напрямків градієнтів або напрямків країв для пікселів, що знаходяться всередині осередку. Комбінація цих гістограм і є дескриптором. Для збільшення точності локальні гістограми піддаються нормалізації по контрасту.

З цією метою обчислюється міра інтенсивності на великій фрагменті зображення, який називається блоком, і отримане значення використовується для нормалізації. Нормалізовані дескриптори мають кращу инваріантностью по відношенню до висвітлення.

Кінцевим кроком в розпізнаванні об'єктів з використанням гістограми орієнтованих градієнтів є класифікація дескрипторів за допомогою системи навчання з учителем

Переваги методу орієнтованих градієнтів:

— оскільки HOG працює локально, метод підтримує інваріантість геометричних і фотометричних перетворень, за винятком орієнтації об'єкта. Подібні зміни з'являються тільки у великих фрагментах зображення.

— розбиття простору, точне обчислення напрямків і сильна локальна фотометрична нормалізація дозволяють ігнорувати різні положення тварин.

1.2 Існуючі проблеми в задачі розпізнавання зображень тварин.

Описані методи засновані на виділенні ознак. У цьому міститься основна проблема цих методів — структура ознак задається вручну різними способами. Це означає, що під кожну конкретну задачу необхідно сформувати власний набір ознак. Це може викликати складності в завданні розпізнавання тварин, у зв'язку з особливістю об'єктів, що виявляються:

— об'єкти мають високу внутрішньокласову мінливість (колір шерсті, освітлення, відстань, розмір і т.д.).

— об'єкти можуть перекривати один одного.

Алгоритми, розглянуті раніше, можуть виділяти тільки дуже низькорівневі ознаки, такі, як межі, кути, плями і т.д., що не дозволяє оперувати більш високорівневими ознаками об'єктів. Дану проблему можна вирішити використанням згорткових нейронних мереж для розпізнавання зображень тварин. Детальніше такі моделі будуть розглянуті далі.

1.3 Обгрунтування вибору аналогу до програми розпізнавання зображень тварин.

Пошук програм для розпізнавання зображень тварин показав, що їх є дуже мало. Однією з програм—аналогів, яка вирішує поставлену задачу є інструмент під назвою Google ML Kit [6]. Скріншот інструмента Google ML Kit: зображено на рис. 1.5.

Рисунок 1.5 — Скріншот інструмента Google ML Kit

Інструмент Google ML Kit має кілька модифікацій, але побудованай на використанні пірамідальної нейронної мережі, яка працює із попередньо отриманими ознаками зображення. Достовірність розпізнавання зображень тварин у Google ML Kit становить в середньому 72 %.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН НА ОСНОВІ НЕЙРОМЕРЕЖ

2.1 Обгрунтування вибору типу нейронної мережі

Штучні нейронні мережі — це обчислювальні системи, натхнені біологічними нейронними мережами, що складають мозок тварин. Такі системи навчаються задач (поступально покращують свою продуктивність на них), розглядаючи приклади, загалом без спеціального програмування під задачу. Наприклад, у розпізнаванні зображень вони можуть навчатися ідентифікувати зображення, які містять тварин, аналізуючи приклади зображень, мічені як «тварина» і «не тварина», і використовуючи результати для ідентифікування тварин в інших зображеннях. Вони роблять це без жодного апріорного знання про тварин, наприклад, що вони мають хутро, хвости, вуса та ін. Натомість, вони розвивають свій власний набір доречних характеристик з навчального матеріалу, який вони оброблюють [2,3].

Обгрунтуємо вибору типу нейронної мережі для задачі розпізнавання зображень тварин.

2.1.1 Нейронна мережа прямого поширення

Це один з найпростіших видів штучних нейронних мереж. У цьому типі нейронної мережі дані проходять через різні вхідні вузли до тих пір, поки вони не досягнуть вихідного вузла.

Із цього слідує, що дані рухаються лише в одному напрямку від першого рівня вгору, поки не доходять до вихідного вузла. Це також відоме як передня розповсюджена хвиля, яка зазвичай досягається за допомогою класифікаційної функції активації.

На відміну від більш складних типів нейронних мереж, у ній немає зворотного розповсюдження і дані рухаються лише в одному напрямку. Нейронна мережа може мати один шар або вона може мати приховані шари, що

дозволяють обчислювати суму продуктів входів та їх ваги, які потім подаються на вихід.

Нейронні мережі прямого поширення застосовуються в технологіях для розпізнавання обличчя та комп'ютерного зору, а також дана мережа обладнана для обробки даних, що містять багато шуму. Ця нейронна мережа досить проста в обслуговуванні. Нижче ми можемо бачити приклад нейронної мережі прямого поширення (Feedforward Neural Network) (рис.2.1) [2].

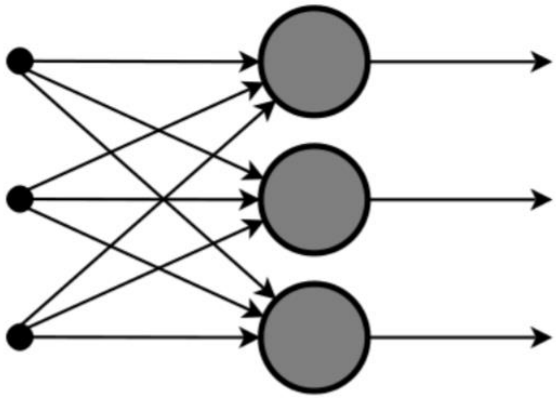

Рисунок 2.1 — Нейронна мережа прямого поширення

2.1.2 Мережа радіальних базисних функцій (Radial Basis Function Neural Network)

Мережа радіальних базисних функцій (RBF) — це поширений тип штучної нейронної мережі для задач наближення функції. Функціональні мережі радіальної бази відрізняються від інших нейронних мереж завдяки їх універсальному наближенню та більшій швидкості навчання. Мережа RBF — це тип нейронної мережі подачі вперед, що складається з трьох шарів, а саме вхідного шару, прихованого рівня та вихідного шару, кожен з цих шарів має різні завдання (рис.2.2).

Навчання моделі RBF припиняється після того, як обчислена помилка досягне бажаних значень (наприклад 0,01) або вже завершена кількість ітерацій тренувань (наприклад 500). Вибирається мережа RBF з певною кількістю вузлів

(наприклад 10) у своєму прихованому шарі. Функція Гаусса використовується як функція передачі даних в обчислювальних одиницях. Залежно від конкретного випадку, як правило, спостерігається, що мережі RBF потрібно менше часу для завершення навчання.

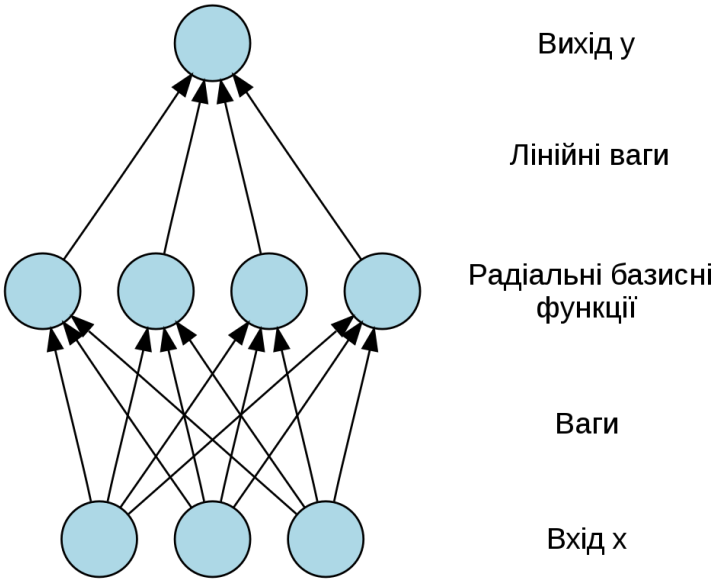

Рисунок 2.2 — Нейронна мережа радіальних базисних функцій

Нейронна мережа радіальних базисних функцій широко функціонує в системах відновлення енергії. В останні десятиліття енергосистеми стають більшими та складнішими, що збільшує ризик затемнення. Ця нейронна мережа використовується в системах відновлення енергії з метою відновлення живлення в найкоротші терміни [3].

2.1.3 Згорткові нейронні мережі

Згорткова нейронна мережа (ЗНМ) — це особлива архітектура штучної нейронної мережі, що імітує особливості зорової області кори головного мозку.

Згорткова нейронна мережа будується на основі операції згортки, що дозволяє навчати ЗНМ на окремих частинах зображення, ітераційно збільшуючи локальну область навчання окремого ядра згортки.

Ключовим моментом в розумінні ЗНМ є поняття так званих спільних ваг, тобто частина нейронів деякого шару нейронної мережі може використовувати одні і ті самі вагові коефіцієнти. Нейрони, що використовують одні й ті самі ваги, об’єднуються в карти ознак, а кожен нейрон карти ознак пов’язаний з частиною нейронів попереднього шару. При обчисленні мережі виходить, що кожен нейрон виконує згортку деякої області попереднього шару (яка визначається множиною нейронів, пов’язаних з даними нейроном). Шари нейронної мережі, побудовані описаним чином, називаються згортковими шарами. Крім згорткових шарів в згортковій нейронній мережі можуть бути шари агрегації (субдискретізації), що виконують функції зменшення розмірності карти ознак, і повнозв’язані шари (класифікатор, який знаходиться на виході мережі). Згорткові шари та шари агрегації можуть чергуватися, найчастіше шари агрегації розміщують за шарами згортки. Загальний вигляд ЗНМ показано на рисунку 2.3.

Рисунок 2.3 — Загальний вигляд згорткової нейронної мережі

Навчання згорткової нейронної мережі можна розподілити на чотири етапи: ініціалізація вагових коефіцієнтів, пряме проходження еталонних вхідних сигналів, розрахунок функції похибки, обернене поширення похибки та оновлення вагових коефіцієнтів.

Першим етапом навчання згорткової мережі є ініціювання вагових коефіцієнтів. В загальному випадку, якщо в згортковій нейронній мережі використовуються шари згортки, агрегуючі та повнозв’язані шари, то необхідно

ініціювати лише ядра згортки згорткових шарів та повнозв’язані шари. Якщо в мережі використовуються також розгорткові шари, то необхідно також ініціювати їх вагові коефіцієнти.

Для навчання ЗНМ використовується метод зворотного поширення похибки. Однією із серйозних проблем навчання ЗНМ є проблема перенавчання, коли модель добре пояснює тільки приклади з навчальної вибірки, адаптуючись до навчальних прикладів, замість того щоб вчитися класифікувати приклади, які не брали участі в навчанні (втрачаючи здатність до узагальнення). Існують наступні підходи для уникнення такої ситуації: штучні дані, рання зупинка, обмеження кількості параметрів, методи проріджування або виключення, виключення з’єднань, ослаблення ваг, ієрархічних координатних сіток.

Метод Dropout також значно покращує швидкість тренування, що робить поєднання моделей практичним, навіть для глибинних нейронних, і послаблює взаємодії між вузлами, ведучи їх до навчання надійніших ознак, що краще узагальнюються на нові дані [4].

