

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА
Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту

" _____ " _____ 2023 р.

Зав. кафедри, к.т.н., доцент

_____ Ліщук Р.І.

КВАЛІФІКАЦІЙНА РОБОТА
МАГІСТРА

Інформаційна система розпізнавання зображень на основі
згорткової нейронної мережі

Виконав – студент 2 курсу, групи 21м-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки

Янкова Анастасія Сергіївна

Науковий керівник – д.т.н., професор

Кучерук Володимир Юрійович

Умань – 2023

Реферат

на кваліфікаційну роботу «Інформаційна система розпізнавання зображень на основі згорткової нейронної мережі»

студента факультету економіки та підприємництва
Уманського національного університету садівництва

Янкової Анастасії Сергіївни

Структура та обсяг роботи - пояснювальна записка до кваліфікаційної роботи магістра складається з 3 розділів, містить 59 сторінок, 23 рисунка, 2 додатка, 40 літературних джерел.

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

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. Наведено класифікацію методів розпізнавання зображень тварин, детально описано засоби розпізнавання тварин на основі традиційних підходів, проаналізовано недоліки розглянутих засобів розпізнавання тварин та запропоновано використовувати для побудови засобів розпізнавання тварин, згорткові нейронні мережі, оскільки вони мають переваги.

Обґрунтовано вибір типу нейронної мережі для задачі розпізнавання зображень – глибинна мережа переконань. Розроблена структура та проаналізована математична модель глибинної мережі переконань, обґрунтовано вибір функції активації нейронів. Проаналізована структура даних програми, а також розроблено алгоритм роботи програми та структуру програмного забезпечення розпізнавання зображень тварин.

Обґрунтовано вибір середовища розробки, мови програмування та технологій, які будуть використовуватися при розробці програми та наведено їх основні переваги. Обрано середовище розробки Visual Studio Code, мову програмування C# та бібліотеку Accord.NET. Проведено реалізацію програмних

модулів системи, включно з модулем тренування мережі, виведення графіку помилки навчання та класифікації зображень.

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, глибинна мережа переконань Accord.Net Network має порівняно зі пірамідальною нейронною мережею зменшену похибку при однаковому часі навчання.

Анотація

У результаті виконання кваліфікаційної роботи магістра 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.

Annotation

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.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА

Факультет економіки і підприємництва
Кафедра Інформаційних технологій
Освітня програма Комп'ютерні науки
Спеціальність 122 Комп'ютерні науки
(код, назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Ліщук Р.І.
(підпис) (ініціали, прізвище)
" " 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студенту Янковій Анастасії Сергіївні
(прізвище, ім'я, по батькові)

1. Тема проекту Інформаційна система розпізнавання зображень на основі згорткової нейронної мережі
керівник роботи Кучерук Володимир Юрійович, д.т.н., професор кафедри інформаційних технологій,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "12" червня 2023 року № 01-06/130

2. Строк подання студентом роботи 8 пункт, 28.05.2023

3. Вихідні дані до роботи 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 текстової вибірки – не менше 100, середовище програмування – об'єктно-орієнтоване.

4. Перелік питань, які потрібно розробити
Аналітичний огляд предметної області, алгоритмічне та програмне забезпечення засобу, тестування застосунку

5. Перелік графічного матеріалу 3 таблиці, 23 рисунка, 2 додатка, 40 джерел

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 07.09.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1.	Вибір теми кваліфікаційної роботи	05.09.2022	
2.	Одержання індивідуального завдання	07.09.2022	
3.	Складання календарного плану написання кваліфікаційної роботи	08.09.2022	
4.	Підготовка до виконання кваліфікаційної роботи - підбір та вивчення літератури - участь у виконанні науково-дослідних робіт - виконання курсових робіт - інші заходи	Впродовж I семестру	
5.	Уточнення теми кваліфікаційної роботи та календарного плану-графіка, виходячи зі специфіки базового підприємства, установи	До початку II семестру	
6.	Підготовка: - першого розділу - другого розділу - третього розділу - висновків	09.02.2023 23.03.2023 26.04.2023 10.05.2023	
7.	Доопрацювання кваліфікаційної роботи з урахуванням зауважень наукового керівника	18.05.2023	
8.	Написання та оформлення роботи в остаточному варіанті	25.05.2023	
9.	Перевірка роботи на академічний плагіат	12.06.2023	
10.	Одержання відгуку наукового керівника	13.06.2023	
11.	Попередній захист роботи на кафедрі	14.06.2023	
12.	Затвердження роботи завідувачем кафедри	15.06.2023	
13.	Одержання рецензії	16.06.2023	
14.	Захист кваліфікаційної роботи	19-21.06.2023	

Студент _____ Янкова А.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Кучерук В.Ю.
(підпис) (прізвище та ініціали)

ЗМІСТ

Перелік умовних позначень, скорочень і термінів	8
ВСТУП	9
РОЗДІЛ1. АНАЛІЗ ПРЕДМЕТНОЇ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН	12
1.1 Класифікація існуючих методів розпізнавання зображень тварин	12
1.2. Існуючі проблеми в задачі розпізнавання зображень тварин	15
1.3. Постановка задачі	15
РОЗДІЛ 2. РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН НА ОСНОВІ НЕЙРОМЕРЕЖ	17
2.1. Обґрунтування вибору типу нейронної мережі	17
2.2. Структура та математична модель глибинної мережі переконань	22
2.3. Розробка структури нейромережевої системи	27
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОМЕРЕЖ	32
3.1. Обґрунтування вибору середовища програмування	32
3.2. Обґрунтування вибору мови програмування	32
3.3. Використання фреймворку Accord.Net	33
ВИСНОВКИ	42
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	44
ДОДАТОК А. Схема алгоритму роботи головного компонента програми	48
ДОДАТОК Б. Лістинг програми	49

Перелік умовних позначень, скорочень і термінів

HOG	histogram of oriented gradients
RBF	Radial basis function
ЗНМ	Згорткова нейронна мережа
ГМП	Глибинна мережа переконань
DBN	Deep Belief Networks
КД	Контрастивна дивергенція
LIFO	Last In First Out

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;
- обґрунтувати 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 полягає в подальшому розвитку інформаційної технологія розпізнавання зображень тварин, яка відрізняється використанням глибинної 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.

