

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
РОЗПІЗНАВАННЯ ТРИВИМІРНИХ ОБ'ЄКТІВ НА ОСНОВІ ПОЛЯ
ОПТИЧНОГО ПОТОКУ

RESEARCH AND SOFTWARE IMPLEMENTATION OF THE METHODS FOR
3D OBJECT RECOGNITION BASED ON THE OPTICAL FLUX FIELD

Виконав: студент 2 курсу, групи КНТ-110м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Тімохін А.А.

(прізвище та ініціали)

Керівник Коцур М.І.

(прізвище та ініціали)

Рецензент Гофман Є.О.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти магістр

Спеціальність 121 Інженерія програмного забезпечення

(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

С.О. Субботін

“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Тімохіна Артура Анатолійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів розпізнавання тривимірних об'єктів на основі поля оптичного потоку. Research and Software Implementation of the Methods for 3D Object Recognition Based on the Optical Flux Field

керівник проєкту (роботи) Коцур Михайло Ігорович, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “05” листопада 2021 року № 435

2. Строк подання студентом проєкту (роботи) грудень 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблемної області та постановка завдань дослідження. 2. Матеріали та методи. 3. Розробка архітектури програми. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Коцур М.І., доцент		
Нормоконтроль	Липовець М.В., асистент		

7. Дата видачі завдання “4” вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й методів вирішення задачі.	4–5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7–8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Розробка організаційно-економічної частини	10 тиждень	Розділ 6
8	Розробка заходів з охорони праці та цивільної безпеки.	11 тиждень	Розділ 7
9	Оформлення пояснювальної записки та документів до неї.	12-13 тиждень	Додатки
10	Нормоконтроль та рецензування.	14–15 тижні	
11	Захист роботи.	16 тиждень	

Студент(ка)

(підпис)

Тюнін А.А.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Коцур М.І.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
92 с., 25 рис., 6 табл., 4 дод., 27 джерел.

РОЗПІЗНАВАННЯ ОБРАЗІВ, МАШИННЕ НАВЧАННЯ,
КОМП'ЮТЕРНИЙ ЗІР, ШТУЧНІ НЕЙРОННІ МЕРЕЖІ, ГІБРИДНІ
ТОПОЛОГІЇ, БЕЗПЛОТНІ ЛІТАЛЬНІ ПРИСТРОЇ, ЗОБРАЖЕННЯ.

Об'єкт дослідження – процес розпізнавання тривимірних об'єктів.

Предмет дослідження – методи розпізнавання тривимірних об'єктів.

Мета роботи – дослідити та розробити програмну реалізацію методів розпізнавання тривимірних об'єктів на основі поля оптичного потоку.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, технології машинного навчання з учителем, мова програмування C++, бібліотека Caffe, відеокарта Nvidia з підтримкою технології CUDA персональний комп'ютер під управлінням операційної системи Microsoft Windows 10.

Результати. Створено програмний застосунок призначений для розпізнавання тривимірних об'єктів на графічних з метою аналізу та дослідження графічних зображень для пошуку та розпізнавання складних тривимірних зображень.

Висновки. Розроблено програмну систему для розпізнавання образів на базі машинного навчання з використанням гібридних топологій та паралельного підходу.

Галузь використання – системи прийняття рішень в галузях логістики, військової справи та охорони довкілля.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 92 pages, 25 figures, 6 tables, 4 appendixes, 27 sources.

PATTERN RECOGNITION, MACHINE LEARNING, COMPUTER VISION, ARTIFICIAL NEURAL NETWORKS, HYBRID TOPOLOGIES, UNMANNED AERIAL VEHICLES, IMAGES.

The object of research is the process of recognizing three – dimensional objects.

The subject of the research is Methods for recognizing three – dimensional objects.

The aim of the work is to investigate and develop a software implementation of methods for recognizing three – dimensional objects based on the optical flux field.

Materials, methods and technical means: structural and Object-Oriented Programming, machine learning technologies with a teacher, C++ programming language, Caffe library, Nvidia video card with CUDA technology support personal computer running the Microsoft Windows 10 operating system.

Results. A software application has been created designed to recognize three-dimensional objects on graphic ones in order to analyze and study graphic images to search for and recognize complex three-dimensional images.

Conclusions. A software system for Pattern Recognition based on machine learning using hybrid topologies and a parallel approach has been developed.

The field of use is decision-making systems in logistics, military affairs and Environmental Protection.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	9
Вступ.....	10
1 Аналіз проблемної області та постановка завдань дослідження.....	12
1.1 Введення в МН	12
1.2 Введення в теорію РО.....	14
1.3 Введення в КЗ.....	15
1.4 Класичні методи МН, РО і КЗ	16
1.4.1 Фільтрація.....	16
1.4.2 Аналіз	19
1.4.3 Structure from Motion	20
1.4.4 Оптичний потік	20
1.5 Використання ШНМ для КЗ та РО.....	20
1.5.1 КЗ та ШНМ.....	22
1.6 Сучасні види глибоких ШНМ.....	23
1.6.1 Сучасні гібридні архітектури глибокого навчання	23
1.6.2 Softmax-шари.....	24
1.6.3 Субдискретизація, Max-Pooling-шари	24
1.6.4 LeNet.....	24
1.6.5 AlexNet	25
1.6.6 GoogLeNet, Inception.....	25
1.6.7 ResNet (Residual Network), Inception-ResNet.....	26
1.7 Порівняння сучасних ШНМ для задач КЗ.....	26
1.8 Постановка завдання.....	27
1.9 Висновки за розділом 1	27
2 Матеріали та методи	28
2.1 Загальний опис паралельного методу	28

2.2	Опис методу взаємодії двох нейромереж, що працюють з даними різних розмірностей	30
2.3	Вибір архітектури нейромереж	30
2.4	Базова архітектура нейромережі	31
2.4.1	Архітектура нейромережі для двовірних даних	32
2.4.2	Архітектура нейромережі для тривимірних даних	32
2.5	Опис методу SFM для формування тривимірних даних.....	33
2.5.1	Bundler SFM	33
2.6	Висновки за розділом 2	34
3	Розробка архітектури програми.....	35
3.1	Порівняння бібліотек глибокого навчання	35
3.1.1	Torch	35
3.1.2	Theano	36
3.1.3	Caffe	36
3.1.4	Порівняння та вибір бібліотек глибокого навчання.....	37
3.2	Вибір мови програмування	38
3.3	Опис основних вимог до програми	41
3.4	Проектування розробки.....	41
3.5	Вимоги до технічних засобів	42
3.6	Висновки за розділом 3	43
4	Основні рішення щодо реалізації компонентів системи.....	44
4.1	Використовуване обладнання та ПЗ	44
4.2	Вибір моделей для проведення експерименту	44
4.3	Підготовка навчальної вибірки для ШНМ1	47
4.4	Підготовка навчальної вибірки для ШНМ2	48
4.5	Модель Caffe.....	48
4.6	Навчання класифікаторів	49
4.7	Висновки за розділом 4	50
5	Експлуатація, тестування та експериментальне дослідження програми	51

5.1 Результати випробування системи	51
5.2 Аналіз результатів	52
5.3 Висновки за результатами тестування	53
Висновки	54
Перелік джерел посилання	55
Додаток А Технічне завдання	59
Додаток Б Опис програми	64
Додаток В Текст програми	67
Додаток Д Слайди презентації	85

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

КЗ – комп'ютерний зір;

МН – машинне навчання;

РО – розпізнавання образів;

ШНМ – штучні нейронні мережі;

RGB – Red-Green-Blue (червоний-зелений-синій);

SFM – Structure from Motion (структура з руху).

ВСТУП

За останні два-три роки штучний інтелект змінив правила гри в індустрії безпілотних літальних апаратів [1]-[4]. Штучний інтелект може використовуватися для автономного виконання планів безпечних польотів, прогнозування потреб в обслуговуванні безпілотних літальних апаратів і захисту безпілотних літальних апаратів від атак кібербезпеки [1].

Під час польоту штучний інтелект також може використовуватися для виявлення і відстеження об'єктів, що цікавлять в режимі реального часу за допомогою комп'ютерного зору [4]. Ця потужна технологія відкриває двері для нових варіантів використання безпілотних літальних апаратів, які раніше були неймовірні [3]. Це може допомогти поліпшити реагування на надзвичайні ситуації, охорону тварин, охорону периметра, інспекції об'єктів і багато іншого [1].

У міру розвитку технологій безпілотних літальних апаратів якість відеозапису з камер і запису з безпілотних літальних апаратів також поліпшується [1]-[4]. Сьогодні ми можемо отримувати зображення з високою роздільною здатністю практично з будь-якого комерційного безпілотника [5].

Це підштовхнуло всіх власників вертикальних доменів, які вклали значні кошти в операції з безпілотниками, такі як управління об'єктами та безпека, до подальшої обробки та використання своїх аерофотознімків з безпілотників [6].

Зокрема, більшість користувачів хочуть мати можливість шукати свою голку в стозі сіна – автоматично знаходити порушників, виявляти незаконно припарковані автомобілі, виявляти тріщини на будівлях і багато іншого [5], [6].

Хоча важко перевершити природну здатність людини до виявлення та розпізнавання об'єктів, комп'ютерне бачення виграє у витривалості та послідовності. Комп'ютерний зір (КЗ) [5], [6], машинне навчання (МН) та розпізнавання образів (РО) [1]-[4] може допомогти автоматично швидко

аналізувати багатогодинні кадри з безпілотних літальних апаратів і генерувати бізнес-аналітику, яка дасть перевагу перед конкурентами [1]-[6].