2.1.4 Глибинна мережа переконань

В машинному навчанні глибинна мережа переконань (ГМП) — це породжувальна графова модель, або, інакше, один із типів глибинних нейронних мереж, що складено з кількох шарів латентних змінних («прихованих вузлів»), зі з'єднаннями між шарами, але не між вузлами всередині кожного шару.

При тренуванні на наборі прикладів спонтанним чином ГМП може навчатися ймовірнісно відбудовувати свої входи. Шари тоді виступають в ролі детекторів ознак на входах. Після етапу навчання ГНМ може бути треновано далі керованим чином для здійснення класифікації.

ГМП можна розглядати як композицію простих, спонтанних мереж, таких як обмежені машини Больцмана (ОМБ) або автокодувальники, в якій

прихований шар кожної підмережі слугує видимим шаром для наступної. Це також веде до швидкої пошарової процедури спонтанного тренування, в якій порівняльна розбіжність застосовується до кожної підмережі по черзі, починаючи з «найнижчої» пари шарів (де найнижчим видимим шаром є тренувальний набір).

Спостереження, зроблене Yee—Whye Teh, учнем Джефрі Хінтона, про те, що ГМП може бути треновано жадібно шар за шаром, привело до одного з перших дієвих алгоритмів глибинного навчання.

Схематичне представлення глибинної мережі переконань наведено на рисунку 2.4. Стрілки представляють спрямовані з'єднання у графовій моделі, яку представляє мережа.

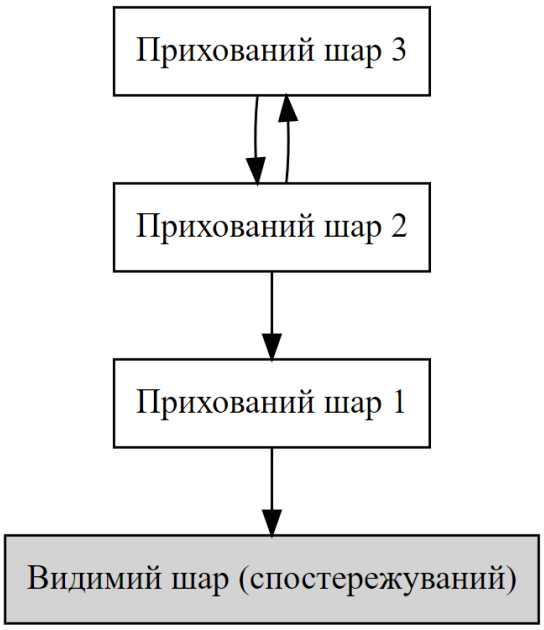

Рисунок 2.4 — Схематичне представлення глибинної мережі переконань

Алгоритм тренування ГМП працює наступним чином. Нехай X буде матрицею входів, що розглядається як набір векторів ознак.

1) натренувати обмежену машину Больцмана на X, щоби отримати матрицю її вагових коефіцієнтів, W. Використовувати її як матрицю вагових коефіцієнтів між двома нижніми шарами мережі;

2) перетворити X за допомогою цієї ОМБ для отримання нових даних X', або шляхом вибірки, або обчисленням середньої активації прихованих вузлів;

3) повторювати цю процедуру з $X \leftarrow X'$ для наступної пари шарів, поки не буде досягнуто двох верхніх шарів мережі;

4) здійснити тонке налаштування всіх параметрів цієї глибинної архітектури по відношенню до того, що виконує роль логарифмічної правдоподібності ГМП, або по відношенню до критерію керованого тренування (після додавання додаткових механізмів навчання для перетворення навченого представлення на керовані передбачення, наприклад, лінійного класифікатора) [5].

Для створення програмного забезпечення було обрано глибинну мережу переконань, оскільки вона має графову модель та при навчанні може ймовірнісно відбудовувати свої входи.

2.2 Структура та математична модель глибинної мережі переконань

Глибинна мережа переконань (deep belief networks, DBN) складається з декількох з'єднаних обмежених машин Больцмана і варіаційних автокодувальників. Подібні нейромережі навчаються по блоках, де кожен блок повинен бути натренований кодувати попередній.

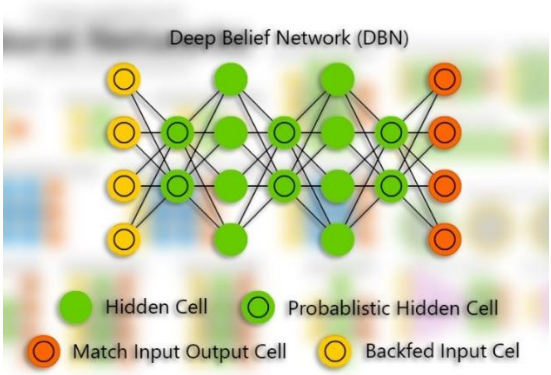

Рисунок 2.5 — Графічне зображення глибинної мережі переконань

Подібні нейромережі навчаються по блоках, де кожен блок повинен бути натренований кодувати попередній. Ця техніка називається "жадібне навчання" і полягає у виборі локальних оптимальних рішень, які не гарантують оптимальний кінцевий результат. Також DBN можна натренувати — методом зворотного поширення помилки — дані у вигляді ймовірнісної моделі [6].

Метод навчання для ГМП, запропонований Джеффрі Хінтоном для використання в навчальних моделях, називається контрастивною дивергенцією (КД). КД забезпечує наближення до методу максимальної ймовірності, який в ідеалі застосовувався б для вивчення ваг. При тренуванні одиночного РБМ оновлення ваги виконується з градієнтним спуском за наступним рівнянням:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\delta \log(p(v))}{\delta w_{ij}}, \tag{2.1}$$

де $p(v)$ — це ймовірність видимого вектора, який задається як (2.2)

$$p(v) = \frac{1}{Z} \sum_h e^{-E(v,h)}, \tag{2.2}$$

де Z — функція розділу (використовується для нормалізації);

$E(v,h)$ — енергетична функція, присвоєна стану мережі.

Менша енергія вказує на те, що мережа перебуває у більш "бажаній" конфігурації. Градієнт $\frac{\delta \log(p(v))}{\delta w_{ij}}$ має просту форму (2.3)

$$w_{ij}(t+1) = w_{ij}(t) + \eta \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}, \tag{2.3}$$

де $\langle \dots \rangle_p$ представляють середні показники щодо розподілу p.

Питання виникає під час відбору проб $\langle v_i h_j \rangle_{model}$ оскільки для цього потрібна розширена вибірка Гіббса, що чергується . КД замінює цей крок, виконуючи чергування вибірки Гіббса для n кроки (значення $n = 1$ добре

виконувати). Після n кроків, дані відбираються, і цей зразок використовується замість $\langle v_i h_j \rangle_{model}$. Процедура КД працює наступним чином. Спочатку ініціалізюються видимі одиниці до навчального вектора.

Після цього паралельно оновлюються приховані одиниці, враховуючи видимі одиниці:

$$p(h_j = 1|V) = \sigma(b_j + \sum_i v_i w_{ij}), \tag{2.4}$$

де σ — сигмоїдна функція і b_j є упередженість h_j.

Паралельно оновлюються видимі одиниці, враховуючи приховані одиниці:

$$p(v_i = 1|H) = \sigma(a_i + \sum_j h_j w_{ij}), \tag{2.5}$$

де a_i є упередженість v_i. Це називається кроком "реконструкції".

Потім повторно оновлюються приховані одиниці паралельно, враховуючи відновлені видимі одиниці.

І в кінці виконується оновлення ваги:

$$\Delta w_{ij} \propto \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{reconstruction}, \tag{2.6}$$

Після того, як RBM тренується, інший RBM "укладається" на нього, беручи його вхідні дані з остаточного тренованого шару. Новий видимий шар ініціалізується до навчального вектора, і значення для одиниць у вже навчених шарах призначаються з використанням поточних ваг та упереджень. Потім новий RBM проходить навчання відповідно до наведеної вище процедури. Весь цей процес повторюється до тих пір, поки не буде виконаний бажаний критерій зупинки. Хоча наближення КД до максимальної ймовірності є грубим (не відповідає градієнту будь—якої функції), воно є емпірично ефективним [5].

2.3 Обґрунтування вибору функції активації

Функція активації визначає вихідне значення нейрона в залежності від результату зваженої суми входів і порогового значення. Розглянемо нейрон:

$$Y = \sum(weight * input) + bias. \quad (2.7)$$

Значення Y може бути будь—яким в діапазоні $(-\infty; +\infty)$. Насправді нейрон не знає межу, після якої слід активуватись. Для цієї мети вирішили додавати активаційну функцію. Вона перевіряє вироблене нейроном значення Y на предмет того, чи повинні зовнішні зв'язки розглядати цей нейрон як активований, або його можна ігнорувати.

Лінійна функція являє собою пряму лінію і пропорційна входу (тобто зваженій сумі на цьому нейроні).

$$A = cx. \quad (2.8)$$

Такий вибір функції активації дозволяє отримувати спектр значень, а не тільки бінарну відповідь. Можна з'єднати декілька нейронів разом і, якщо більше одного нейрона активовано, рішення приймається на основі застосування операції max (або softmax).

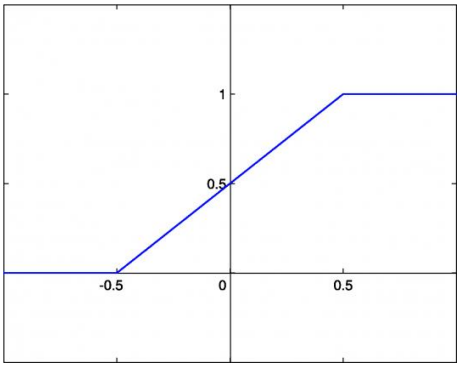

Рисунок 2.6 — Графік лінійної функції активації

Похідна від $A = cx$ по x дорівнює c. Це означає, що градієнт ніяк не пов'язаний з X. Градієнт є постійним вектором, а спуск проводиться по постійному градієнту. Якщо проводиться помилкове пророцтво, то зміни, зроблені зворотним поширенням помилки, теж постійні і не залежать від зміни на вході $delta(x)$.

Розглянемо пов'язані шари. Кожен шар активується лінійною функцією. Значення з цієї функції йде в наступний шар в якості входу, другий шар вважає зважену суму на своїх входах і, в свою чергу, включає нейрони в залежності від іншої лінійної функції активації. Не має значення, скільки шарів ми маємо. Якщо всі вони за своєю природою лінійні, то фінальна функція активації в останньому шарі буде просто лінійною функцією від входів на першому шарі. Це означає, що N шарів можуть бути замінені одним шаром. Ми втратили можливість робити набори з шарів. Не важливо, як ми навчаємо, вся нейронна мережа все одно буде подібна одному шару з лінійною функцією активації.

Сигмоїда виглядає гладкою і подібна ступінчастої функції.

$$A = \frac{1}{1+e^{-x}}. \tag{2.9}$$

По—перше, сигмоїда — нелінійна за своєю природою, а комбінація таких функцій виробляє теж нелінійну функцію.