РОЗДІЛ 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.

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 рис. 1.1.

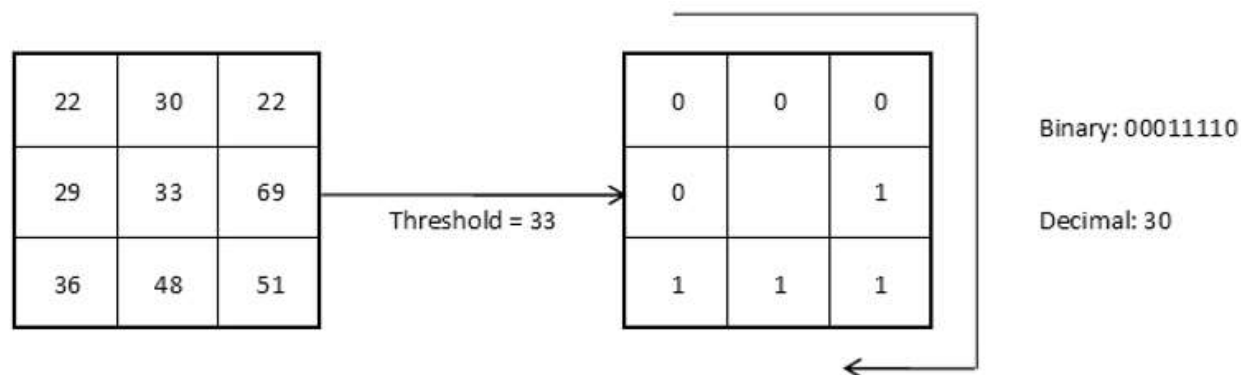


Рисунок 1.1 - Базовий оператор локального бінарного шаблону

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 рис. 1.2.

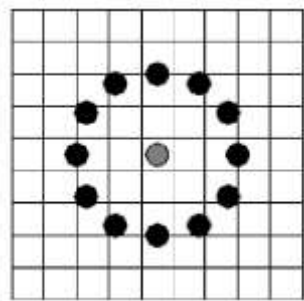


Рисунок 1.2 - Розширений оператор локального бінарного шаблону

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 рис. 1.3

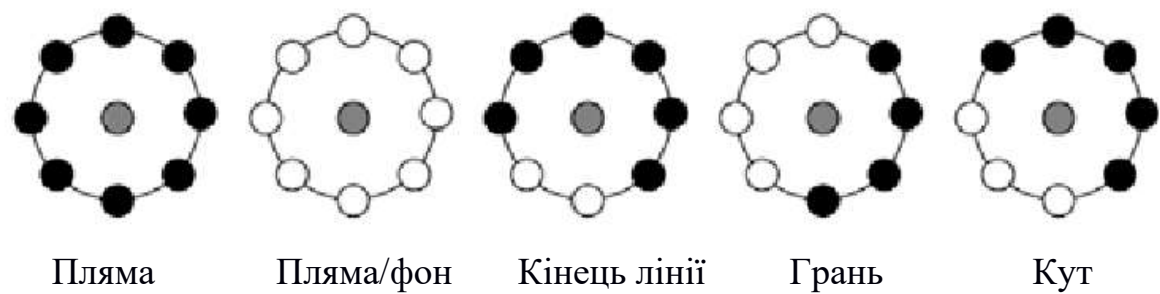


Рисунок 1.3 - Приклад локальних ознак, які детектуються локальними бінарними шаблонами

Подальший розвиток ідея вилучення ознак із зображення отримала в алгоритмі гістограми орієнтованих градієнтів (HOG) [5]. Основна ідея

алгоритму гистограми 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 гистограми напрямків градієнтів або напрямків країв для пікселів, що знаходяться всередині осередку. Комбінація цих гистограм і є дескриптором. Для збільшення точності локальні гистограми 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 інваріантність по відношенню до висвітлення.

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

Переваги методу орієнтованих градієнтів:

- оскільки HOG працює локально, метод підтримує інваріантність 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 тощо).
- об'єкти можуть перекривати один одного.

Алгоритми, розглянуті раніше, можуть виділяти тільки дуже низькорівневі ознаки, такі, як межі, кути, плями тощо, 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. Постановка задачі

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 побудований на використанні пірамідальної нейронної мережі, яка працює із попередньо отриманими ознаками зображення. Достовірність розпізнавання зображень тварин у Google ML Kit становить в середньому 72 %.

РОЗДІЛ 2. РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ТВАРИН НА ОСНОВІ НЕЙРОМЕРЕЖ

2.1. Обґрунтування 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 ідентифікації тварин в інших зображеннях. Вони роблять це без жодного апіорного 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].

Обґрунтуємо вибору типу нейронної мережі для задачі розпізнавання зображень тварин.

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].