Подібне забезпечення може використовуватися для різних галузей людської діяльності в тому числі для логістики, військової справи та охорони довкілля [1]-[6].

В процесі виконання поставленого завдання необхідно:

- розглянути основні принципи та існуючі методи вирішення завдань МН, РО і КЗ [6]-[12], провести аналіз доступних існуючих систем [13]-[19];
- розробити метод і детально описати запропонований підхід до вирішення поставленого завдання з використанням моделей [13]-[19];
- навчити класифікатор з використанням моделей [9]-[14];
- провести випробування запропонованої підсистеми розпізнавання з використанням моделей [13]-[19].

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

Перший розділ буде присвячено детальному аналізу проблемної області пов'язаної з МН та використанням МН для задач КЗ та РО. Особливу увагу буде приділено етапам РО в КЗ та аналізу класичних методів для кожного з етапів. Під час роботи буде проведено дослідження основних термінів таких, МН, КЗ, РО, оптичний потік, штучні нейронні мережі (ШНМ) та сучасні топології (архітектури) ШНМ.

1.1 Введення в МН

Світ стає розумнішим з кожним днем, і, щоб відповідати очікуванням споживачів, компанії все частіше використовують методи МН, щоб спростити завдання [1]-[4]. Можна побачити їх у використанні на пристроях кінцевих користувачів (за допомогою розпізнавання обличчя для розблокування смартфонів) або для виявлення шахрайства з кредитними картками (наприклад, для запуску попереджень про незвичайні покупки) [1]-[4].

У рамках штучного інтелекту та МН існує два основних підходи: навчання з учителем та навчання без учителя [1]-[4]. Основна відмінність полягає в тому, що один використовує позначені дані, щоб допомогти передбачити результати, а інший цього не робить. Однак між цими двома підходами є деякі нюанси і ключові області, в яких один перевершує інший. Цей пост прояснить відмінності, щоб ви могли вибрати найкращий підхід для вашої ситуації [1]-[4].

Що таке навчання з учителем. МН з учителем – це підхід, який визначається використанням позначених наборів даних [1]-[4]. Ці набори даних призначені для навчання або контролю методів класифікації даних або точного прогнозування результатів [1]-[4]. Використовуючи помічені входи і

виходи, модель може вимірювати свою точність і вчитися з плином часу [1]-[4]

Контрольоване навчання можна розділити на два типи проблем при інтелектуальному аналізі даних: класифікація та регресія [1]-[4]:

Завдання класифікації використовують методи для точного розподілу тестових даних по певних категоріях, таким як відділення яблук від апельсинів. Або, в реальному світі, методи контрольованого навчання можуть бути використані для класифікації спаму в окремій папці з вашої поштової скриньки [1]-[4]. Лінійні класифікатори, машини опорних векторів, дерева рішень і випадковий ліс - все це поширені типи методів класифікації [1]-[4].

Регресія – це ще один тип методу контрольованого навчання, який використовує метод для розуміння взаємозв'язку між залежними і незалежними змінними [1]-[4]. Регресійні моделі корисні для прогнозування числових значень на основі різних точок даних, таких як прогнози виручки від продажів для даного бізнесу. Деякі популярні методи регресії-лінійна регресія, логістична регресія та поліноміальна регресія [1]-[4].

Навчання без учителя. МН без учителя використовує методи МН для аналізу та кластеризації немаркованих наборів даних. Ці методи виявляють приховані закономірності в даних без необхідності втручання людини (отже, вони неконтрольовані) [1]-[4].

Моделі неконтрольованого навчання використовуються для трьох основних завдань: кластеризації, об'єднання та зменшення розмірності [1]-[4].

Кластеризація – це метод інтелектуального аналізу даних для групування немаркованих даних на основі їх подібності або відмінностей [1]-[4]. Наприклад, метод кластеризації k -середніх розподіляють аналогічні точки даних по групах, де значення K представляє розмір угруповання і ступінь деталізації. Цей метод корисний для сегментації ринку, стиснення зображень і т. д. [1]-[4].

Асоціація – це ще один тип неконтрольованого методу навчання, який використовує різні правила для пошуку взаємозв'язків між змінними в даному наборі даних [1]-[4]. Ці методи часто використовуються для аналізу ринкової корзини та механізмів рекомендацій, відповідно до рекомендацій: клієнти, які купили цей товар, також купили [1]-[4].

Зменшення розмірності – це метод навчання, який використовується, коли кількість об'єктів (або вимірювань) в даному наборі даних занадто велика [1]-[4]. Це зменшує кількість введених даних до керованого розміру, зберігаючи при цьому цілісність даних. Часто цей метод використовується на етапі попередньої обробки даних, наприклад, коли автокодери видаляють шум з візуальних даних для поліпшення якості зображення [1]-[4].

1.2 Введення в теорію РО

Теорія РО існує як розділ інформатики та суміжних дисциплін, що розвиває методи класифікації та ідентифікації об'єктів різної природи: сигналів, ситуацій, предметів, що характеризуються кінцевою множиною деяких ознак [1]-[4].

При постановці класичної задачі розпізнавання образів прийнято використовувати сувору математичну мову, ґрунтуючись на логічних міркуваннях і математичних доказах [1]-[4]. На противагу цьому підходу, існують методи розпізнавання образів з використанням МН і штучних нейронних мереж (ШНМ), сформовані не настільки строго формалізованими підходами до розпізнавання, але, як буде показано далі, демонструють не гірший, а в деяких випадках значно перевершує класичні методи результат [1]-[4].

Характер проблематики РО також обмежує область застосування різних традиційних методів вузькими спеціалізованими напрямками, в кожному з яких найбільш ефективними виявляються одні методи і

неефективними інші. Цей фактор обумовлює складність загального вирішення проблеми індукції і в цій дисципліні [1]-[4].

1.3 Введення в КЗ

Область КЗ, в свою чергу, розвивається як теорія і технологія створення машин, що виробляють виявлення, відстеження і класифікацію об'єктів [5], [6]. Як наукова дисципліна, КЗ тісно пов'язаний з машинним навчанням і теорією розпізнавання образів, але відноситься до більш спеціалізованої сфери теорії і технології створення штучних систем, які отримують інформацію і оперують інформацією із зображень [5], [6]. Автоматичне планування або прийняття рішень на основі підсистем КЗ так само займає важливу частину в області штучного інтелекту, оскільки автономні системи досить складного рівня організації, що виконують деякі механічні дії (наприклад, переміщення робота через деяке середовище), потребують високорівневих даних, що представляють інформацію про середовище, в якому вони функціонують [5], [6]. Тому КЗ, як і розпізнавання образів, тісно пов'язане з обробкою сигналів, адже багато методів обробки одновимірних сигналів можуть бути природним шляхом розширені для обробки двовимірних або багатовимірних сигналів в рамках теорії комп'ютерного зору, що використовує статистику, методи оптимізації та геометрії [5], [6].

КЗ також тісно пов'язаний з областями обробки зображень і машинного зору [5], [6]. Область обробки зображень зосереджена на аналізі і перетвореннях одних зображень в інші з використанням математичних методів [5], [6]. Область машинного зору, часто розглядається як розділ більш загальної теорії КЗ, зосереджена на технологіях промислового застосування [5], [6]. Існують також і інші більш вузькі області, пов'язані, залежні або виникли на основі теорії КЗ, такі як, наприклад, область візуалізації, зосереджена на процесі створення зображень, їх обробкою і аналізом [5], [6].

Теорія КЗ досліджує методи обробки не тільки двовірних зображень, але і працює з тривимірними сценами, що буде продемонстровано в рамках даної роботи [5], [6]. В даний момент не існує ні стандартного формулювання проблеми комп'ютерного зору, ні формулювання того як повинна вирішуватися проблема КЗ, замість чого розроблена маса методів для вирішення різних строго визначених завдань, при цьому використовувані методи рідко узагальнюються для широкого кола застосування [5], [6]. Як і в розглянутих вище двох великих теоріях, досліджуваних і застосовуваних в даній роботі, цей фактор породжує складності при вирішенні загальної проблеми індукції в теорії КЗ [5], [6].

1.4 Класичні методи МН, РО і КЗ

Тут і далі теорія КЗ, теорія РО, теорія МН і обробка зображень будуть розглядатися в сенсі РО в КЗ. В іншому випадку буде вказано інший сенс. З метою стислості може згадуватися тільки одна з цих дисциплін в залежності від контексту [7], [8].

Відомі методи КЗ, РО і МН можна розділити на три групи за умовами виконуваних дій [7], [8]:

- методи фільтрації;
- методи аналізу;
- методи навчання.

Основні відомі методи та методи реалізовані в бібліотеці КЗ OpenCV, написаної на C/C++, і займає лідуючу позицію в рішеннях завдань КЗ з використанням класичних підходів [7], [8].

1.4.1 Фільтрація

У задачах КЗ фільтрація найчастіше використовується для попередньої обробки зображення перед аналізом його внутрішніх

морфологічних ознак, але зустрічаються і завдання, в яких достатнім і бажаним буде використання тільки фільтрації як такої (наприклад, в задачах машинного зору) [1]-[8].

Бінаризація за порогом та вибір області гистограми. Для Red-Green-Blue (RGB) зображення та чорно-білих зображення порогом – є значення кольору. Вибір порога, за яким відбувається бінаризація, є визначальним для загального процесу бінаризації [1-8]. Адаптивний метод для вибору порога зазвичай обирається для виконання бінаризації. Такий метод може на вибір математичного сподівання, моди або піків гистограми. При роботі з гистограмою бінаризація ефективна для сегментації кольорів [1]-[8].