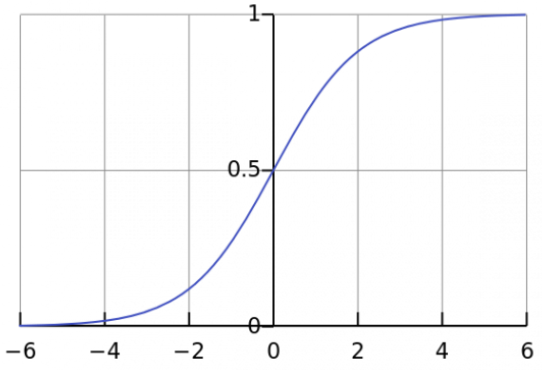

Рисунок 2.7 — Графік сигмоїдної функції активації

Ще одна перевага такої функції — вона не бінарна, що робить активацію аналоговою, на відміну від ступінчастої функції. Для сигмоїди також характерний гладкий градієнт. В діапазоні значень X від —2 до 2 значення Y змінюється дуже швидко . Це означає, що будь—яка мала зміна значення X в цій області тягне істотні зміни Y. Така поведінка функції вказує на те, що Y має тенденцію притискатися до одного з країв кривої.

Сигмоїда дійсно виглядає підходящої функцією для задач класифікації. Вона прагне привести значення до однієї зі сторін кривої. Така поведінка дозволяє знаходити чіткі межі при прогнозі.

Інша перевага сигмоїди над лінійною функцією полягає в наступному. У першому випадку маємо фіксований діапазон значень функції — $[0,1]$, тоді як лінійна функція змінюється в межах $(-\infty; +\infty)$. Така властивість сигмоїди дуже корисна тому, що не призводить до помилок в разі великих значень активації [7].

Потрібно вирішити, яку з функцій активації використовувати. Коли відомо деякі характеристики функції, яку потрібно апроксимувати, потрібно обирати активаційну функцію, яка апроксимує шукану функцію краще і веде до більш швидкого навчання. Наприклад, сигмоїда добре показує себе в задачах класифікації. Так як апроксимацію функції класифікації комбінацією сигмоїд можна провести легше, ніж використовуючи ReLu, наприклад, то для розроблюваного програмного забезпечення було обрано сигмоїдну функцію активації.

2.4 Розробка структури нейромережевої системи

Глибинна мережа переконань — це особливий вид нейронних мереж, де для належного навчання їй потрібно пройти стадії безконтрольного навчання на кожному рівні, а потім супроводжуватися керованим навчанням у всій мережі.

Він визначається як стек обмежених машин Больцмана, де кожен шар взаємодіє як з попереднім, так і з наступним шарами.

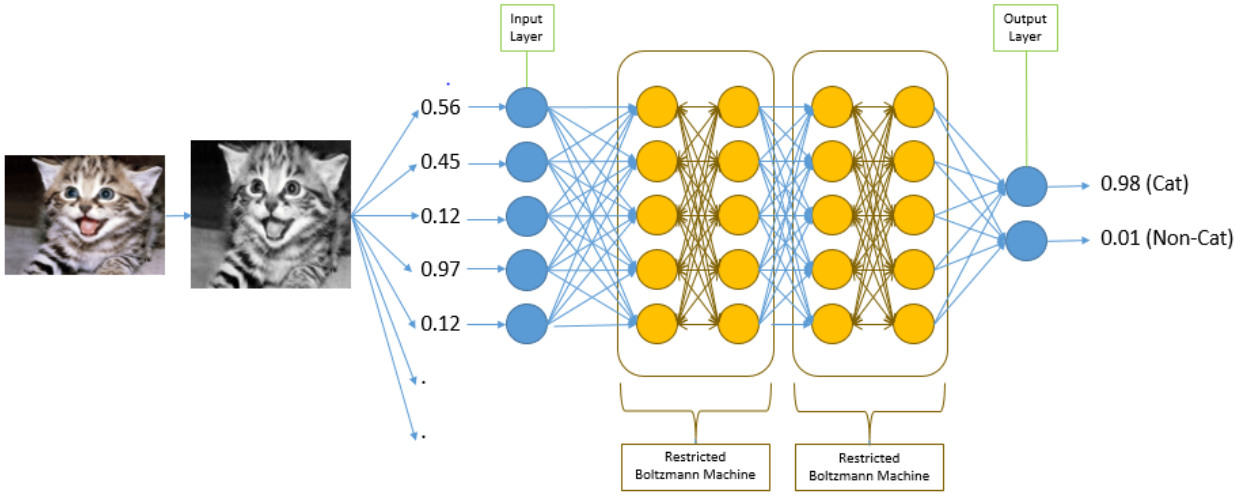

Рисунок 2.8 — Схема глибинної мережі переконань

Хоча спочатку це здається складним, цю мережу просто потрібно навчити, використовуючи цілий набір даних на кожній обмеженій машині Больцмана, спочатку без нагляду, а після нижчого порогового рівня помилок, навчання може переходити до контрольованого етапу навчання. В цілому мережа, що використовує ті самі траєкторії зворотного поширення або еластичного зворотного поширення, які також використовуються для повнозв'язних мереж [6]. Структура розробленої глибинної мережі переконань представлена на рис. 2.9.

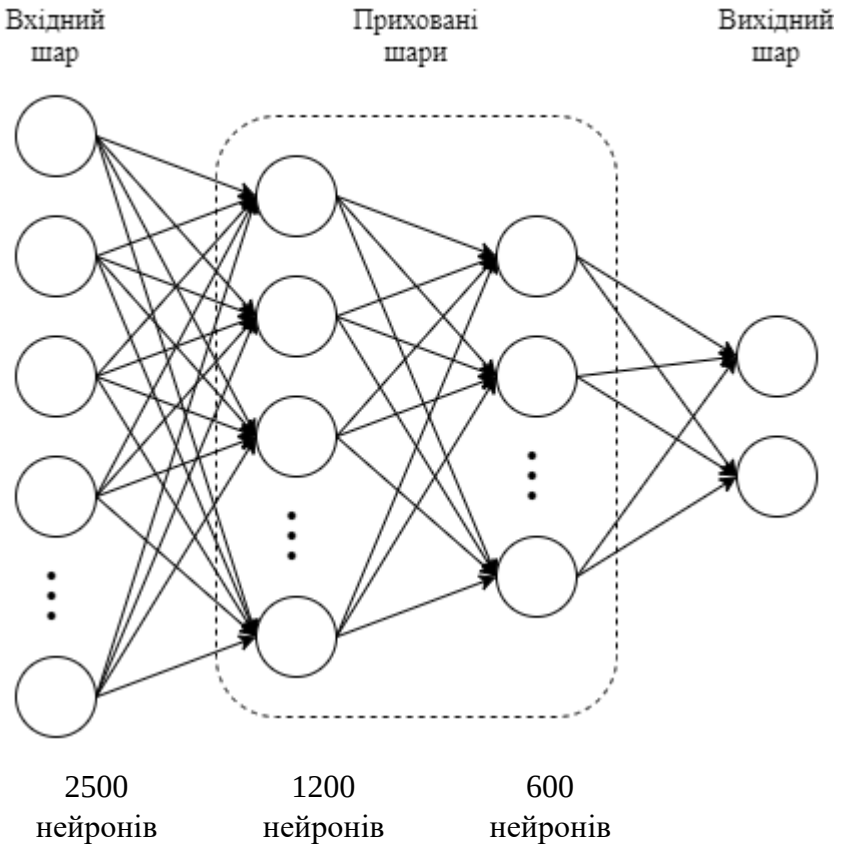

Рисунок 2.9 — Структура нейромережевої системи

Для розробленої мережі було обрано такі параметри:

— розмір вхідного зображення – 50*50;

— вхідний шар складається з 2500 нейронів;

— 2 прихованих шари.

— вихідний шар складається з 2 нейронів;

— функція активації – сигмоїдна;

— помилка, для зупинки тренування сторінки – 0,0001;

— помилка, для зупинки тренування мережі – 0,001;

— максимальна кількість ітерацій на сторінці – 1000;

— максимальна кількість безконтрольних ітерацій на сторінці – 100.

2.5 Структури даних програми

В програмуванні та комп'ютерних науках структура даних — це спосіб організації даних в комп'ютерах. Часто разом зі структурою даних пов'язується

і специфічний перелік операцій, що можуть бути виконаними над даними, організованими в таку структуру.

Правильний підбір структур даних є надзвичайно важливим для ефективного функціонування відповідних алгоритмів їх обробки. Добре побудовані структури даних дозволяють оптимізувати використання машинного часу та пам'яті комп'ютера для виконання найкритичніших операцій.

Відома формула «Програма = Алгоритми + Структури даних» дуже точно виражає необхідність відповідального ставлення до такого підбору. Тому іноді навіть не обраний алгоритм для обробки масиву даних визначає вибір тієї чи іншої структури даних для їх збереження, а навпаки [8].

В мові програмування C# наявні такі структури даних:

— бінарне дерево — це структура даних, яка складається з вузлів, при цьому кожен вузол може мати не більше двох дочірніх, де перший вузол називається кореневим або батьківським, а дочірні — правим і лівим спадкоємцем (нащадком).

— граф — абстрактна структура даних, яка складається з набору вершин і з'єднань між ними — ребер . При цьому кожне ребро може мати вагу;

— стек — це абстрактна структура даних, в якій елементи організовані за принципом LIFO (Last In First Out — "прийшов останнім, вийшов першим") [8];

— масив (список) — деяка кількість елементів, упорядкованих певним чином, зазвичай ці елементи належать до одного й того ж типу, доступ до елементів відбувається завдяки цілочисельному індексу необхідного елементу (хоча самі елементи можуть бути майже будь—якого типу);

— зв'язаний список — це лінійна колекція елементів даних будь—якого типу, які називаються вузлами, де кожен вузол містить дані, і вказівник на наступний вузол у зв'язаному списку, навідміну від зв'язного списку є те, що елементи завжди можна ефективно додавати й видаляти без перерозподілу всього списку, але інші операції, такі як довільний доступ до конкретного елементу, є повільнішими в таких списках, ніж у масивах;

— асоціативний масив (словник, map) — більш гнучка варіація масиву, можна вільно додавати та видаляти пари «ім'я—значення»;

— запис (структура) — агрегована структура даних, що містить інші значення, зазвичай це фіксоване число та послідовність, які індексуються поіменно;

— об'єднання — структура даних, що зазначає, які із дозволених примітивних типів можуть зберігатись у її екземплярах, на відміну від запису, який можна визначити таким чином, щоб він зберігав і ціле число, і число з рухомою комою, у об'єднанні можна зберігати лише один певний тип, для зберігання об'єднання виділяється стільки пам'яті скільки необхідно для найбільшого типу даних [8].

У розроблюваній програмі будуть використовуватись:

— масиви, для роботи з файлами та зображеннями;

— списки, для проведення математичних операцій, що мають заздалегідь невідому кількість вихідних рішень.

2.6 Структура та алгоритм роботи програмного забезпечення

Під час етапу проектування було розроблено структуру та алгоритм розроблюваного програмного забезпечення. Розглянемо детальніше модулі, які будуть входити до системи та етапи виконання програми.

Структура розроблюваного програмного забезпечення представлена на Рис. 2.10.