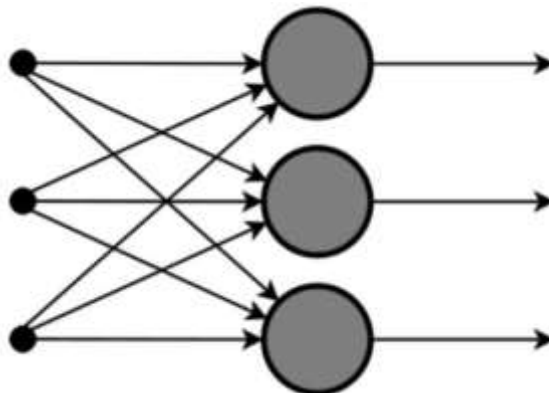


Рисунок 2.1 - Нейронна мережа прямого поширення

Мережа радіальних базисних функцій (Radial Basis Function Neural Network) - 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 Гауса використовується як функція передачі даних в обчислювальних одиницях. Залежно від конкретного випадку, як правило, спостерігається, що мережі RBF потрібно менше часу для завершення навчання.

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].

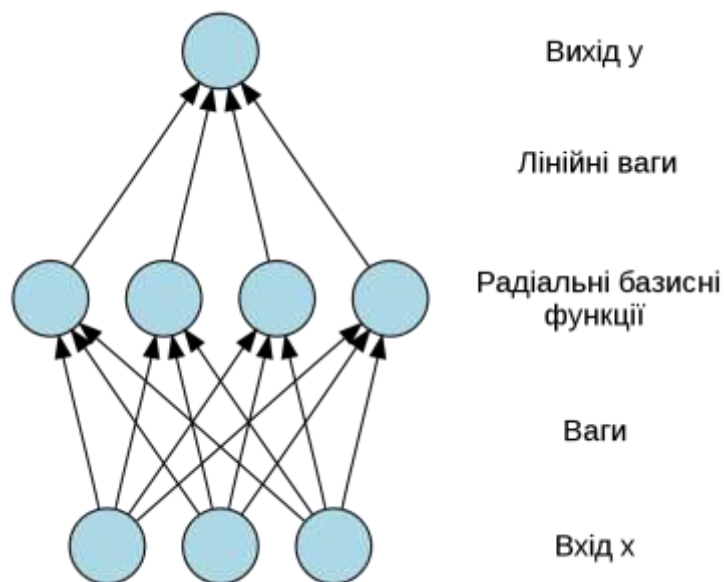


Рисунок 2.2 - Нейронна мережа радіальних базисних функцій

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 (субдискретизації), 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 рис. 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].

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 ймовірно 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 рис. 2.4. Стрілки представляють спрямовані з'єднання у графовій моделі, яку представляє мережа.

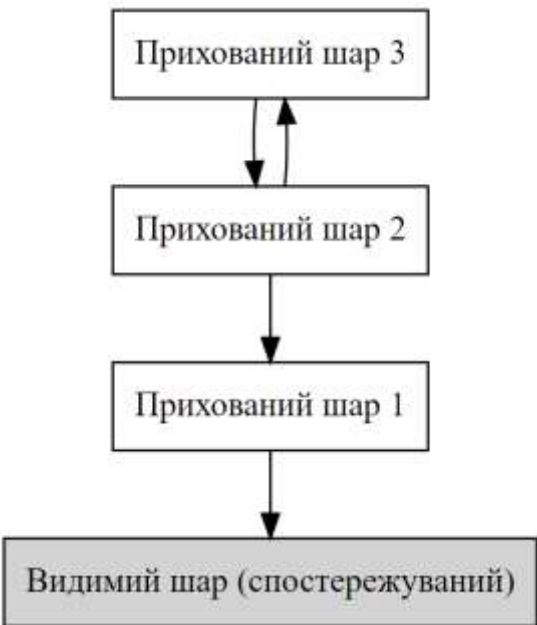


Рисунок 2.4 - Схематичне представлення глибинної мережі переконань

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 ймовірно 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.

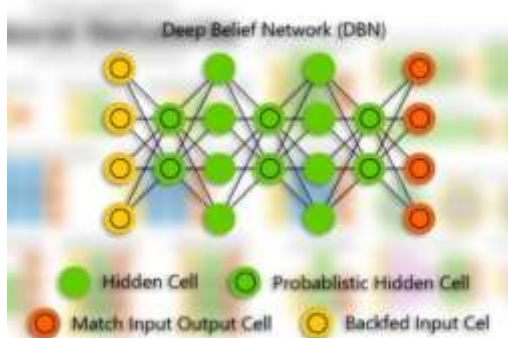


Рисунок 2.5 - Графічне зображення глибинної мережі переконань

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 RBM оновлення ваги виконується з градієнтним спуском за наступним рівнянням:

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

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

$$p(v) = \frac{1}{Z} \sum_h \exp(-E(v, h)), \quad (2.2)$$

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 "бажаний" конфігурації. Градієнт $\frac{\delta \log(p(v))}{\delta \omega_{ij}}$ має просту форму

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

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 ініціалізуються видимі одиниці до навчального вектора.

Після цього паралельно оновлюються приховані одиниці, враховуючи видимі одиниці:

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

де σ - сигмоїдна функція і b_j є упередженість h_j .

Паралельно оновлюються видимі одиниці, враховуючи приховані одиниці:

$$p(v_j = 1|H) = \sigma(a_j + \sum_i h_i \omega_{ij}), \quad (2.5)$$

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:

$$\Delta \omega_{ij} \propto \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{reconstruction}, \quad (2.6)$$

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].

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).