Перетворення Фур'є рідко використовується при обробці зображень в класичному вигляді через те що для аналізу зображень одновимірного перетворення не вистачає і просто виникає необхідність використання значно більш ресурсоємного двовимірного перетворення [1]-[8]. Більш доречним є використання цього методу для виконання аналізу спектра за необхідністю. Оскільки використання згортки цікавить області з уже готовим фільтром виявляється швидше і простіше в реалізації. Проте, одновимірне перетворення Фур'є застосовується при компресії зображень [1-8].

Фільтри частот. Найпростіший приклад фільтра низьких частот – фільтр Гауса, фільтра високих частот – фільтр Габора [1]-[8]. Для кожної точки зображення вибирається вікно, в рамках якого виконується добуток вихідних даних з фільтром того ж розміру (згортка) [1]-[8]. Такий підхід отримав досить широке поширення в практиці, дозволяючи виділяти на зображенні цікаву інформацію, редукуючи зайву інформацію [1]-[8]. Зокрема, підхід поширений як одна з реалізацій швидкого шумозаглушення в багатьох галузях науки і техніки.

Вейвлет-перетворення. Для згортки з сигналом (областю зображення) використовується якась характеристична функція, звана вейвлетом, що визначається як вейвлет-перетворення [1]-[8]. Вейвлети – це сімейства функцій, локальних за часом і за частотою, в яких всі функції виходять з

однієї за допомогою її зрушень і розтягувань по осі часу. Існує набір класичних функцій, що використовуються в вейвлет-аналізі. До них відносяться вейвлет Хаара, вейвлет Морле, вейвлет МНат, вейвлет Добеши [1]-[8]. На практиці вейвлет-аналізом називається пошук довільного патерну на зображенні за допомогою згортки зображення з моделлю цього патерну [1-8]. Класичні вейвлети використовуються для стиснення або класифікації зображень [1]-[8].

Фільтри функцій. У цьому підході використовуються математичні фільтри, що дозволяють виявляти прості математичні функції на зображенні, для чого формується акумулюючий зображення (накопичувальний простір), в якому для кожної точки вихідного зображення будується безліч породжують її функцій [1]-[8]. Класичним прикладом є узагальнене перетворення Хафа, що застосовується до бінаризованих зображень, що дозволяє знаходити на зображенні будь-які параметризовані функції [1]-[8]. Модифіковане перетворення Хафа дозволяє шукати будь-які фігури, але в обробці зображень його використання пов'язане з недостатньою стабільністю: висока чутливість до якості бінаризації і низька швидкість роботи змушують шукати цьому методу більш ефективні альтернативи [1]-[8]. Аналогом перетворення Хафа для прямих є перетворення радону, що застосовується до небінаризованих зображень і обчислюється через Фільтр Фур'є, що дає значний виграш у продуктивності на великих масивах даних [1]-[8].

Інші фільтри. Модифікації існуючих класичних фільтрів надають змогу вирішити переважну кількість завдань фільтрації, але крім них існують десятки більш спеціалізованих методів фільтрації, проте на практиці такі застосовуються значно рідше [1]-[8]. Найбільш перспективними є ітераційні фільтри, такі як [1]-[8]:

- активна модель зовнішнього вигляду;
- ріджлет-перетворення та курвлет-перетворення, що розроблені на основі вже відомих вейвлет-функцій і аналізу в полі перетворення радону;

– бімлет-перетворення, що реалізують від частин завдання фільтрації та частково – завдання аналізу, що дає змогу відшукувати контури об'єктів [1]-[8].

Такі фільтри відрізняються рідкісною специфікою, тому обмежені за оглядом в рамках даної роботи [1]-[8].

1.4.2 Аналіз

Використовуючи фільтрацію користувачі зазвичай отримують набір даних для наступної обробки, проте більшість задач області КЗ потребують аналізу внутрішньої структури та морфологічних ознак власне зображень та об'єктів на них, для чого використовується в подальшому методи логічного аналізу зображень [1]-[8].

Методи математичної морфології є результатом теорії і техніки аналізу і обробки геометричних структур, заснованих на теорії множин, топології і випадкових функціях [1]-[8].

Ці методи реалізуються базовими операціями перенесення, нарізання та ерозії двійкових зображень, а також похідними операціями [1]-[8]. Операції двійкової морфології є деяким перетворенням упорядкованої множини або підмножини (області зображення) за допомогою структурного елемента [1]-[8].

Структурним елементом є двійкове зображення довільного розміру і довільної структури, але найчастіше використовуються симетричні елементи, як прямокутник фіксованого розміру або коло фіксованого діаметра [1]-[8]. Результатом перетворення також є двійкове зображення [1]-[8].

Методи математичної морфології дозволяють видаляти шуми з двійкових зображень, а також реалізують методи пошуку контурів, але на практиці ці методи використовуються в зв'язці з іншими методами.

1.4.3 Structure from Motion

Сьогодні в реалізації Structure from Motion (SFM) - методів застосовується кілька класичних методів, заснованих на різних методах супроводу особливих точок [1]-[8]. Методи супроводу особливих точок можна розділити на детерміністські та імовірнісні методи [1]-[8].

Детерміністські методи використовують якісні евристики руху, і завдання супроводу зводиться до мінімізації функції відповідності наборів точок [1]-[8]. Типовими представниками детерміністських методів є методи, засновані на обчисленні щільного і розрідженого оптичного потоку, і методи зіставлення дескрипторів ключових точок, найбільшого поширення з яких отримали сьогодні методи bundle adjustment [1]-[8].

1.4.4 Оптичний потік

Існує безліч різновидів методів обчислення оптичного потоку, таких як фазова кореляція, блокові методи, дискретні методи оптимізації, але в рамках даної роботи варто згадати диференціальні методи, що обчислюють рух між кадрами в кожному пікселі на основі наближення сигналу відрізком ряду Тейлора, використовуючи приватні похідні за часом і просторовим координатам, в результаті чого формують поле швидкостей [1]-[8]. Методи розрізняються з точки зору універсальності, точності і щільності одержуваного векторного поля. В основі більшості сучасних диференціальних методів обчислення оптичного потоку лежать ідеї методу Лукаса-Канаді [1]-[8].

1.5 Використання ШНМ для КЗ та РО

ШНМ – це комп'ютерні системи, призначені для машинного навчання, які імітують роботу людського мозку-природної нейронної мережі [1-8].

Іноді називають комп'ютерні нейронні мережі, ці системи навчаються шляхом моделювання процесу навчання людини: вони приймають дані, шукають закономірності та засвоюють ці закономірності для створення логічних правил обробки даних або ідентифікації речей [9-14].

Наприклад, можна надати нейронної мережі два набори даних [9]-[14]:

- набір даних один називається кішка, який містить зображення кішок;
- набір даних номер два називається без кішки і складається з зображень без кішок.

Виходячи з цих наборів даних, ШНМ може навчитися розпізнавати зображення котів, навіть не знаючи, що таке кішка. Вона створює правила, що дозволяють зрозуміти, що спільного у кожного набору зображень серед усіх зображень і чим набори зображень відрізняються один від одного [9]-[14].

Звичайно, найпростіші терміни не відповідають такій складній системі і процесу. Насправді ШНМ – це надзвичайно складні системи, і вчені мають багато роботи, коли мова йде про розуміння та використання машинного навчання. Точно так само, як людське навчання та пам'ять, нейронні мережі виконують завдання, для виконання яких ми не можемо безпосередньо запрограмувати їх, або для яких пряме програмування було б нереалістичним, наприклад [9]-[14]:

- прогнозування розвитку фондового ринку;
- визначення авторства рукописів на основі вибору слів і стилю.

Незважаючи на їх складність, специфіку і те, як багато ще належить дізнатися про ШНМ і супроводжує їх машинному навчанні, ви вже можете знайти цю технологію в багатьох практичних додатках. Однією з найбільш поширених налаштувань для ШНМ є область КЗ [9]-[14].

1.5.1 КЗ та ШНМ

КЗ – неймовірно складна область з оманливим простим визначенням: Автоматизація видів завдань зі збору інформації, які зорова система людини виконує автоматично. На практиці це означає, що комп'ютери навчаються витягувати інформацію з зображень, незалежно від того, чи є вони [9]-[14]:

- цифрові зображення;
- потокове або збережене відео;
- відеозапис в реальному часі з відеокамери.

Області застосування такого роду технологій включають [9]-[14]:

- розпізнавання осіб;
- розпізнавання об'єктів;
- виявлення та аналіз руху;
- відновлення зображення.

Деякі з найпростіших завдань вже повинні звучати знайомо – комп'ютерне бачення на роботі, коли Facebook пропонує вам відзначити себе або друга на фотографії [9]-[14].

Приклад із зображеннями котів може здатися дурним, але це ілюструє основний процес, за допомогою якого комп'ютери використовують КЗ для виконання завдань, які колись виконували люди [9]-[14]. У міру зростання набору даних нейронна мережа може вдосконалювати свій метод, підвищуючи його точність і правильно виконуючи все більш складні тести [9]-[14].

Більш практичним прикладом, можливо, є розпізнавання рукописного тексту. Хоча ідентифікація літери «а» може бути легким завданням для спостерігача-людини, програмування, необхідне для передачі геометрії літери «а», значно складніше, особливо якщо врахувати всі можливі змінні в реальному зразку почерку [9]-[14]:

- висота;
- ширина;

- пропорція;
- відстань;
- акуратність.