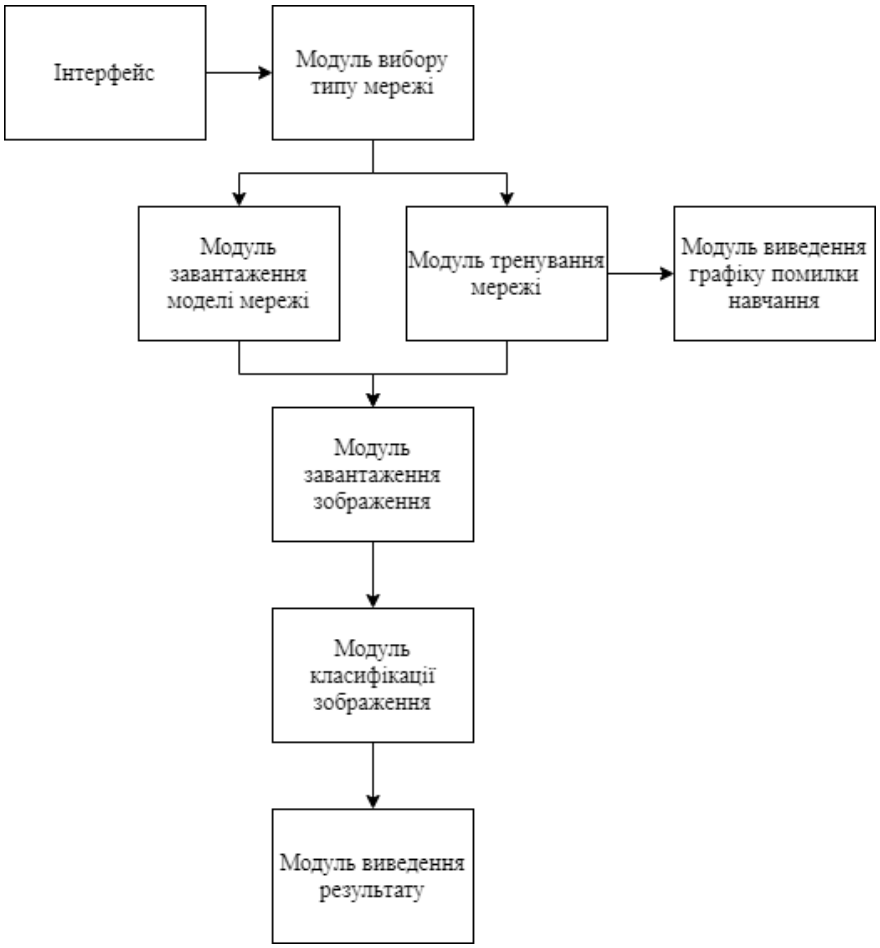

Рисунок 2.10 — Структура програмного забезпечення

Розроблювана система буде складатись з наступних модулів:

— 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.

Інтерфейс пов’язаний з більшістю модулів, таких як: модуль вибору типу мережі, модуль завантаження моделі мережі, модуль тренування мережі, модуль виведення графіку помилки навчання, модуль завантаження

зображення, модуль виведення результату. Інтерфейс виконує функцію відображення результатів тренування, тестування та класифікації зображень.

Модуль вибору типу моделі дозволяє користувачеві обрати між двома нейронними мережами: Accord.Net і ConvNetSharp. В залежності від вибору типу мережі, користувач має можливість провести класифікацію зображення за допомогою різних нейронних мереж, які мають різні вхідні параметри та функції активації.

Модуль завантаження моделі мережі необхідний для можливості завантаження натренованої моделі нейронної мережі обраного типу. Якщо моделі мережі немає, то потрібно провести тренування мережі за допомогою модулю тренування мережі. Модуль завантаження зображення дозволяє завантажувати зображення для тренування, тестування та класифікації. Модуль виведення графіку помилки навчання виводить графік помилки навчання в реальному часі, тому з його допомогою можна відстежувати прогрес навчання, якого досягла модель.

Модуль класифікації зображення проводить локалізацію зображення за допомогою анкерних коробок та визначає чи на завантаженому зображенні знаходиться кіт. Модуль виведення результату виводить отриманий в результаті класифікації результат на інтерфейс користувача і показує чи на зображенні «Кіт» чи «Не кіт», яке виводиться на вхідному зображенні разом з анкерними коробками.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН НА ОСНОВІ НЕЙРОМЕРЕЖ

3.1 Обґрунтування вибору середовища програмування

Нині існує багато середовищ розробки програм [9,10]. Вони поділяються на редактори коду, інтегровані середовища розробки (IDE) та хмарні IDE. Кожен обирає те середовище, яке йому найзручніше для використання. Наведемо таблицю, у якій порівнюються декілька середовищ розробки з різних категорій (табл. 3.1).

Таблиця 3.1 — Порівняння середовищ розробки

Назва	JetBrains IDE’s	Codeanywhere	Atom	Adobe Brackets	VS Code
Підсвітка	+	+	+	+	+
Гарячі клавіші	+	+	+	+	+
Перевірка орфографії	+		+	+	+
Розширення	+		+	+	+
Вкладки		+	+	+	+
Порівняння двох файлів			+	+	+
Підтримка Git	+	+	+	+	+
Відладка	+	+	+	+	+

Для створення системи було обрано Visual Studio Code, оскільки він багатофункціональний і компактний кодовий редактор. Також за допомогою технології IntelliSense редактор може дописувати назви оголошених раніше класів та функцій. Підказки містять посилання на документацію з потрібним матеріалом.

3.2 Обґрунтування вибору мови програмування

Програмне забезпечення можна розробити багатьма мовами. Проведемо порівняння найпопулярніших мов програмування [11] та оберемо найкращу для розробки (табл. 3.2).

Таблиця 3.2 — Порівняння мов програмування

Мова	Передбачуване використання	Імперативне	Об'єктно—орієнтоване	Функційне	Процедурне	Узагальнене	Рефлективне
C#	Application, RAD, business, client—side, general, server—side, Web, Robotics	X	X	X		X	X
C++	Application, system	X	X	X	X	X	
Python	Application, general, Web, scripting, AI, scientific computing	X	X	X			X
Java	Application, business, client—side, general, server—side, Web	X	X			X	X
JavaScript	Client—side, Server—side, Web	X	X	X			X

Для створення програмного забезпечення було обрано мову програмування C# тому, що це об'єктно—орієнтована мова програмування з безпечною системою типізації. Мова має строгу статичну типізацію, підтримує

поліморфізм, перевантаження операторів, вказівники на функції—члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML.

3.3 Використання фреймворку Accord.Net

Accord.NET — це фреймворк для наукових обчислень у .NET. Фреймворк містить набір бібліотек, доступних у вихідному коді, а також через виконувані інсталятори та пакунки NuGet . Основні сфери охоплюють числову лінійну алгебру, чисельну оптимізацію, статистику, машинне навчання, штучні нейронні мережі, обробку сигналів та зображень, а також бібліотеки підтримки. Проект був спочатку створений для розширення можливостей AForge.NET Framework , але з тих пір включив AForge.NET всередину себе. Новіші випуски об’єднали обидва фреймворки під назвою Accord.NET.

Accord.NET Framework був представлений у багатьох книгах, таких як Mastering.NET Machine Learning, видавництво PACKT та F# для програм машинного навчання, представлений у QCON San Francisco. Багато наукових публікацій було опубліковано з його використанням [12].

3.4 Реалізація глибинної мережі переконань для розпізнавання зображень за допомогою Accord.Net

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. Еквалізацію гістограм також можна зробити, але просте перетворення зображення у напівтонове (сіре) зберігає його ознаки більш природніми. Можна використовувати і будь—який інший розмір,

наприклад, 75x75 або 100x100, але збільшення роздільної здатності також збільшить кількість вхідних нейронів і різко збільшить час навчання та розпізнавання, коли використовуються мережі без згортки.

```
public static Image<Gray, Byte>
   ConvertOriginalImageToGrayScaleAndProcess(Image<Bgr, Byte> orginalImage)
{
   var grayScale = orginalImage.Convert<Gray, Byte>();
   return grayScale.Resize(50,50, Emgu.CV.CvEnum.Inter.Cubic, false);
}
```

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.

```
public static double[] GetNetworkFeedArray(Image<Gray, Byte> image)
{
   var imageBytes = image.Bytes;
   double[] networkFeed = new double[imageBytes.Count()];
   for (int i = 0; i < imageBytes.Length; i++)
   {
      networkFeed[i] = ((double)imageBytes[i] / 256);
   }
   return networkFeed;
}
```

Глибинна мережа переконань навчається належним чином шляхом неконтрольованого навчання на кожній обмеженій машині Больцмана, після чого відбувається контрольоване навчання по всій мережі. Отже, для цього потрібні два різні алгоритми навчання. У Accord.NET є багато класів, які можна використати, але для створення глибинної мережі переконань існує

DeepBeliefNetwork клас. Для неконтрольованого навчання існує DeepBeliefNetworkLearning клас, а для контрольованого навчання доступні BackpropagationLearning, ResilientBackpropagationLearning або ParallelResilientBackpropagationLearning класи на вибір. Глибинна мережа переконань ініціалізується випадковими вагами за допомогою NguyenWidrow класу. Наведений нижче код використовується для двійкової класифікації, отже, кількість вихідних нейронів становить лише 2. Він має два приховані шари з 1200 і 600 нейронами в них відповідно. Ініціалізація мережі:

```
public AccordNetwork()
{
   network = new DeepBeliefNetwork(new BernoulliFunction(), inputLength, 1200, 600, 2);
   new NguyenWidrow(network).Randomize();
   network.UpdateVisibleWeights();
   unsuperVisedTeacher = GetUnsupervisedTeacherForNetwork(network);
   supervisedTeacher = GetSupervisedTeacherForNetwork(network);
}
```

Параметри навчання мережі:

```
private DeepBeliefNetworkLearning
    GetUnsupervisedTeacherForNetwork(DeepBeliefNetwork deepNetwork)
{
   var teacher = new DeepBeliefNetworkLearning(deepNetwork)
   {
      Algorithm = (hiddenLayer, visibleLayer, i) =>
            new ContrastiveDivergenceLearning(hiddenLayer, visibleLayer)
      {
         LearningRate = 0.1,
```

```
            Momentum = 0.5
        }
    };
    return teacher;
}

private ResilientBackpropagationLearning GetSupervisedTeacherForNetwork
                                (DeepBeliefNetwork deepNetwork)
{
    var teacher = new ResilientBackpropagationLearning(deepNetwork)
    {
        LearningRate = 0.1
        //Momentum = 0.5
    };
    return teacher;
}
```

З набору даних лише вхідні зображення використовуються для неконтрольованого навчання. Нижче наведено код, який бере вхідні зображення з нашого набору даних та навчає всі обмежені машини Больцмана шляхом їх ітерації.

```
private void TrainUnsupervised(double[][] batchInputs, int iterations,
                        Action<double, int, string> progressCallback)
{
    for (int layerIndex = 0; layerIndex < network.Machines.Count — 1;
layerIndex++)
    {
        unsuperVisedTeacher.LayerIndex = layerIndex;
        var layerData = unsuperVisedTeacher.GetLayerInput(batchInputs);
```

```
            foreach (int i in Enumerable.Range(1, iterations))
            {
                var error = unsuperVisedTeacher.RunEpoch(layerData) / batchInputs.Length;
                if (progressCallback != null)
                {
                    progressCallback(error, i, $"Unsupervised Layer {layerIndex}");
                }
                if (this.ShouldStopTraning)
                {
                    this.ShouldStopTraning = false;
                    break;
                }
            }
        }
    }
```

Другий крок вимагає як вхідних зображень, так і вихідних даних (міток класу) із набору даних. Вони використовуються для контрольованого навчання. Нижче наведено код, який передає набори даних контрольованому вчителю та повідомляє про рівень помилок. Цей коефіцієнт помилок можна відобразити на графіку або для інших цілей.

```
    private void TrainSupervised(double[][] batchInputs, double[][] batchOutputs,
                    int iterations, Action<double, int, string> progressCallback)
    {
        foreach (int i in Enumerable.Range(1, iterations))
        {
            var error = supervisedTeacher.RunEpoch(batchInputs, batchOutputs) / batchInputs.Length;
```

```
            if (progressCallback != null)
            {
                progressCallback(error, i, "Supervised");
            }
            if (this.ShouldStopTraning)
            {
                this.ShouldStopTraning = false;
                break;
            }
        }
    }
```

3.6 Тестування та аналіз результати роботи програми

Проведемо тестування розробленого програмного забезпечення. Спочатку було проведено навчання моделі для нейронної мережі. Для цього потрібно обрати на початковому вікні тип мережі, яку хочете навчити (рис.3.1). В нашому випадку, це мережа Accord.Net.