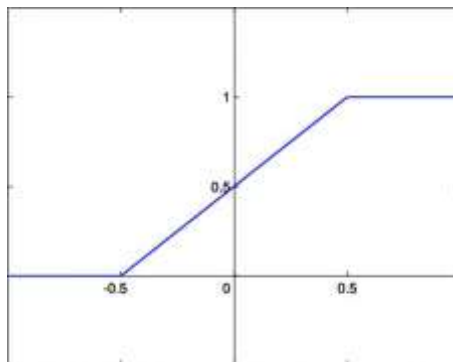


Рисунок 2.6 - Графік лінійної функції активації

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.

$$A = \frac{1}{1 + \exp(-x)}. \quad (2.9)$$

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

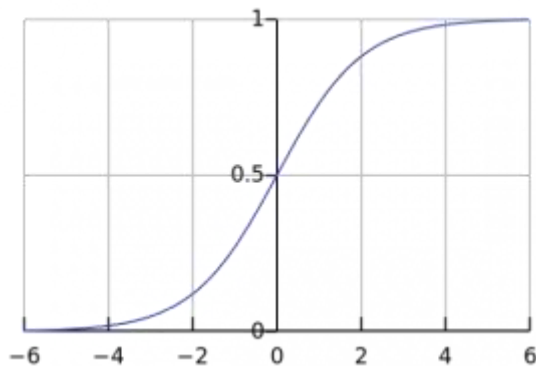


Рисунок 2.7 - Графік сигмоїдної функції активації

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 підходящою 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.

2.3. 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.

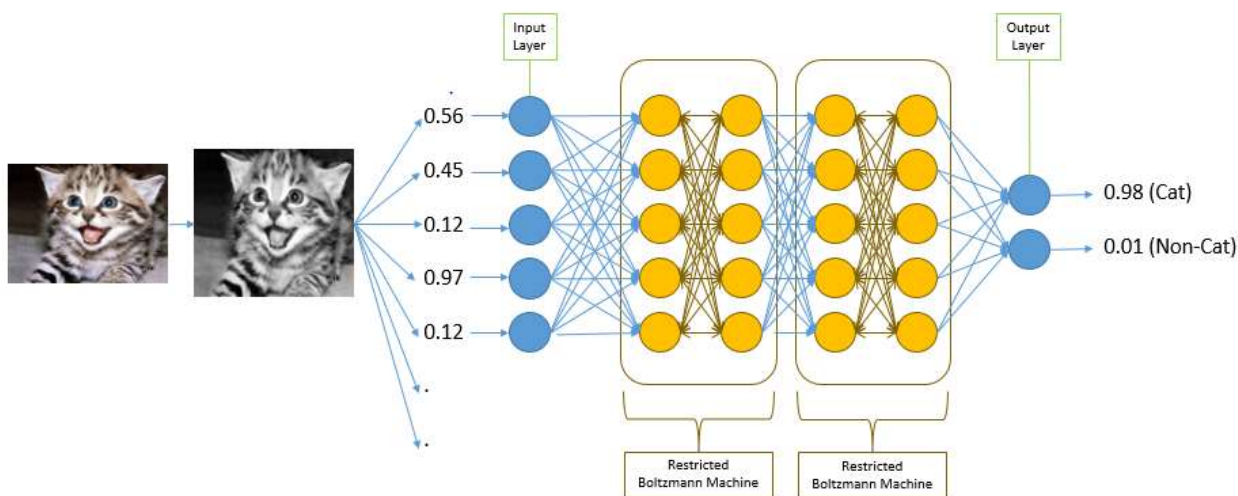


Рисунок 2.8 - Схема глибинної мережі переконань

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.

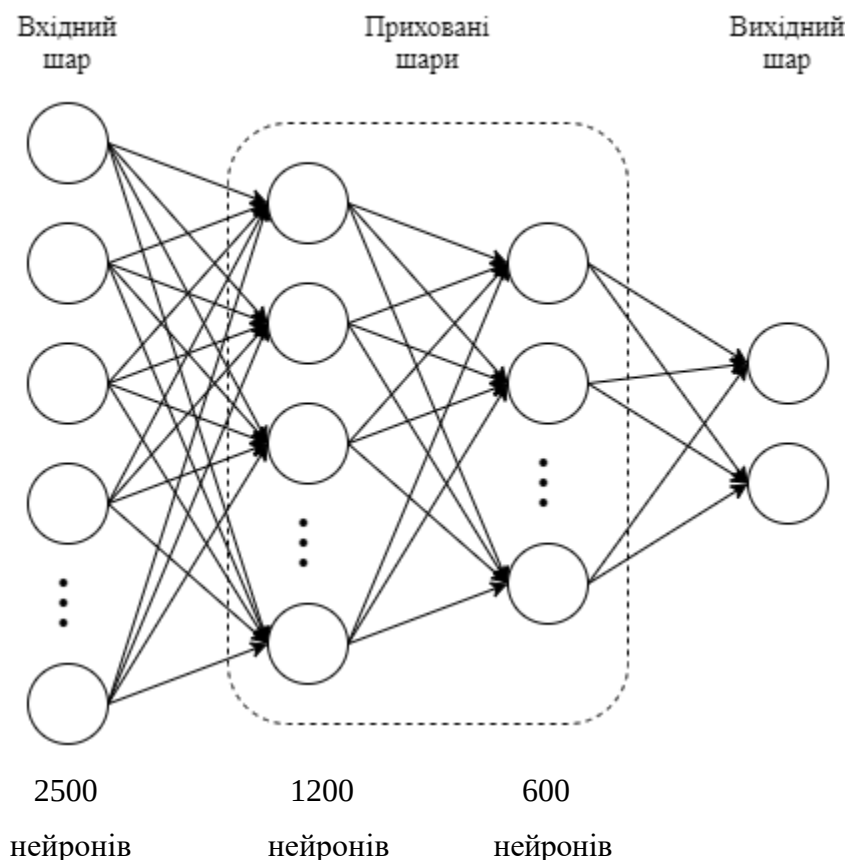


Рисунок 2.9 - Структура нейромережевої системи

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.