Тим не менш, нейронна мережа, навчена на найрізноманітніших зразках почерку, може навчитися розпізнавати кожну букву У всіх її нескінченних варіаціях, дозволяючи нам робити такі речі, як оцифровка рукописних заміток або розшифровка погано написаного або частково знищеного почерку [9]-[14].

1.6 Сучасні види глибоких ШНМ

Протягом останніх десятиліть МН мало величезний вплив на наше повсякденне життя з прикладами, включаючи ефективний пошук в Інтернеті, системи самостійного водіння, КЗ і оптичне РО [15]-[19].

Зокрема, моделі глибоких ШНМ стали потужним інструментом машинного навчання та штучного інтелекту. Глибока ШНМ – це така ШНМ з декількома шарами між вхідним і вихідним шарами [15]-[19].

Успіх глибоких нейронних мереж призвів до таких проривів, як зниження частоти помилок в словах при розпізнаванні мови на 30% в порівнянні з традиційними підходами (найбільший приріст за 20 років) або різке зниження частоти помилок в конкурсі з розпізнавання зображень з 2011 року (з 26% до 3,5%, в той час як люди досягають 5%) [15]-[19].

1.6.1 Сучасні гібридні архітектури глибокого навчання

З кінця 90-х років великого поширення набули гібридні архітектури глибокого навчання на основі неокогнітрона, що домагаються вражаючих практичних результатів. Ці архітектури називаються згортковими нейронними мережами [15]-[19].

1.6.2 Softmax-шари

У сучасних ШНМ для моделювання ймовірного розподілу використовуються softmax-шари, що включають N нейронів за цільовою кількістю класів. Вихід кожного нейрона залежить від суматорів всіх інших нейронів шару [15]-[19].

1.6.3 Субдискретизація, Max-Pooling-шари

У сучасних ШНМ використовуються субдискретні шари, що виконують зменшення вхідної карти ознак. Найчастіше для цього використовується метод вибору максимального елемента (maxpooling). Використання max-pooling дозволяє зробити мережу інваріантною до масштабних перетворень [15]-[19].

1.6.4 LeNet

LeNet – перша сучасна згорткова нейромережа, розроблена в 90-х роках Яном ЛеКуном, навчалася для розпізнавання рукописних символів. Остання версія мережі LeNet – 5 демонструвала на той момент вражаючі результати за рахунок реалізації декількох новаторських ідей. Основною відмінністю від неокогні – трону було включення субдискретизуючого max-pooling шару після кожного згорткового шару і включення повнозв'язних шарів на виході мережі (рис. 1.1). З цього моменту згорткові і max-pooling-шари стають серцем сучасних глибоких мереж [15]-[19].

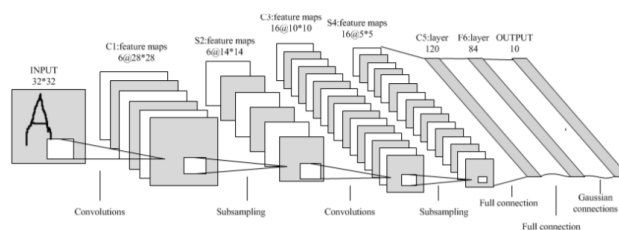


Рисунок 1.1 – Архітектура навчалася

1.6.5 AlexNet

У 2012 році відбулася ще одна поворотна подія: вперше в історії конкурс з класифікації зображень бази ImageNet на 1000 класів з перевагою по точності розпізнавання майже в два рази виграла згорткова нейромережа AlexNet, навчена Алексом Крижевським [15]-[19]. Архітектура включала згорткові шари, max-pooling-шари, повнозв'язні шари на виході мережі. Використовувалися методи dropout і локальної нормалізації. В якості функції активації використовувалася функція ReLU [15]-[19]. Навчання мережі проходило на двох потужних графічних прискорювачах протягом тижня [15]-[19].

1.6.6 GoogLeNet, Inception

У 2014 році ImageNet Recognition Challenge виграла нейромережу Google GoogLeNet, на основі першої In-ceptionархітектури [15]-[19]. В архітектурі був зменшений розмір згортки, були паралельно включені невеликі згортки різного масштабу, видалені повнозв'язні шари на виході мережі і замість них включені шари, названі global average pooling. Нова архітектура отримала активний розвиток завдяки своїй ефективності [15]-[19]. Inception-v3 – остання реалізація, що включає кілька десятків шарів (рис. 1.2) [15]-[19].

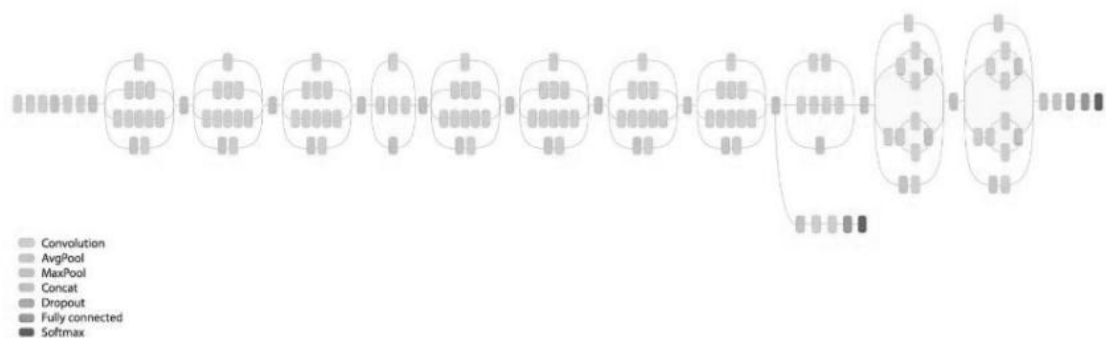


Рисунок 1.2 – Архітектура Inception-v3 [17]

1.6.7 ResNet (Residual Network), Inception-ResNet

Inception-v3 повинен був виграти ImageNet 2015, чого не сталося, оскільки з'явилася архітектура ResNet від Microsoft [15]-[19], в основі якої лежить метод передбачення різниці між тим, що видають попередні шари і метою розпізнавання. Таким чином, глибокі шари отримують можливість обнулити ваги або просто пропустити сигнал. Архітектура отримала подальший розвиток, наприклад, у вигляді високоефективної гібридної мережі Inception-ResNet [15]-[19].

1.7 Порівняння сучасних ШНМ для задач КЗ

У таблиці 1.1 наведено порівняння сучасних архітектур ШНМ, що зазвичай використовуються при вирішенні задач КЗ та РО. Варто зазначити, що чисельні роботи демонструють використання усіх представлених топологій.

Таблиця 1.1 – Порівняння топологій ШНМ

	Багатошарові перцептрони	Рекурентні ШНМ	Глибокі ШНМ	Гібридні види глибоких ШНМ
Швидкість навчання	Висока	Середня	Низька	Середня
Якість розпізнавання	Середня	Висока	Висока	Висока
Методи навчання	Класичні	Класичні	Класичні	Класичні
Підтримка у фреймворках МН	Підтримується	Підтримується	Підтримується	Підтримується (за умови)

За результатами порівняння можна зробити висновок, що саме гібридні види глибоких ШНМ є найбільш перспективним напрямком на сьогоднішній день, адже відрізняються помірно високою швидкістю

навчання та високим рівнем точності. Для її навчання застосовуються усі відомі класичні методи, що не накладає особливих вимог та потреб у зв'язку з цим. Більш того, більшість бібліотек МН вже мають попередньо реалізовані архітектури таких ШНМ та основною задачею стає виключно їх навчання.

1.8 Постановка завдання

Дослідження та програмна реалізація методів розпізнавання тривимірних об'єктів на основі поля оптичного потоку для інтегрованої автоматизованої гібридної системи виявлення об'єктів при аналізі зображень з безпілотних літаючих пристроїв, наприклад, квадрокоптерів.

Подібне забезпечення може використовуватися для різних галузей людської діяльності в тому числі для логістики, військової справи та охорони довкілля.

1.9 Висновки за розділом 1

Під час роботи над першим розділом велику увагу було присвячено детальному аналізу проблемної області пов'язаної з МН та використанням МН для задач КЗ та РО. Було детально розглянуто РО в КЗ та проведено аналіз класичних методів для кожного з етапів. Під час роботи були розглянуті основні терміни: МН, КЗ, РО, оптичний потік, ШНМ та сучасні топології (архітектури) ШНМ.

2 МАТЕРІАЛИ ТА МЕТОДИ

Безпілотні літальні апарати (наприклад, квадрокоптери) на сьогоднішній день оснащуються високоточним обладнанням, що забезпечує відео- і фото-зйомку в поєднанні з реєстрацією поточних параметрів телеметрії. Вузькоспеціалізоване ж обладнання взагалі часто оснащено крипто захищеним командно-телеметричним каналом і крипто захищеним каналом передачі фото- та відео-інформації, включаючи координати, висоту польоту і номер кадру.

В першому розділі було розглянуто, як МН може допомогти у вирішенні задач пов'язаних із РО. Було наведено обґрунтування використання сучасних гібридних топологій ШНМ для побудови нейромоделей.

В другому розділі буду більш детально розглянуто запропонований метод використання ШНМ та паралельних обчислень для задач РО.