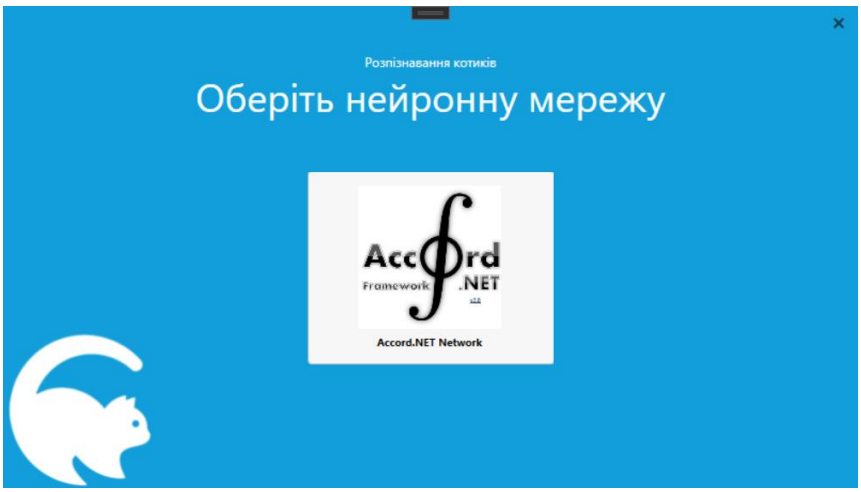

Рисунок 3.1 – Вікно вибору типу мережі

Далі, після вибору типу мережі, необхідно перейти до самого вікна тренування. Для цього необхідно на вікні, яке відкрилось (рис.3.2) натиснути кнопку «Тренування і тестування». Після цього відкриється вікно для тренування і тестування мережі (рис.3.3) в яке можна завантажити зображення.

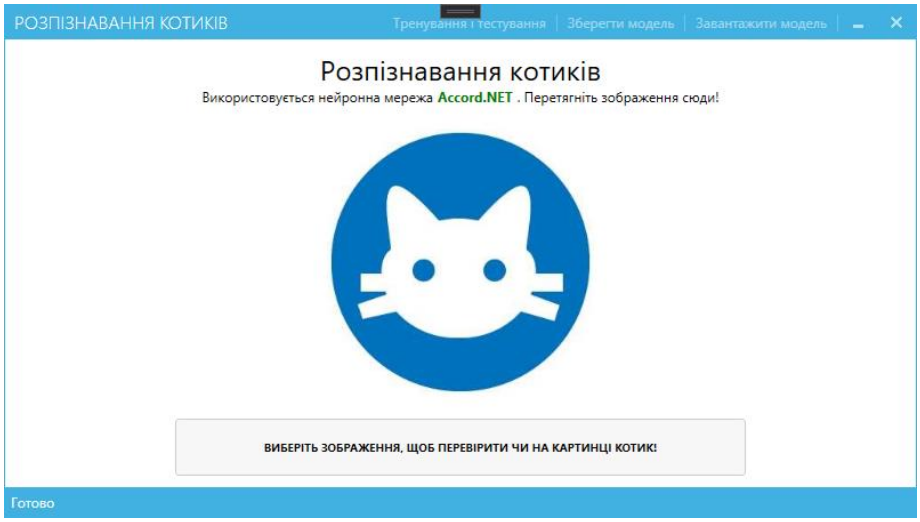

Рисунок 3.2 – Вікно розпізнавання

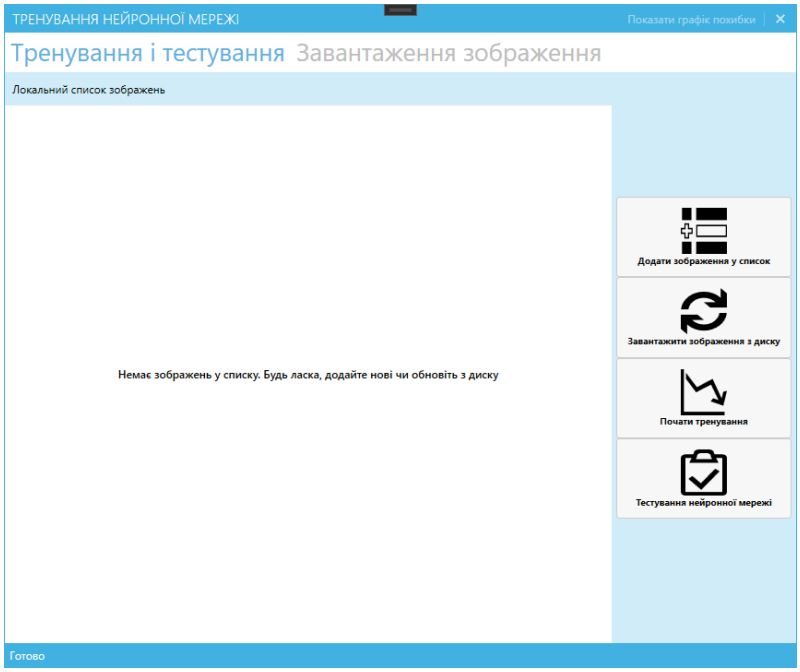

Рисунок 3.3 – Вікно «Тренування і тестування»

Далі потрібно вказати чи на цьому зображені зображений кіт (рис.3.4), і почати тренування. Під час тренування можна переглянути графік помилки навчання, натиснувши на кнопку на вікні тренування і тестування (рис.3.5). Даний графік змінюється динамічно, в залежності від результатів навчання, тому можна постійно, з його допомогою, спостерігати результат навчання.

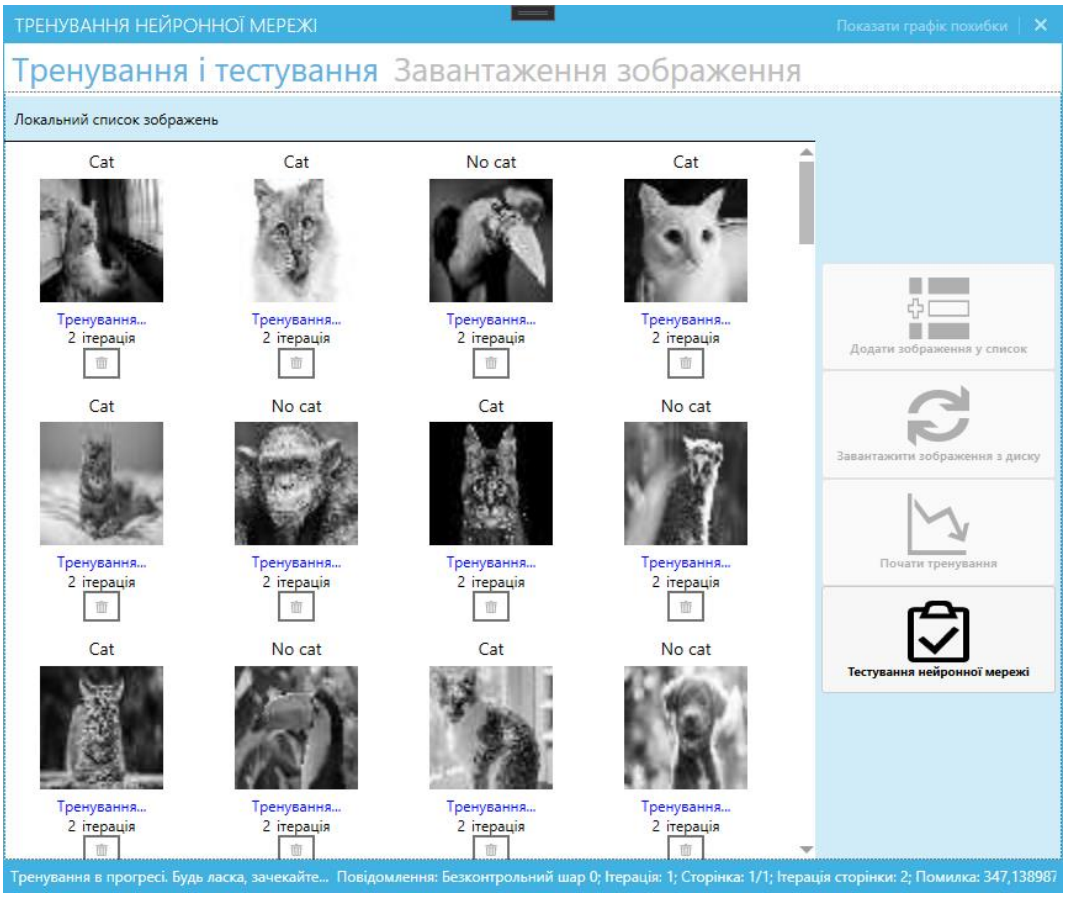

Рисунок 3.4 — Вікно «Тренування і тестування» із завантаженими зображеннями та початком тренування нейронної мережі

Тренування нашої нейронної мережі проходило на вибірці з 500 зображень, з яких 250 – були зображення котів, а інші 250 – зображеннями інших тварин.