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 ціло-чисельному 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, на відміну 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.

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.

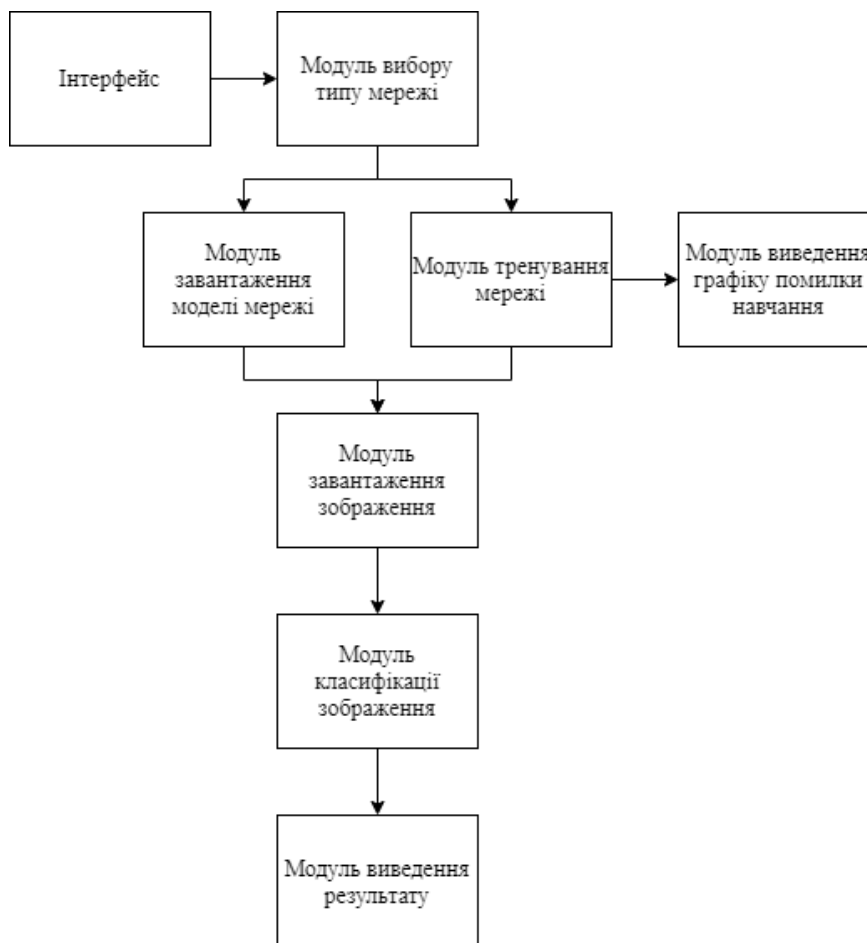


Рисунок 2.10 - Структура програмного забезпечення

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 входні 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 входному зображенні разом з анкерними коробками.

РОЗДІЛ 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 багатифункціональний 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 виберемо найкращу для

розробки (табл. 3.2).

Таблиця 3.2 - Порівняння мов програмування

Мова	Предбачуване використання	Імперативне	Об'єктно-орієнтоване	Функційне	Процедурне	Узагальнене	Рефлексивне
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].