2.1 Загальний опис паралельного методу

Для реалізації системи розпізнавання були обрані сучасні ШНМ, оскільки ефективність цих систем сьогодні продемонстрована революційними практичними результатами. В ході роботи над системою було встановлено, що, незважаючи на точність сучасних ШНМ, через особливості реєстрації навколишнього оточення з борту квадрокоптерів за допомогою гіростабілізованих камер середньої роздільної здатності або дзеркальних камер високої роздільної здатності зі світлосильними об'єктивами і повнокадровими матрицями, для системи залишаються вельми характерними помилки першого і другого роду, що не допустимо через специфіку призначення комплексу. Внаслідок чого було прийнято рішення випробувати сміливі експериментальні ідеї для досягнення необхідного результату, а також не зупинятися в розвитку архітектури мережі [20]-[22].

Одним з випробуваних рішень, що збільшують точність розпізнавання, стала розробка паралельного методу з двох ШНМ, що працюють з даними різної розмірності (рис.2.1).

Нейронна мережа ШНМ1 для двовірних даних організована звичним сьогодні чином і призначена для розпізнавання об'єктів на двовимірних зображеннях або кадрах відео потоку [20]-[22]. ШНМ2 для тривимірних даних організована для розпізнавання об'єктів в тривимірному нормалізованому хмарі точок, відновленому за допомогою SFM-методу на базі детерміністського методу супроводу особливих точок. Фактично, ШНМ2 необхідна для уточнення і валідації результатів роботи ШНМ1, що вирішує ряд виникаючих проблем, пов'язаних з помилками першого і другого роду [20]-[22].

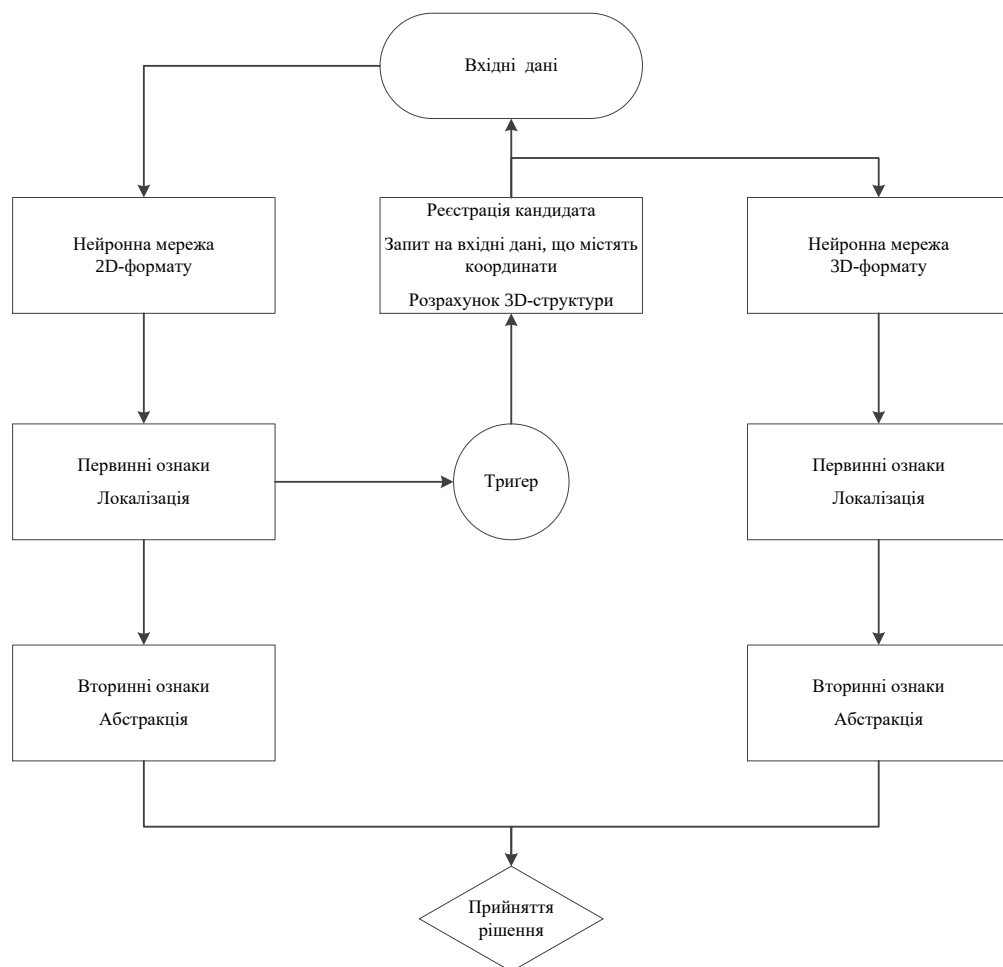


Рисунок 2.1 – Блок-схема паралельного методу

2.2 Опис методу взаємодії двох нейромереж, що працюють з даними різних розмірностей

Метод системи використовує властивість локалізації первинних ознак поверхневих шарів глибоких мереж. Локалізуючи первинні ознаки, ШНМ допомагає визначити кандидата на ціль і зіставити його координати з безліччю координат кожного зображення у вхідних даних. Кандидату присвоюється унікальний дескриптор, що містить його передбачуваний клас і координати. На основі цього дескриптора, з урахуванням деякої області навколо даної точки, проводиться пошук кандидата на інших зображеннях або кадрах у вхідних даних. Знайдена група зображень або кадрів містить виявлений об'єкт або його частину [20]-[22].

2.3 Вибір архітектури нейромереж

Для реалізації ШНМ була обрана бібліотека Caffe [23], [24] зважаючи на її зручною і прозорою структури, високу швидкість розробки при її використанні, досить повної документації і можливості звернення до широкої спільноти розробників, знайомих з нею. Крім того, Caffe демонструє відмінні результати тестування продуктивності. Caffe підтримує CUDA [23], [24], і збірка бібліотеки з імплементацією коду для Nvidia GPU дає колосальний приріст в швидкості роботи нейромережі, оскільки багато обчислювальні операції в процесі функціонування нейромережових класифікаторів реалізуються в кілька разів або на порядок швидше на графічних чіпах.

Більш детально обґрунтування фреймворку буде наведено в третьому розділі.

2.4 Базова архітектура нейромережі

Для реалізації базової архітектури була обрана і модифікована архітектура нейронної мережі VGG-16 (рис. 2.2) [17]-[19]. Дана надглибока архітектура, як і всі ШНМ останнього покоління, використовує надмалі розміри згорток, що дозволяє мережі з ростом кількості шарів до певного рівня, збільшуючи репрезентативну потужність, не втрачати колосальні ресурси на навчання.

Важливим критерієм вибору була зручність і швидкість розробки. Хоча сьогодні існують більш ефективні архітектури мереж, VGG-16 досить легко реалізується за допомогою Caffe, що повністю підходить для тестування експериментальної системи з перспективою подальшого удосконалення [17]-[19].

Існує також більш глибока версія архітектури-VGG-19, але, оскільки дані архітектури розроблялися для класифікації по 1000 класам і володіють значним числом параметрів при даній глибині, з метою адекватності обсягу мережі [17]-[19] наявної навчальної вибірки і адекватності репрезентативної потужності мережі цільовій кількості класів, було вирішено реалізувати менш об'ємний варіант для запобігання запам'ятовування мережею вхідної вибірки і втрати здатності до класифікації. VGG-16, як і VGG-19, демонструє хороші результати на відомих дата-сетах (таблиця 2.1, таблиця 2.2) [17]-[19].

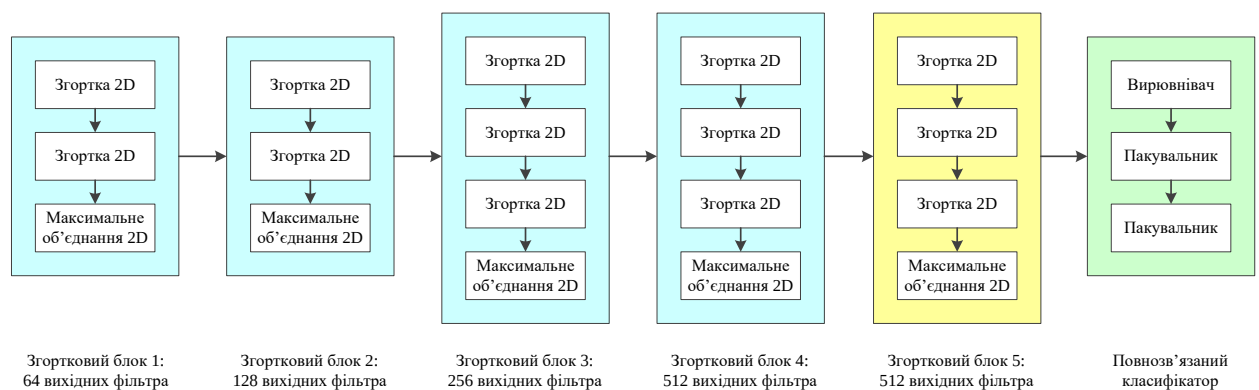


Рисунок 2.2 – Архітектура VGG-16

Таблиця 2.1 – Результати VGG на ILSVR-2012

Модель	Помилка класифікації на ILSVR-2012	
	Набір даних для валідації	Набір даних для тестування
16 шарів	7,5%	7,4%
19 шарів	7,5%	7,3%
Злиття моделей	7,1%	7%

Таблиця 2.2 – Результати VGG на VOC та Caltech

Модель	VOC-2007 середня точність	VOC-2012 середня точність	Caltech-101 середня точність	Caltech-256 середня точність
16 шарів	89,3%	89%	91,8%	85%
19 шарів	89,3%	89%	92,3%	85,1%
Злиття моделей	89,7%	89,3%	92,7%	85,3%

2.4.1 Архітектура нейромережі для двовірних даних

Архітектура нейромережі для двовірних даних повністю повторює архітектуру VGG-16, за винятком повнозв'язних шарів на виході моделі, число нейронів в яких було зменшено з 4096 до 96 і останнього повнозв'язного softmax-шару, число нейронів в якому було зменшено для випадку п'яти класів об'єктів. VGG реалізується з функцією активації ReLU [17]-[19].