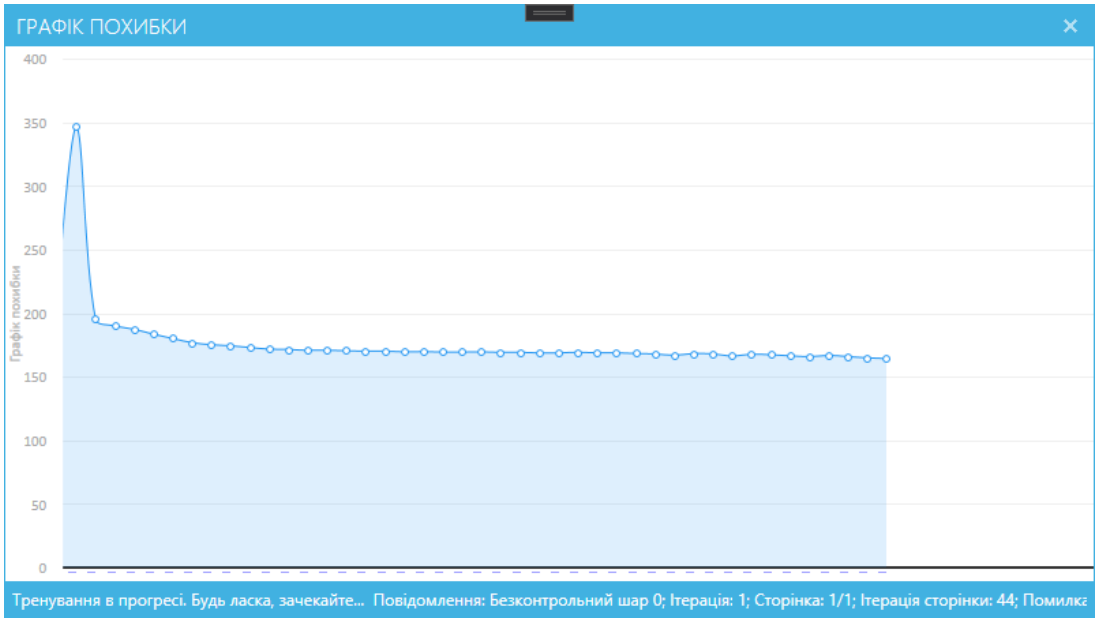

Рисунок 3.5 – Графік зменшення похибки навчання

Після того як мережа буде натренована, потрібно перевірити чи правильно мережа класифікує зображення. Для цього потрібно провести тестування (рис.3.6) на зображеннях, які не входили в тестову вибірку. Завантажимо у вікно (рис.3.2) зображення кота (рис.3.7) та іншої тварини (рис.3.8) та перевіримо чи правильно мережа їх класифікує.

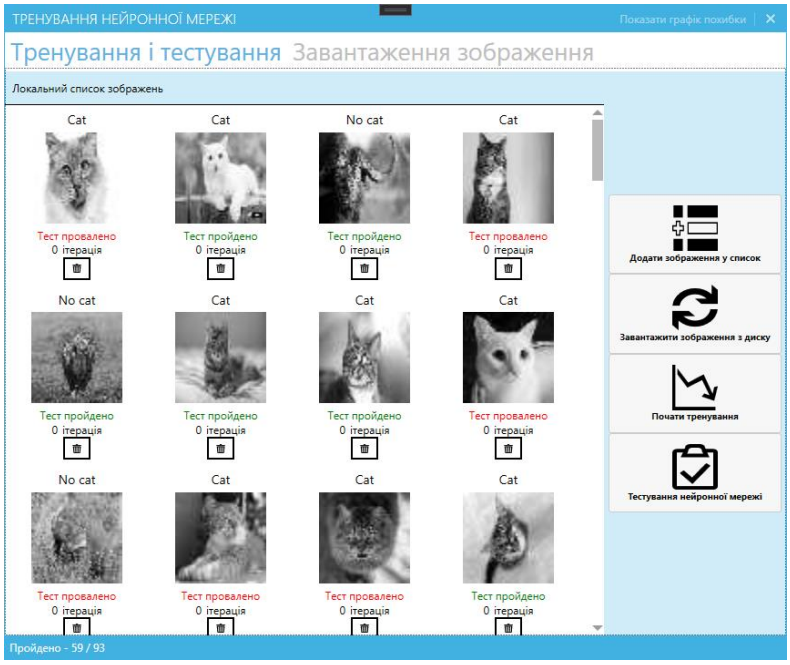

Рисунок 3.6 – Тестування навченої нейронної мережі

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

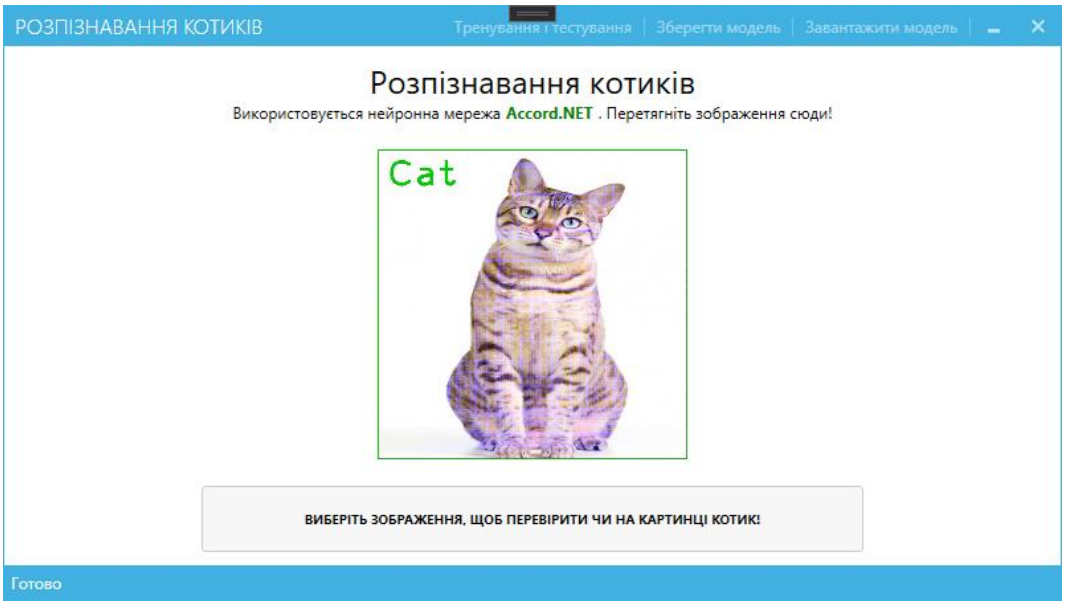

Рисунок 3.7 – Вікно розпізнавання із зображенням кота

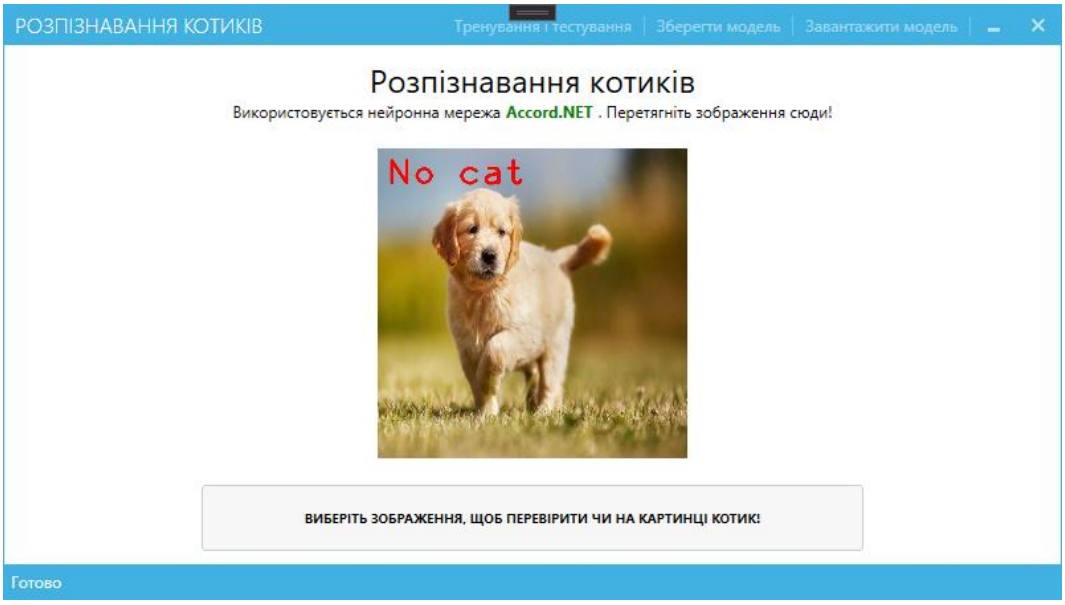

Рисунок 3.8 – Вікно розпізнавання із зображенням іншої тварини

Тестування показує, що навчена нейронна мережа правильно класифікує 83 зображень із 100, які були у тестовій вибірці. У відсотковому співвідношенні результат становить 83%, тобто помилка– 17%, що означає, що мета роботи та параметри, вказані в індивідуальному завданні, досягнуті.

Результати порівняння параметрів розробленої глибинної мережі переконань Accord.Net Network із пірамідальною нейронною мережею розробленою в Google ML Kit, наведено у табл. 3.3.

Таблиця 3.3 – Порівняння параметрів розробленої глибинної мережі переконань Accord.Net Network із мережею Google ML Kit

	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
Google ML Kit	100	72	72%	8 години

Із табл. 3.3 видно, що глибинна мережа переконань Accord.Net Network тренувалась біля 8 годин та досягла похибки в 17% (достовірність 83%). А пірамідальна нейронна мережа Google ML Kit досягла похибки в 28% при при такому ж часу тренування. (достовірність 72%).

Таким чином, можна зробити висновок, що глибинна мережа переконань Accord.Net Network має порівняно зі пірамідальною нейронною мережею Google ML Kit більшу точність розпізнавання образів. Тобто мета роботи досягнута – швидкість навчання підвищена. Але ця перевага досягається за рахунок зменшення достовірності розпізнавання. Значить глибинну мережу переконань Accord.Net Network варто використовувати для задач, де потрібна висока швидкість навчання, але не є критичним невисока достовірність розпізнавання.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково—технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового програмного продукту нейромережевого модулю для розпізнавання зображень, створений для розпізнавання тварин на зображенні.

Особливістю програми є вдосконалення процесу навчання згорткової нейронної мережі, який відрізняється аргументацією тренувальних даних, зміною розміру вікна передбачення об'єктів та використання додаткової регуляризації ваг мережі, що дозволило підвищити достовірність ідентифікації тварин.

Аналогом може бути Google ML Kit вартість ліцензії складає 2350$ або 95000грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3—х незалежних експертів. Оцінювання науково—технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12—ма критеріями, у відповідності із табл. 4.1.

Таблиця 4.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5—ти бальною шкалою)					
Кри-терій	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 роботоздат—ність продукту в реальних умовах

Продовження табл. 4.1

Ринкові переваги					
2	Багато аналогів на малому ринку	Ринкові п 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 аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
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 ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на 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 із 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 використовуються у виробництві

Продовження табл. 4.1

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

Усі дані по кожному параметру занесено в таблиці 4.2

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	3	4
Наявність аналогів на ринку	4	4	4
Цінова політика	3	4	3
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	3	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	2	3	4
Фінансування	4	4	3
Матеріально—технічна база	3	3	3
Термін реалізації ідеї	4	4	4
Супровідна документація	3	3	4
Сума	40	42	43
Середньоарифметична сума балів	(40+42+43) / 3 = 41,67		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0—10	Низький
11—20	Нижче середнього
21—30	Середній
31—40	Вище середнього
41—48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок того, що програмний продукт є нейромережевим модулем для розпізнавання зображень, створений для розпізнавання тварин на зображенні. Особливістю програми є вдосконалення процесу навчання згорткової нейронної мережі, який відрізняється аргументацією тренувальних даних, зміною розміру вікна передбачення об'єктів та використання додаткової регуляризації ваг мережі, що дозволило підвищити достовірність ідентифікації тварин.