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>
originalImage)
{
    var grayScale = originalImage.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;
        }
    }
}

```

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.

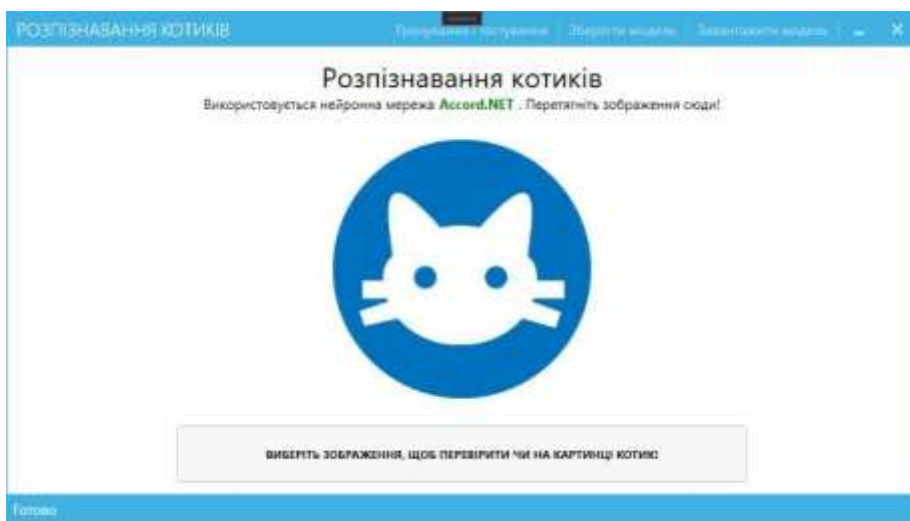


Рисунок 3.2 – Вікно розпізнавання

Далі потрібно вказати чи на цьому зображенні 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.

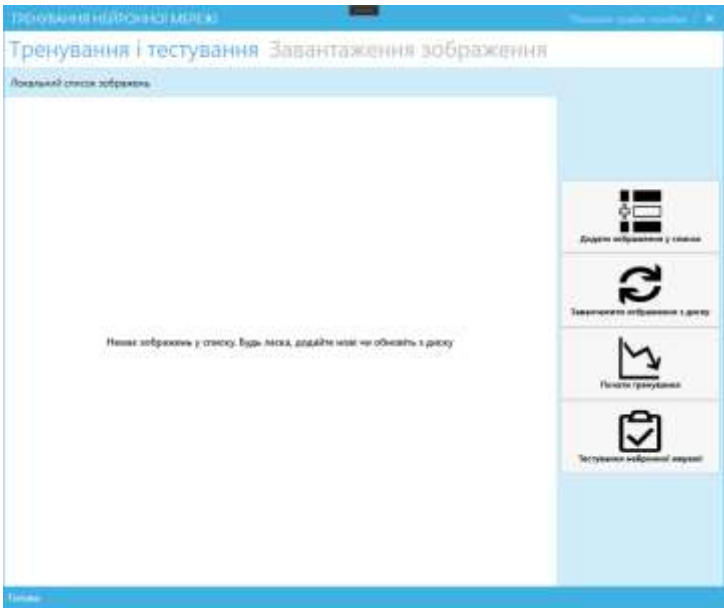


Рисунок 3.3 – Вікно «Тренування і тестування»

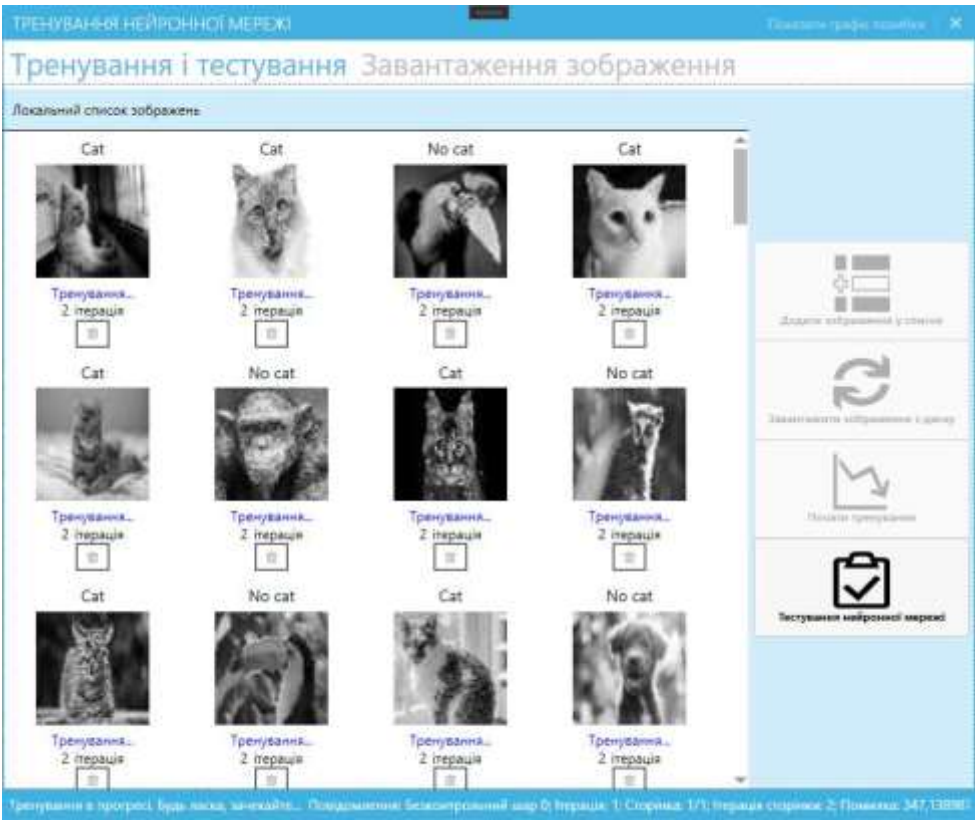


Рисунок 3.4 - Вікно «Тренування і тестування» із завантаженими зображеннями та початком тренування нейронної мережі

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

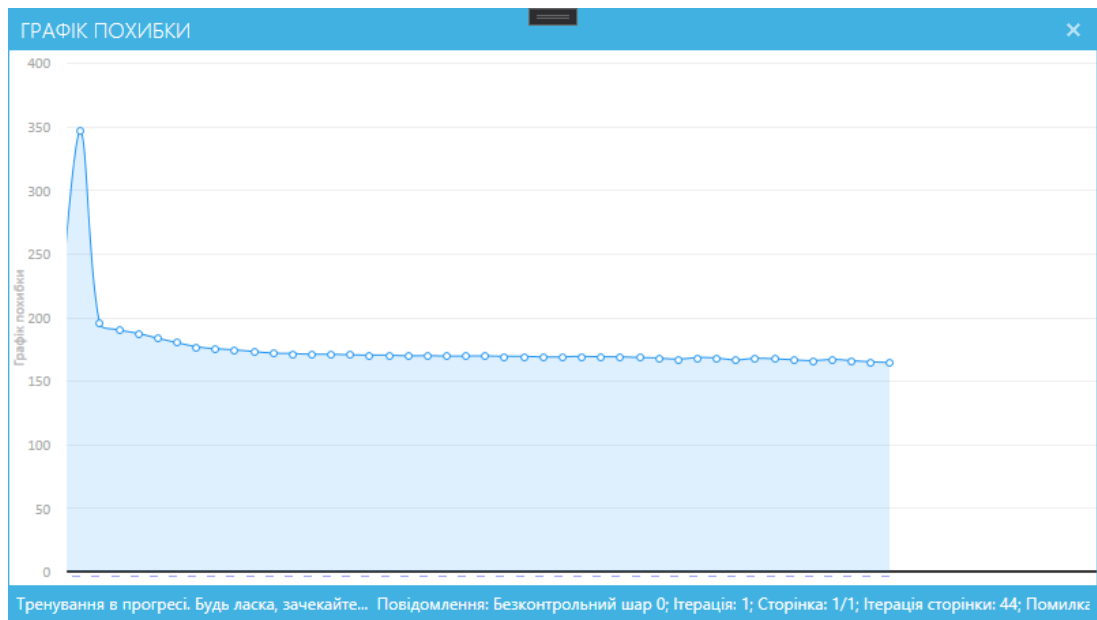


Рисунок 3.5 – Графік зменшення похибки навчання

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.

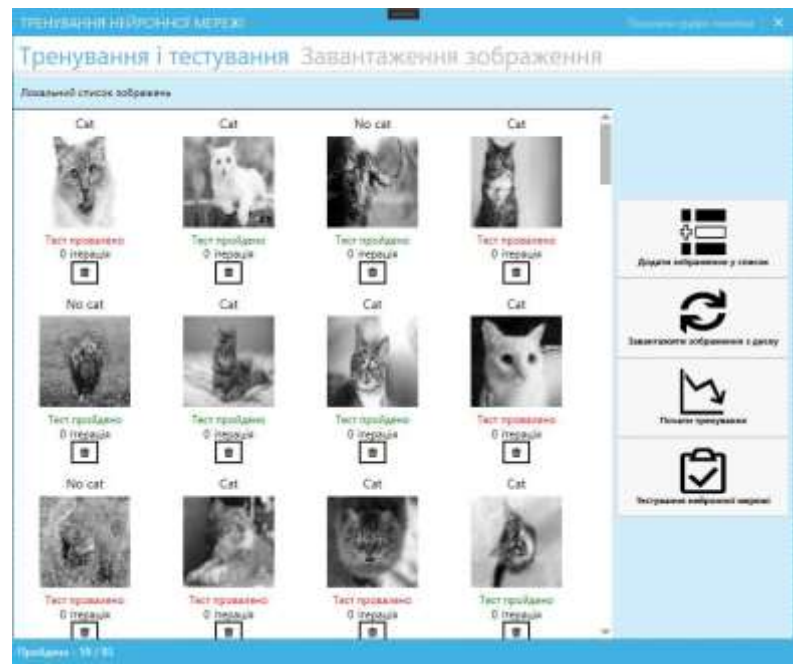


Рисунок 3.6 – Тестування навченої нейронної мережі

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

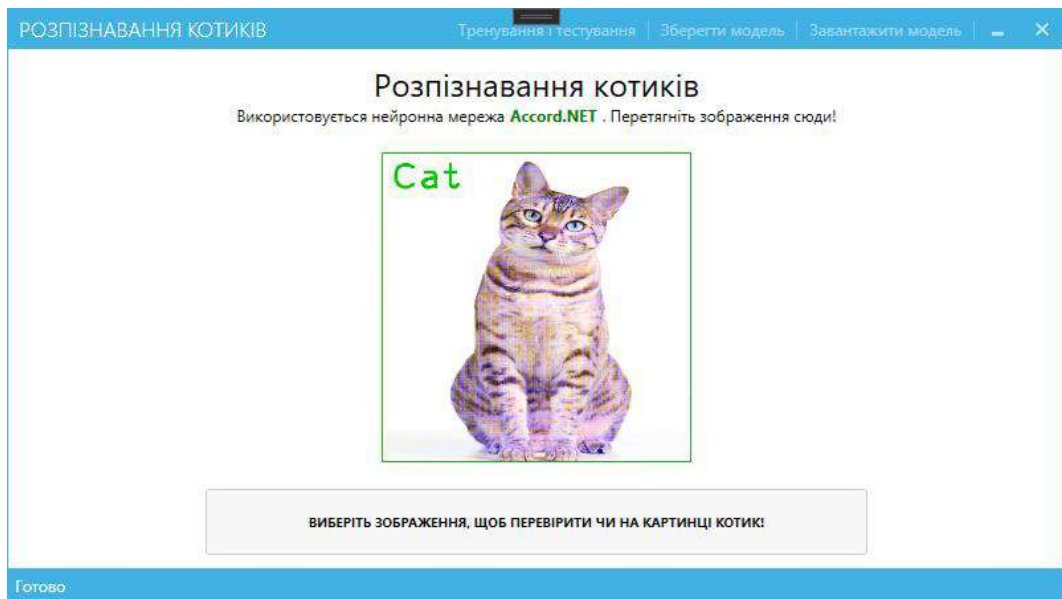


Рисунок 3.7 – Вікно розпізнавання із зображенням кота

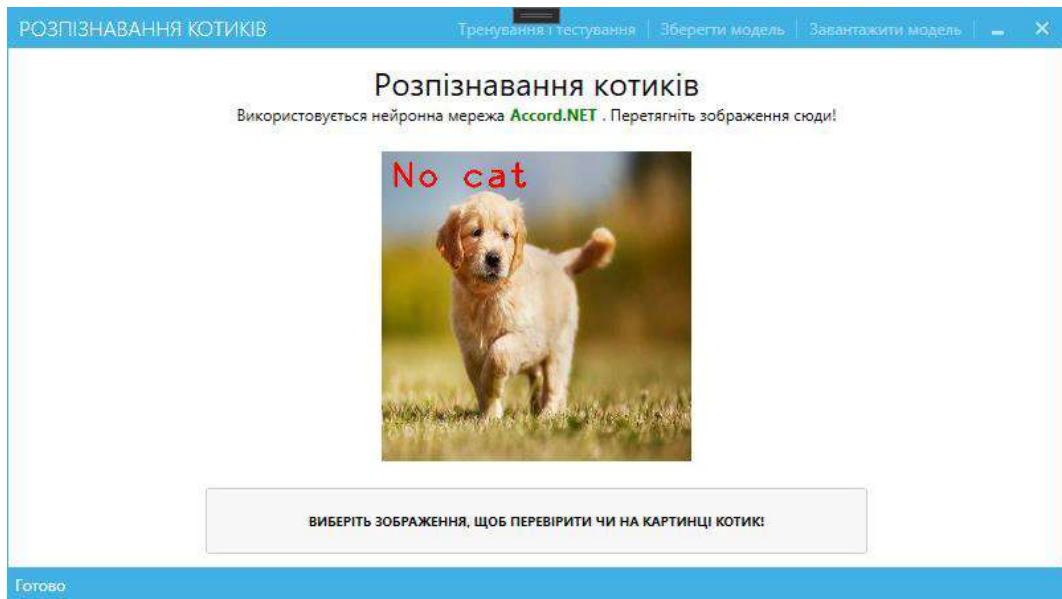


Рисунок 3.8 – Вікно розпізнавання із зображенням іншої тварини

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 розпізнаних 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 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.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи магістра 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 розпізнавання 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, обґрунтовано 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 однаковому часі навчання. Тобто мета роботи досягнута – швидкість навчання підвищена та збільшена точність.

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 Cognitive 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. Красовицький М. Найкращі IDE для програмування на C# [Електронний ресурс] / Максим Красовицький – 2022. – Режим доступу до ресурсу: <https://deveducation.com/blog/luchshie-ide-dlya-razrabotki-na-c/>.