2.4.2 Архітектура нейромережі для тривимірних даних

Архітектура ШНМ для тривимірних даних аналогічна розглянутій архітектурі для двовірних даних у всьому, крім внутрішнього устрою ознак в каналі blob і формату оброблюваних даних [17]-[19]. Крім розглянутих RGB-ознак, для представлення тривимірних даних необхідно визначити ознаки трьох просторових вимірювань в полі `input_shape` при налаштуванні моделі мережі в Caffe. Дані, що не відносяться до зображень в звичному сенсі, в

нейромережах Caffe відображаються через вектори, що зберігаються в бінарному форматі HDF5 [17]-[19].

2.5 Опис методу SFM для формування тривимірних даних

Для вирішення задачі побудови тривимірних моделей об'єктів був обраний детерміністський метод формування тривимірної хмари точок на основі дескрипторів, зважаючи на достатню ефективність подібних методів, їх високу продуктивність і простоту імплементації [11]-[14].

2.5.1 Bundler SFM

Bundler – це поширена бібліотека, що реалізує досить надійний SFM-метод, в основі якого лежить дескриптор особливих точок SIFT і метод bundle adjustment [20]-[22]. Маючи групу зображень і будь-яку відому пару параметрів-кут огляду об'єктива, фокусна відстань об'єктива і розмір матриці реєструючого обладнання, за допомогою bundler можна отримати детально параметризовані набори точок об'єкта в тривимірному просторі. Особливо добре метод працює з зображеннями, знятими в просторі з вираженою топологією (рисунок 2.3) [20]-[22].



Рисунок 2.3 – Використання Bundle із зображеннями, отриманими при зйомці з борту квадрокоптера

2.6 Висновки за розділом 2

В другому розділі буду більш детально розглянуто запропонований метод використання ШНМ та паралельних обчислень для задач РО, для цього було розглянуто та обґрунтовано:

- паралельний метод використання двох ШНМ, що працюють з вхідними даними різної розмірності;
- метод взаємодії двох ШНМ;
- вибір гібридної архітектури ШНМ;
- використання SFM для формування тривимірних даних.

Безпілотні літальні апарати (наприклад, квадрокоптери) на сьогоднішній день оснащуються високоточним обладнанням, що забезпечує відео- і фото-зйомку в поєднанні з реєстрацією поточних параметрів телеметрії. Вузькоспеціалізоване ж обладнання взагалі часто оснащено крипто захищеним командно-телеметричним каналом і крипто захищеним каналом передачі фото- та відео-інформації, включаючи координати, висоту польоту і номер кадру.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

Другий розділ було присвячено дослідженню запропонованих методів для РО за тривимірними даними.

Третій розділ буде присвячено висвітленню питань з програмної реалізації запропонованих методів. В ході роботи буде досліджено та обґрунтовано технічні рішення з вибору мови програмування, бібліотеки МН, опису вимог до ПЗ та технічних засобів та платформ для роботи та тестування.

3.1 Порівняння бібліотек глибокого навчання

Сьогодні існує більше десятка бібліотек глибокого навчання, деякі з яких мають вузьку специфікою. Грунтуючись на рекомендаційній інформації з широкого кола джерел, для подальшого розгляду було обрано три бібліотеки: Torch, Theano і Caffe [23], [24].

3.1.1 Torch

Torch – це бібліотека МН з відкритим вихідним кодом, наукова обчислювальна платформа та мова сценаріїв, заснована на мові програмування Lua [23], [24]. Вона надає широкий спектр алгоритмів для глибокого навчання і використовує мову сценаріїв LuaJIT і базову реалізацію мови С. Вона була створений в IDIAP в EPFL. З 2018 року Torch більше не знаходиться в активній розробці [23], [24]. Однак PyTorch, заснований на бібліотеці Torch, активно розвивається з червня 2021 року [23], [24]. Основним пакетом Torch є torch. Він надає гнучкий n -мірний масив або тензор, який підтримує базові процедури для індексування, нарізки, транспонування, приведення типів, зміни розміру, спільного використання

сховища і клонування [23], [24]. Цей об'єкт використовується більшістю інших пакетів і, таким чином, утворює основний об'єкт бібліотеки [23], [24]. Тензор також підтримує математичні операції, такі як `max`, `min`, `sum`, статистичні розподіли, такі як рівномірний, нормальний і поліноміальний, і операції BLAS, такі як точковий добуток, множення матриці на вектор, множення матриці на матрицю, матрично-векторний добуток і матричний добуток [23], [24].

3.1.2 Theano

Theano – це бібліотека Python та оптимізуючий компілятор для обробки та оцінки математичних виразів, особливо матричних [23], [24]. У Theano обчислення виражаються з використанням синтаксису NUMPY-esque і компілюються для ефективної роботи на архітектурах CPU або GPU [23], [24]. Theano-це проект з відкритим вихідним кодом, розроблений в основному Монреальським інститутом алгоритмів навчання при Монреальському університеті [23], [24]. Назва програмного забезпечення відсилає до стародавнього філософа Теано, довгий час пов'язаному з розвитком золотої середини. 17 Травня 2018 року Кріс Фоннесбек написав від імені команди розробників PyMC, що розробники PyMC офіційно візьмуть на себе контроль над обслуговуванням системи, як тільки вони підуть у відставку. 29 січня 2021 року вони почали використовувати ім'я Aesara для своєї вилки Теано [23], [24].

3.1.3 Caffe

Caffe реалізована на C++ і має обгортки для Python і Matlab. Топологія ШНМ, вихідні дані і спосіб навчання задаються за допомогою конфігураційних protobuf-файлів (технологія і протокол серіалізації даних, що перевершує за ефективністю xml і json) у форматі prototxt [23], [24].

Побудова структури мережі виконується з простотою, зручністю і наочністю [23], [24]. З розглянутих бібліотек є найзручнішою у використанні, крім того, перевершує інші розглянуті фреймворки в швидкості навчання [23], [24]. Бібліотека Caffe підтримується досить великою спільнотою розробників і користувачів і на сьогоднішній день є найпоширенішою бібліотекою глибокого навчання широкого кола застосування [23], [24].

3.1.4 Порівняння та вибір бібліотек глибокого навчання

У таблиці 3.1 наведено порівняння розглянутих бібліотек для глибокого МН.

З результатів порівняння видно, що бібліотека Caffe найбільш перспективний вибір для завдань глибокого МН, адже на сьогоднішній день налічую найбільшу кількість топологій та моделей нейромереж, що активно підтримуються та розширюються. Також офіційно підтверджено, що швидкість навчання методів бібліотеки Caffe найбільша. Більш того, бібліотека була написана на мові програмування C++, що дало змогу в подальшому активно підтримувати та реалізовувати технологію CUDA для виконання паралельних обчислень на ядрах GPU.

Таблиця 3.1 – Результати порівняння бібліотек глибокого МН

	Torch	Theano	Caffe
Підтримка мов програмування	C++, Lua	Python	C/C++, Python, Matlab
Підтримка моделей нейромереж	Середня	Висока	Найвища
Швидкість навчання моделей	Середня	Середня (для глибоких ШНМ збільшується)	Найвища
Паралелізація обчислень	Можлива	Можлива	Можлива
Підтримка паралельних систем на базі GPU	CUDA	CUDA (наявні обмеження)	CUDA та технологія multi-GPU
Спільнота	Менш активна	Активна	Найбільша

3.2 Вибір мови програмування

При роботі із бібліотекою Caffe найбільш доречною є вибір мови програмування C++, проте для детальності проведемо огляд найбільш привабливих характеристик цієї мови програмування [25], [26].

C++ – це кроссплатформна мова, яку можна використовувати для створення високопродуктивних застосунків [25], [26].

C++ була розроблена Б'ярне Страуструпом як розширення мови C [25], [26].

C++ надає програмістам високий рівень контролю над системними ресурсами і пам'яттю [25], [26].

Мова була оновлена 3 рази в 2011, 2014 і 2017 роках до C++11, C++14 і C++17.

Навіщо Використовувати C++. C++ – одна з найпопулярніших мов програмування в світі. C++ можна знайти в сучасних операційних системах, графічних інтерфейсах користувача та вбудованих системах [25], [26].

C++ – це об'єктно-орієнтована мова програмування, яка дає чітку структуру програмам і дозволяє повторно використовувати код, знижуючи витрати на розробку [25], [26].

C++ є портативною і може використовуватися для розробки застосунків, які можуть бути адаптовані до декількох платформ [25], [26].

Оскільки C++ близька до C# і Java, програмістам легко перемикається на C++ або навпаки [25], [26].