4.2 Прогнозування витрат на виконання науково—дослідної (дослідно—конструкторської) роботи

4.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t, \tag{4.1}$$

де М — місячний посадовий оклад конкретного розробника (дослідника), грн.;

$Т_р$ — число робочих днів в місяці, 23 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.1.

Таблиця 4.1 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	50000	2173,91	45	97826,087
Програміст	48500	2108,70	45	94891,304
Всього				192717,39

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

4.2.2 Додаткова заробітна плата розробників, які приймали участь в розробці обладнання.

Додаткова заробітна плата прийнято розраховувати як 13,3 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 13,3\ \% / 100\ \% \quad (4.2)$$

$$З_д = (192717,39 \cdot 13,3\ \% / 100\ \%) = 25631,41 \ \text{(грн.)}$$

4.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (4.3)$$

$$Н_з = (192717{,}39 + 25631{,}41) \cdot 22\ \% / 100\ \% = 48036{,}74 \text{ (грн.)}$$

4.2.4. Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

4.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.]}, \qquad (4.4)$$

де Ц – балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ — термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп’ютер балансова вартість якого становить 28000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,96 міс.

$$A_{обл} = \frac{28000}{2} \times \frac{1{,}96}{12} = 2282{,}61 \text{ грн.}$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.2. Але, так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів є безкоштовною (Visual Studio Code, Accord.Net), то $B_{нем.ак.}$ = 0 грн.

Таблиця 4.2 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза—ційні відрахуван—ня, грн.
Комп'ютер та комп'ютерна периферія	28000	2	1,96	2282,609
Офісне обладнення (меблі)	25000	4	1,96	1019,022
Приміщення	800000	20	1,96	6521,739
Всього				9823,37

5.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1—й або 2—й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п . \quad (4.5)$$

де В — вартість 1 кВт—години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П — встановлена потужність обладнання, кВт. П = 0,5 кВт;

Ф — фактична кількість годин роботи обладнання, годин.

$К_п$ — коефіцієнт використання потужності, $К_п$ = 0,9.

$$В_е = 0,9 \cdot 0,5 \cdot 8 \cdot 45 \cdot 6,2 = 1004,4 \text{ (грн.)}$$

5.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}. \quad (4.6)$$

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

$$I_в = 192717,39 * 90\% / 100\% = 173445,7 \text{ (грн.)}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково—технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}. \quad (4.7)$$

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

$H_{нзв}$ = 192717,39 * 130 % / 100 % = 250533 (грн.)

5.2.9 Витрати на проведення науково—дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково—дослідної роботи:

$B_{заг}$ = 192717,39+25631,41+48036,74+9823,37+1004,40+173445,7+
+250533 = 701191,57 грн.

5.2.11 Розрахунок загальних витрат на науково—дослідну (науково—технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково—дослідної (науково—технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} \text{ (грн)}, \quad (4.8)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково—дослідної роботи.

Так, якщо науково—технічна розробка знаходиться на стадії: науково—дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

ЗВ = 701191,57 / 0,5 = 1402383 грн.

4.3 Розрахунок економічної ефективності науково—технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним 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. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково—технічної розробки необхідно:

1) вказати, з якого часу можуть бути впроваджені результати науково—технічної розробки;

2) зазначити, протягом скількох років після впровадження цієї науково—технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3—х років після її впровадження);

3) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково—технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

4) визначити ціну реалізації на ринку науково—технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково—технічних розробок, розрахунок економічної ефективності науково—технічної розробки за її можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

4.3.1 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:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.10)$$

Де $\pm\Delta Ц_o$ — зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково—технічної розробки в аналізовані періоди часу;

N — кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково—технічної розробки;

$Ц_o$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Цб$ — вартість програмного продукту у році до впровадження результатів розробки;

ΔN — збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

ρ — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 12000 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *1000* грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 13000 шт., протягом другого року – на 10000 шт., протягом третього року на 8000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$\Delta П_1$ = (0*1000 + (12000 + 1000)*13000)* 0,8333* 0,15) * (1 — 0,18) = 15989999,360 грн.
$\Delta П_2$ = (0*1000+(12000+1000)*(13000+10000)* 0,8333* 0,15) * (1 — 0,18) = 30647498,774 грн.
$\Delta П_3$ = (0*1000 + (12000 + 1000)*(13000+10000+8000)* 0,8333* 0,15) * (1 — 0,18) = 41307498,348 грн.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 87944996,48 грн.

4.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

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

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}. \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково—дослідної (науково—технічної) роботи, грн;

Т — період часу, протягом якою виявляються результати впровадженої науково—дослідної (науково—технічної) роботи, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t — період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП = $(15989999,360/(1+0,1)^1)+(30647498,774/(1+0,1)^2)+(41307498,348/(1+0,1)^3)$ = 14536363,05 + 25328511,38 + 31034934,9 = 70899809,34 грн.

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

$$PV = k_{інв} * ЗВ. \quad (4.12)$$

Де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково—технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ — загальні витрати на проведення науково—технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 1402383 = 2804766,29 \text{ грн.}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково—технічної розробки становитиме:

$$E_{абс} = ПП - PV. \quad (4.13)$$

$$E_{абс} = 70899809,34 - 2804766,29 = 68095043,05 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково—дослідної (науково—технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розра—хувати внутрішню економічну дохідність або показник внутрішньої но—рми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь—яку науково—технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (4.14)$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 68095043{,}05/2804766{,}29} - 1 = 1{,}935$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \qquad (4.15)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}. \quad (4.16)$$

$$T_{ок} = 1 / 1{,}935 = 0{,}52 \text{ р.}$$

Оскільки $T_{ок}$ < 3—х років, а саме термін окупності рівний 0,52 роки, то фінансування даної наукової розробки є доцільним.

ВИСНОВКИ

В результаті виконання магістерської кваліфікаційної роботи було розв’язано задачу розробки інтелектуальної інформаційної технології та програмного забезпечення розпізнавання зображень тварин згортковою нейронною мережею.

В першому розділі було проведено аналітичний огляд відомих методів і засобів розпізнавання зображень тварин, виявлені їх можливості, переваги і недоліки. Обґрунтована перспективність застосування методів розпізнавання зображень тварин на основі згорткових нейронних мереж. Було наведено класифікацію методів розпізнавання зображень тварин, детально описано засоби розпізнавання тварин на основі традиційних підходів. Було проаналізовано недоліки розглянутих засобів розпізнавання тварин та запропоновано використовувати для побудови засобів розпізанавання тварин, згорткові нейронні мережі, оскільки вони мають переваги.

У другому розділі було обґрунтовано вибір типу нейронної мережі для задачі розпізнавання зображень – глибинна мережа переконань. Була розроблена структура та проаналізована математична модель глибинної мережі переконань, обгрунтовано вибір функції активації нейронів. Була проаналізована структура даних програми, а також розроблено алгоритм роботи програми та структуру програмного забезпечення розпізнавання зображень тварин.

У третьому розділі було обґрунтовано вибір середовища розробки, мови програмування та технологій, 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 Accord.NET. Було проведено реалізацію програмних модулів системи, включно з модулем тренування мережі, виведення графіку помилки навчання та класифікації зображень.

Також у даному 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 глибинна мережа переконань Accord.Net Network тренувалась біля 8 годин та досягла похибки в 17% (достовірність 83%). Таким чином, можна зробити висновок, що глибинна мережа переконань Accord.Net Network має порівняно зі пірамідальною нейронною мережею зменшену похибку при тоднаковому часі навчання. Тобто мета роботи досягнута – швидкість навчання підвищена та збільшена точність.

У четвертому розділі було проведено економічне обґрунтування доцільності розробки програми для розпізнавання зображень тварин. Економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 1402383 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,52 роки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кучевський О.В., Азаров О. Д. Розробка програмного модуля для розпізнавання зображень тварин. [Електронний ресурс]. – Режим доступу до ресурса: https://conferences.vntu.edu.ua/index.php/all—fitki/all—fitki—21/paper/view/
12692/. – Назва з екрану.

2. Object Detection [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Object_detection..

3. Tang, Y.; Zhang, C.; Gu, R. Vehicle detection and recognition for intelligent traffic surveillance system. Multimed. Tools Appl. 2017, 76, 5817–5832.

4. Maenpaa T. The Local Binary Pattern Approach to Texture Analysis — Extensions and Applications. Oulu University Press, 2003.

5. Azure Congnitive Services [Електронний ресурс] – Режим доступу до ресурсу: https://azure.microsoft.com/en—us/services/cognitive—services/.

6. Amazon Recognition [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/rekognition/.

7. Machine learning for mobile developers (Object Detection) [Електронний ресурс]. URL: https://developers.google.com/ml—kit

8. Carandini M. Area V1 [Електронний ресурс]. URL: http://www:scholarpedia:org/article/Area_V

9. Dougherty G. / Digital Image Processing for Medical Applications / G. Dougherty – Cambridge University Press. – ISBN 978—0—5218—6085—7. 9. Pan, C.; Sun, M.; Yan, Z. The Study on Vehicle Detection Based on DPM

10.Ren S. Faster R—CNN: Towards Real—Time Object Detection with Region Proposal Networks [Електронний ресурс]. URL: https://arxiv.org/pdf/1506.01497

11.Girshick R. Fast R—CNN [Електронний ресурс]. URL:https://www.cvfoundation.org/openaccess/content_iccv_2015/papers/Girshick_Fast_RCNN_ICCV_2015_paper.pdf

12.Redmon J. You Only Look Once: Unified, Real—Time Object Detection [Електронний ресурс]. URL:https://pjreddie.com/media/files/papers/yolo.pdf

13.Liu W. SSD: Single Shot MultiBox Detector [Електронний ресурс]. URL:https://www.cs.unc.edu/~wliu/papers/ssd.pdf

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

15.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.

16.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/.

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

18.Yosinski J. How transferable are features in deep neural networks? [Електронний ресурс]. URL: https://arxiv.org/pdf/1411.1792.pdf (дата обращения: 15.11.2020).

19..ML.NET – Machine Learning made for .NET [Електронний ресурс] – Режим доступу: https://dotnet.microsoft.com/apps/machinelearning—ai/ml—dotnet

20.OpenCV [Електронний ресурс] – режим доступе: https://uk.wikipedia.org/wiki/OpenCV.

21. [Електронний ресурс]. URL:https://keras.io/

22.[Электронный ресурс]. URL:https://www.tensorflow.org/

23.Методичні вказівки до виконання студентами—магістрантами економічної частини магістерських кваліфікаційних робіт / Уклад. В. О. Козловський – Вінниця: ВНТУ, 2012. – 22 с.

ДОДАТОК А

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

_____проф., д.т.н. О. Д. Азаров

«___» _____ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«Удосконалена система нейромережевого розпізнавання зображень»

08—23.МКР.023.00.000 ТЗ

Науковий керівник: проф. каф ОТ

_____ Азаров О.Д.

Магістрант групи 2КІ—21м

_____ Кучевський О. В.

Вінниця 2022

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність дослідження пов’язана з необхідністю підвищення об’єктивності та зменшення упередженості під час оцінювання знань. Об’єктивність найкраще досягається під час інтерактивного тестування з можливістю автоматизованої генерації тесту без повторень запитань, можливістю встановлювати складність завдання та можливістю контролювати співвідношення теоретичних та практичних запитань.