17. Visual Studio Найкращі у своєму класі засоби для розробників [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>.

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 доступу: <https://uk.wikipedia.org/wiki/OpenCV>.

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

22. [Електронний ресурс]. URL:<https://www.tensorflow.org/>

23. Tony Yiu, Understanding Neural Networks [Електронний ресурс] / Towards Data Science. Режим доступу: <https://towardsdatascience.com/understanding-neural-networks-19020b758230>.

24. D. Bahdanau, K. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate [Електронний ресурс] / Arxiv Org. Режим доступу: <https://arxiv.org/pdf/1409.0473.pdf>.

25. Michael Nielsen, Neural Networks and Deep Learning [Електронний ресурс] / Neural Networks and Deep Learning. Режим доступу: <http://neuralnetworksanddeeplearning.com/index.html>.

26. S. Hochreiter and J. Schmidhuber, Long short-term memory. Neural computation [Електронний ресурс] / Bioinf At. Режим доступу:

<https://www.bioinf.jku.at/publications/older/2604.pdf>.

27. T. Tieleman and G. Hinton, Divide the gradient by a running average of its recent magnitude. Neural computation [Электронный ресурс] / Coursera. Режим доступа: https://www.cs.toronto.edu/lecture_slides_lec6.pdf.

28. L. Gaunt, M. Brockschmidt, A probabilistic programming language for program induction [Электронный ресурс] / Arxiv Org. Режим доступа: <https://arxiv.org/pdf/1612.00817.pdf>.

29. Goodfellow, J. Pouget-Abadie, Generative adversarial nets [Электронный ресурс] / Arxiv Org. Режим доступа: <https://arxiv.org/pdf/1406.2661.pdf>.

30. K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition [Электронный ресурс] / Arxiv Org. Режим доступа: <https://arxiv.org/pdf/1409.1556.pdf>.