Серед переваги C++ виділяють цілу низку таких [25], [26]:

– мобільність. C++ надає цю функцію переносимості, що дозволяє розробляти коди, не піклуючись про обладнання. Це дозволяє переносити розробку програми з однієї платформи на іншу. Наприклад, програміст працює в операційній системі (ОС) Windows і з якоїсь причини потрібно переключитися на Linux, коди з ОС Windows також будуть виконуватися в ОС Linux без будь-яких помилок [25], [26];

– мова програмування середнього рівня. Будучи мовою програмування середнього рівня, її можна розглядати як мову як низького, так і високого рівня. Функції мови високого рівня допомагають розробляти ігри та настільні програми, тоді як функції мови низького рівня допомагають створювати ядра та драйвери [25], [26];

– об'єктно-орієнтованість. Концепції об'єктно-орієнтованості, такі як поліморфізм, інкапсуляція, спадкування та абстракція, дають C++ найбільшу перевагу перед іншими мовами програмування. Це виявилось дуже важливим, оскільки цієї функції не було в C, це допомогло користувачам обробляти дані як об'єкти та класи [25], [26];

– багатопарадигменість мови програмування. Парадигма відноситься до планування, пов'язаного з програмуванням. Це стосується логіки, стилю і того, як ми працюємо з програмою. C++ – це багатопарадигмальна мова програмування, оскільки вона слідує трьом парадигмам [25], [26];

– управління пам'яттю. C++ підтримує динамічне виділення пам'яті, яке допомагає звільняти і виділяти пам'ять. Оскільки збірка сміття відсутня, C++ надає програмісту повний контроль над управлінням пам'яттю [25], [26];

– швидкість і потужність. Оскільки C++ є мовою програмування на основі компілятора, не потрібно встановлювати спеціальне середовище виконання під час виконання програми. Отже, вони попередньо інтерпретуються, і це робить код швидшим і потужнішим. Навіть компіляція і виконання виконуються швидше, що дозволяє створювати кілька видів програм - від ігор до драйверів і складних графічних інтерфейсів [25], [26];

– аналогічність з іншими мовами. Синтаксис C++ аналогічний C#, C і Java. Це полегшує вивчення c++, якщо ви вже знаєте один з них. Це також полегшує перемикання на інші мови і з них. Це можна розглядати як додаткову перевагу, що C++ сумісний з програмами на C, тобто кожна запущена програма на C може бути запущена як програма на C++. У більшості випадків нам просто потрібно запустити програму з розширенням .cpp файлу [25], [26];

– стандартні бібліотеки. С++ надає широкий спектр вбудованих бібліотек. Вони допомагають прискорити розробку програмного забезпечення і дозволяють користувачеві робити більше з меншими витратами [25], [26];

– широкий спектр застосування. С++ корисний для створення графічних інтерфейсів, а також ігор. С++ також корисний для розробки графіки та алгебраїчного моделювання в реальному часі. Отже, С++ корисний у кожному потоці [25], [26];

– величезна спільнота. С++ має велику спільноту навколо нього. Розмір спільноти дуже важливий, якщо ви хочете час від часу отримувати підтримку. Чим більше розмір спільноти, тим більше допомоги ви отримаєте у вирішенні своїх проблем. Доступна величезна кількість платних / безкоштовних онлайн-курсів і лекцій, які показують, як працює підтримка спільноти [25], [26];

– масштабованість. Однією з найбільших переваг С++ є її масштабованість, тобто програма на ній може бути масштабована на інший рівень. Отже, ресурсоємні програми можуть бути створені з використанням С++, так як програми можуть бути маломасштабними і великомасштабними [25], [26];

– великий ринок праці. Оскільки знаємо, що С++ має застосування в різних галузях – від фінансів до розробки застосунків, від графічного інтерфейсу до ігор, у С++ дуже великий ринок праці. Знання С++ може допомогти знайти роботу в таких відділах, де С++ стане в нагоді [25], [26].

У табл. 3.2 наведено порівняння мов програмування.

З результатів порівняння можна зробити висновок, що С++ є найбільш збалансованим рішенням при виборі мови програмування для розробки ПЗ для потужних та складних обчислень.

Таблиця 3.2 – Порівняння мов програмування

	Python	Java	C++
Кроссплатформність	Так	Так	Так
Швидкість	Середня	Висока	Найвища
Масштабованість	Середня	Висока	Найвища
Спільнота	Велика	Велика	Велика
Стандартні бібліотеки	Наявні, багато	Наявні	Наявні, багато
Парадигми	багато парадигмальна	Не багато парадигмальна	Багато парадигмальна
Управління пам'яттю	Середнє	Високе	Найбільше

3.3 Опис основних вимог до програми

ПЗ для розпізнавання тривимірних об'єктів на графічних зображеннях призначена для аналізу та дослідження графічних зображень з метою пошуку та розпізнавання складних тривимірних зображень.

Таким чином серед функціональних вимог виділяються:

- завантаження вибірки зображень;
- виконання обробки вибірки;
- реалізація методу навчання нейромоделі;
- запуск та виконання процесів класифікації та розпізнавання тривимірних об'єктів за зображеннями;
- використання паралельних обчислювальних систем для виконання навчання класифікаторів та розпізнавання об'єктів.

Виконання наведених функцій в повній мірі розкриє практичний потенціал запропонованого методу та розроблюваного ПЗ.

3.4 Проектування розробки

На рис. 3.1 наведено принцип загальної організації розроблюваного застосунку із зазначенням рівнів взаємодії системи.

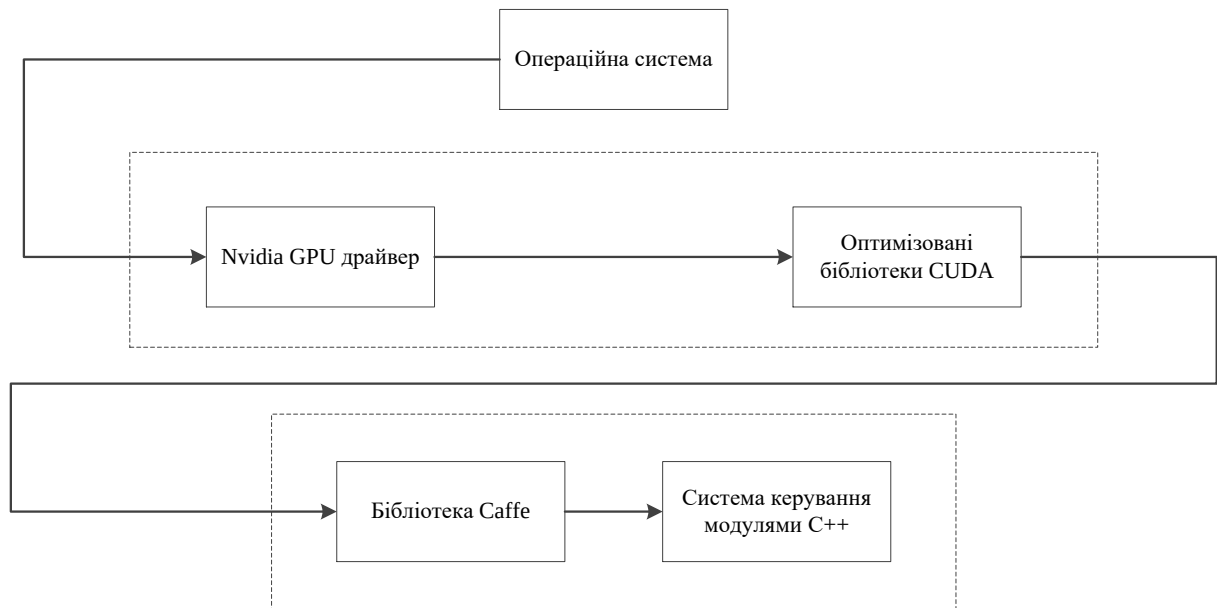


Рисунок 3.1 – Загальна структура організації розроблюваного застосунку

Оскільки планується задіювати паралельні обчислювальні потоки GPU в роботі будуть використовуватися різні ланки подібних систем: починаючи від загальної організації ОС процесів, до використання бібліотек CUDA для спеціалізованого використання потоків GPU.

3.5 Вимоги до технічних засобів

Вимоги до складу і параметрів технічних засобів

Для нормальної роботи програми необхідні комп'ютер і мікрофон.

Технічна характеристика:

- ПК з ОС Microsoft Windows 10 та старше;
- відео карта починаючи від GeForce GTX 980 Ti з підтримкою технології CUDA;
- процесор – Intel (лінійка починаючи з Core i5) або AMD (лінійка Ryzen) з тактовою частотою 3.0 МГц та вище, ядер від: 2, логічних процесорів від: 4.

3.6 Висновки за розділом 3

У третьому розділі було висвітлено питання з програмної реалізації запропонованих методів. В ході роботи було досліджено та обґрунтовано технічні рішення з вибору мови програмування, бібліотеки МН, опису вимог до ПЗ та технічних засобів та платформ для роботи та тестування.

Були спроектовані модулі майбутньої програми та спроектовано взаємодію компонентів ПЗ із обчислювальною системою.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

У третьому розділі було висвітлено питання з програмної реалізації запропонованих методів. Були спроектовані модулі майбутньої програми та спроектовано взаємодію компонентів ПЗ із обчислювальною системою.

Метою четвертого розділу є висвітлення рішень, щодо розробки ПЗ. У розділі описаний поетапний процес розробки програми: опис технічних засобів та платформ для роботи, підготовка навчальних вибірок даних та навчання класифікаторів на базі ШНМ.

4.1 Використовуване обладнання та ПЗ

Розробка всіх основних компонентів системи, реалізація класифікаторів, навчання, тестування і валідація класифікаторів проводилася на високопродуктивному комп'ютері під управлінням ОС Windows 10.

З особливостей обчислювального процесу при використанні нейромережевих класифікаторів, можна виділити вимоги до швидкості багаторазового виконання двомірної згортки, швидкості матричних обчислень і гнучкості паралельних обчислень. Тому явною перевагою при реалізації системи буде використання графічного процесора, зокрема підтримує технологію CUDA. Для навчання, тестування і перевірки використовувався графічний прискорювач на базі Nvidia GPU GeForce GTX 980 ті потужністю 5.6 терафлопс. Докладні технічні характеристики плати представлені в табл. 4.1.