1.2 Наказ про затвердження теми магістерської кваліфікаційної роботи.

2 Мета і призначення МКР

2.1 Мета магістерської роботи полягає у покращенні алгоритму автоматизованої генерації тесту.

2.2 Призначення розробки — виконання магістерської кваліфікаційної роботи.

3 Вихідні дані для виконання МКР

Виконати розробку програмного модуля інтерактивного тестування знань. Схему взаємодії модулів, діаграми активності інтерактивного тестування, діаграми активності модулів та лістинги програми представити в додатках до роботи.

4 Вимоги до виконання МКР

МКР повинна задовольняти такі вимоги:

— запропонувати програмний модуль інтерактивного тестування знань;

— розробити алгоритм та структуру для програмного модуля інтерактивного тестування знань;

— розробити функціонал для створення тестів, тестових завдань та перегляду результатів тестування користувачів;

— забезпечення зручного та легкого для користувача графічного інтерфейсу;

— результат роботи, а саме веб—додаток інтерактивного тестування.

5 Етапи МКР

Етапи МКР та очікувані результати відображені у таблиці А.1

Таблиця А.1 — Етапи виконання роботи

№	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз завдання. Вступ	01.09.21	02.09.21	Вступ
2	Аналіз літературних джерел	03.09.21	09.09.21	Розділ 1
3	Розробка технічного завдання	10.09.2021	24.09.21	Технічне завдання
3	Розробка структури удосконаленого модуля для визначення рівня професійних знань	25.09.21	08.10.21	Розділ 2, розробка структури
4	Розробка веб—додатку, проектування веб—додатку	11.10.21	29.10.21	Розділ 3, розробка програми
5	Практична реалізація, результати.	01.11.21	14.11.21	Розділ 3
6	Розробка економічної частини	15.11.21	30.11.21	Розділ 4
7	Оформлення пояснювальної записки	02.12.21	15.12.21	ПЗ, презентація

6 Матеріали, що подаються до захисту МКР

До захисту МКР подаються: пояснювальна записка МКР, ілюстративні та графічні матеріали, протокол попереднього захисту МКР на кафедрі, відзив наукового керівника, відзив рецензента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів розрахункової та графічної документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8.1 При оформлювання МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104—2006 «Едина система конструкторської документації. Основні написи»;

— Методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія». Кафедра обчислювальної техніки ВНТУ 2022;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ—03.02.02—П.001.01:21».

ДОДАТОК Б

Схема алгоритму роботи головного компонента програми

Рисунок В.1 — Схема алгоритму роботи головного компонента програми

ДОДАТОК В

Лістинг програми

MainWindow.xaml

```
<Controls:MetroWindow x:Class="CatImageRecognizer.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:Controls="clr—
namespace:MahApps.Metro.Controls;assembly=MahApps.Metro"
        xmlns:mc="http://schemas.openxmlformats.org/markup—
compatibility/2006"
        xmlns:localControls="clr—
namespace:CatImageRecognizer.UserControls"
        xmlns:local="clr—namespace:CatImageRecognizer"
WindowStartupLocation="CenterScreen"
        mc:Ignorable="d"
        Title="Розпізнавання котиків" Height="450" Width="800"
ResizeMode="CanMinimize">

    <Controls:MetroWindow.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="/Resources/Icons.xaml" />
            </ResourceDictionary.MergedDictionaries>
            <local:BooleanInverterConverter
x:Key="booleanInverter"></local:BooleanInverterConverter>
            <local:EmgCVImageToBitmapImage
x:Key="cvImageToBitmapImage"></local:EmgCVImageToBitmapImage>
```

```
            <local:BoolToVisibilityConverter
x:Key="boolVisibilityConverter"></local:BoolToVisibilityConverter>
        </ResourceDictionary>
    </Controls:MetroWindow.Resources>

    <Controls:MetroWindow.RightWindowCommands>
        <Controls:WindowCommands>
            <Button Content="Тренування і тестування" Command="{Binding
OpenTrainWindowCommand}"></Button>
            <Button IsEnabled="{Binding IsSavingNetwork,
Converter={StaticResource booleanInverter}}" Content="Зберегти модель"
Command="{Binding SaveNeuralNetwork}"></Button>
            <Button IsEnabled="{Binding IsLoadingNetwork,
Converter={StaticResource booleanInverter}}" Content="Завантажити модель"
Command="{Binding LoadNeuralNetwork}"></Button>
        </Controls:WindowCommands>
    </Controls:MetroWindow.RightWindowCommands>

    <Grid>
        <Image Visibility="{Binding ModelNotLoaded,
Converter={StaticResource boolVisibilityConverter}}" Source="SplashScreen1.jpg"
Stretch="Fill" Margin="0,0,—0.4,0"/>
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition/>
                <RowDefinition Height="Auto"/>
            </Grid.RowDefinitions>

            <Grid Grid.Row="0">
```

```
                <localControls:DetectorArea HorizontalAlignment="Stretch"
VerticalAlignment="Stretch" DataContext="{Binding
DetectorViewModel}"></localControls:DetectorArea>
            </Grid>

            <Grid Grid.Row="1" Background="{StaticResource
AccentColorBrush}">
                <TextBlock Foreground="{StaticResource
IdealForegroundColorBrush}" Margin="5" Text="{Binding
StatusMessage}"></TextBlock>
            </Grid>
        </Grid>
    </Grid>

</Controls:MetroWindow>
```

AccordNetwork.cs

```
using Accord.Neuro;
using Accord.Neuro.ActivationFunctions;
using Accord.Neuro.Learning;
using Accord.Neuro.Networks;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CatImageRecognizer.NeuralNetworks
{
    public static class Randomizer
```

```
{
    private static Random rng = new Random();
    public static void Shuffle<T>(IList<T> list)
    {
        int n = list.Count;
        while (n > 1)
        {
            n——;
            int k = rng.Next(n + 1);
            T value = list[k];
            list[k] = list[n];
            list[n] = value;
        }
    }

    public static void Shuffle(double[][] values1, double[][] values2)
    {
        int n = values1.Length;
        while (n > 1)
        {
            n——;
            int k = rng.Next(n + 1);
            double[] value1 = values1[k];
            values1[k] = values1[n];
            values1[n] = value1;
            double[] value2 = values2[k];
            values2[k] = values2[n];
            values2[n] = value2;
        }
    }
```

```
}

public class AccordNetwork : INeuralNetwork
{
   private int inputLength = 2600;
   DeepBeliefNetwork network;
   DeepBeliefNetworkLearning unsuperVisedTeacher;
   ResilientBackpropagationLearning supervisedTeacher;

   private int batchSubSize = 500;
   private int maxIterationsOnEachPage = 1000;
   private int maxUnSupervisedIterationsOnEachPage = 100;
   private double stopPageTrainingErrorRate = 0.0001;
   private double stopIterationTrainingErrorRate = 0.001;

   public AccordNetwork()
   {
      network = new DeepBeliefNetwork(new BernoulliFunction(), inputLength, 1200, 600, 2);
      new NguyenWidrow(network).Randomize();
      network.UpdateVisibleWeights();
      unsuperVisedTeacher = GetUnsupervisedTeacherForNetwork(network);
      supervisedTeacher = GetSupervisedTeacherForNetwork(network);
   }

   public void BatchTrain(double[][] batchInputs, double[][] batchOutputs, int iterations, Action<double, int, string> progressCallback)
   {
      Randomizer.Shuffle(batchInputs, batchOutputs);
```

```
            TrainUnsupervised(batchInputs,
maxUnSupervisedIterationsOnEachPage, progressCallback);
            TrainSupervised(batchInputs, batchOutputs, iterations,
progressCallback);
        }

        private void TrainUnsupervised(double[][] batchInputs, int iterations,
Action<double, int, string> progressCallback)
        {
            for (int layerIndex = 0; layerIndex < network.Machines.Count — 1;
layerIndex++)
            {
                unsuperVisedTeacher.LayerIndex = layerIndex;
                var layerData = unsuperVisedTeacher.GetLayerInput(batchInputs);
                foreach (int i in Enumerable.Range(1, iterations))
                {
                    var error = unsuperVisedTeacher.RunEpoch(layerData) /
batchInputs.Length;
                    if (progressCallback != null)
                    {
                        progressCallback(error, i, $"Безконтрольний шар
{layerIndex}");
                    }
                    if (this.ShouldStopTraning)
                    {
                        this.ShouldStopTraning = false;
                        break;
                    }
                }
            }
```

```
        }

        private void TrainSupervised(double[][] batchInputs, double[][] batchOutputs, int iterations, Action<double, int, string> progressCallback)
        {
            foreach (int i in Enumerable.Range(1, iterations))
            {
                var error = supervisedTeacher.RunEpoch(batchInputs, batchOutputs) / batchInputs.Length;
                if (progressCallback != null)
                {
                    progressCallback(error, i, "Підконтрольний");
                }
                if (this.ShouldStopTraning)
                {
                    this.ShouldStopTraning = false;
                    break;
                }
            }
        }

        private DeepBeliefNetworkLearning GetUnsupervisedTeacherForNetwork(DeepBeliefNetwork deepNetwork)
        {
            var teacher = new DeepBeliefNetworkLearning(deepNetwork)
            {
                Algorithm = (hiddenLayer, visibleLayer, i) => new ContrastiveDivergenceLearning(hiddenLayer, visibleLayer)
                {
                    LearningRate = 0.1,
```

```
            Momentum = 0.5
        }
    };
    return teacher;
}

private ResilientBackpropagationLearning GetSupervisedTeacherForNetwork(DeepBeliefNetwork deepNetwork)
{
    var teacher = new ResilientBackpropagationLearning(deepNetwork)
    {
        LearningRate = 0.1
        //Momentum = 0.5
    };
    return teacher;
}


public double[] GenerateOutput(double[] inputs)
{
    return network.Compute(inputs);
}

public int GetBatchSubSize()
{
    return batchSubSize;
}

public int GetInputLength()
{
```

```
            return inputLength;
        }

        public string GetNetworkName()
        {
            return "Accord.NET";
        }

        public void LoadNetworkFromFile(string filePath)
        {
            network = DeepBeliefNetwork.Load(filePath);
            supervisedTeacher = GetSupervisedTeacherForNetwork(network);
            unsuperVisedTeacher                                                    =
GetUnsupervisedTeacherForNetwork(network);
        }

        public void SaveNetwork(string filePath)
        {
            network.Save(filePath);
        }

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

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

        public int GetMaxIterationsOnEachPage()
        {
```

```
            return maxIterationsOnEachPage;
        }

        public double GetStopPageTrainingErrorRate()
        {
            return stopPageTrainingErrorRate;
        }

        public double GetStopIterationTrainingErrorRate()
        {
            return stopIterationTrainingErrorRate;
        }

        public string GetModelExtension()
        {
            return "accmodel";
        }
    }
}
```

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 Г

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Удосконалена система нейромережевого розпізнавання зображень

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності
Unicheck

Оригінальність 87.8% Схожість 12.2%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

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

Автор роботи _____ Кучевський О.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Азаров О.Д.
(підпис) (прізвище, ініціали)