31. Karpathy and L. Fei-Fei, Deep visual-semantic alignments for generating image descriptions [Электронный ресурс] / Stanford Edu. Режим доступа: <https://cs.stanford.edu/people/karpathy/cvpr2015.pdf>.

32. K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition [Электронный ресурс] / Arxiv Org. Режим доступа: <https://arxiv.org/pdf/1409.1556.pdf>.

33. W. Zaremba, I. Sutskever, Recurrent neural network regularization [Электронный ресурс] / Arxiv Org. Режим доступа: <https://arxiv.org/pdf/1409.2329.pdf>.

34. H. Zhang, T. Xu, Text to photo-realistic image synthesis with stacked generative adversarial networks [Электронный ресурс] / Arxiv Org. Режим доступа: <https://arxiv.org/abs/1612.03242>.

35. L. Gaunt, A probabilistic programming language for program induction [Электронный ресурс] / Arxiv Org. Режим доступа: <https://arxiv.org/abs/1608.04428>.

36. Eisenman, B. Learning React [Text] / Bonnie Eisenman. 2nd edition. - Sebastopol: O'Reilly Media, 2017. - 242 p.

37. Kirupa, Understanding WebViews [Электронный ресурс] / Kirupa. Режим доступа: <https://www.kirupa.com/apps/webview.htm>.

38. Forsyth D., Ponce J. Computer Vision: A Modern Approach.- New Jersey: Pearson Education, 2011.- 792 p.
39. Ткаченко Р.О. Модель нейронних мереж //Вісник Держ. ун-ту "Львівська політехніка": Комп'ютерна інженерія та інформаційні технології. – 1998. – № 349. – С.83–86.
40. Грицик В.В., Ткаченко Р.О. Нові підходи до навчання штучних нейромереж // Доповіді Національної Академії Наук України, 2002.- № 12. – С.59–64.

ДОДАТОК А

Схема алгоритму роботи головного компонента програми



Рисунок А.1 - Схема алгоритму роботи головного компонента програми

ДОДАТОК Б

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 EmguCVImageToBitmapImage : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter, CultureInfo
culture)
        {
            var emguCVImage = (Emgu.CV.Image)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.Image 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;
            }
        }

```