4.2 Вибір моделей для проведення експерименту

В результаті відпрацювання методів на моделях, розуміння всіх деталей формування навчальної вибірки в конкретному випадку, створення

повноцінної системи класифікації за допомогою дешевого з точки зору ресурсів віртуального моделювання, в майбутньому можна визначити успішну стратегію отримання навчальних даних для реальних ситуацій, оптимізувавши процес формування вибірок і мінімізувавши витрати.

Таблиця 4.1 – Технічні характеристики GPU

Компонент	Характеристика
GPU	GeForce GTX 980 Ti
Базова частота відеочіпа	1000 МГц
Обсяг відеопам'яті	6 Гб
Тип пам'яті	GDDR5
Ефективна частота пам'яті	7096 МГц
Розрядність шини пам'яті	384 бит

Для формування вибірки був обраний віртуальний світ багатокористувацької комп'ютерної гри Arma 3 [27], що володіє вбудованим редактором рівнів, що дозволяє формувати вибірку з урахуванням імітації різних факторів в різних конфігураціях. За допомогою редактора Eden Editor віртуальний простір гри наповнюється об'єктами військової техніки та військової інфраструктури, заснованими на реальних прототипах, до того ж існує можливість додаткового завантаження у віртуальний простір сторонніх моделей, що в умовах поставленого завдання надає грі велику цінність з точки зору ресурсу моделювання. Eden Editor дозволяє виробляти гнучку настройку оточення віртуального світу, включаючи настройку часу, особливостей місцевості і погодних умов (рис.2.3).

Після попереднього налаштування оточення в Eden Editor, здійснюється запуск рівня. Існуючі можливості відв'язки камери від персонажів і управління нею в режимі вільного польоту за допомогою джойстика, фіксації камери на цілі. Запис одержуваних даних здійснюється за допомогою програми захоплення. Формування навчального набору шляхом екскапуляції кадрів, проріджування і виділення необхідної області

кадрів здійснювалося за допомогою бібліотеки `ffmpeg`. Використовуючи дані методи, можливо отримати тисячі прикладів для навчальної вибірки на основі моделей протягом годин (рисунк 4.1).

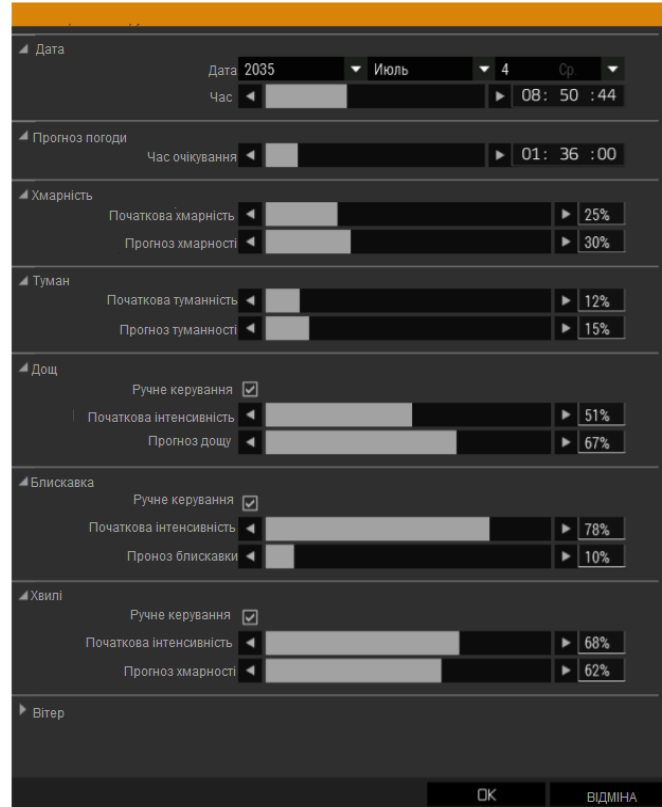


Рисунок 4.1 – Налаштування оточення віртуального простору в Arma 3

На рисунку 4.2 наведено порівняння зображень: праворуч – зображення з борту безпілотної літальної апарату, зліва – зображення з Arma 3.

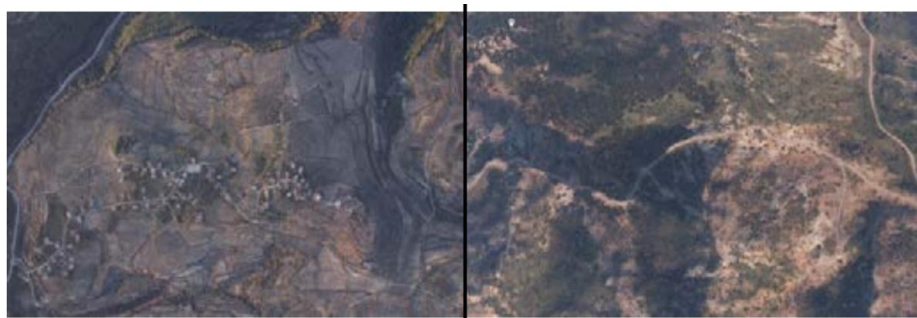


Рисунок 4.2 – Порівняння зображень

4.3 Підготовка навчальної вибірки для ШНМ1

Для експерименту було вирішено використовувати п'ять класів віртуальних об'єктів. Зйомка і кадрування проводилися з варіюванням масштабу і співвідношення об'єкт-оточення (рисунок 4.3). У деяких випадках на зображенні для вибірки присутній тільки частина об'єкта. Нормалізація даних здійснювалася за допомогою Caffe в автоматичному режимі. В результаті додаткових перетворень (трансформації, накладення шуму), розмір навчальної вибірки був збільшений в п'ять разів: з 400 прикладів до 2000 прикладів на об'єкт.

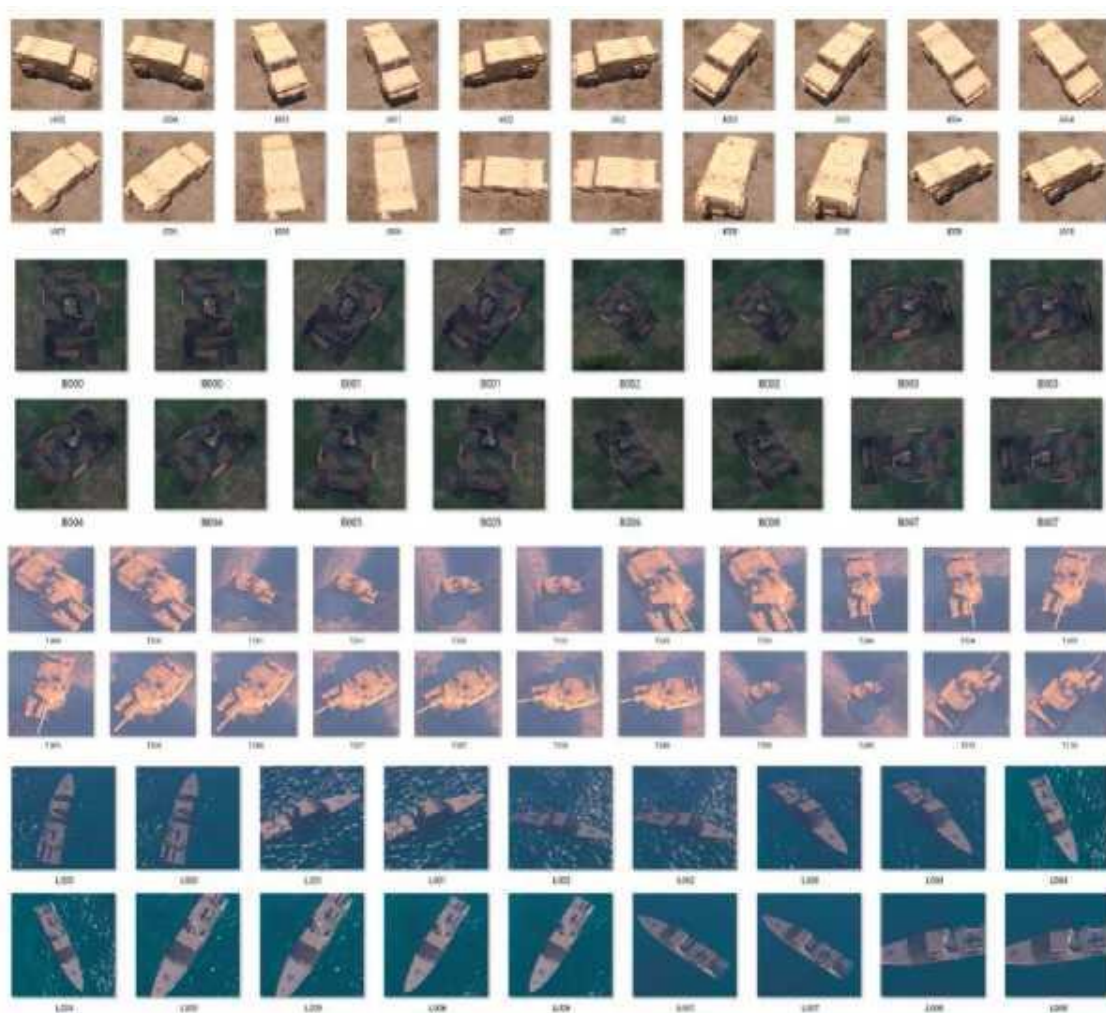


Рисунок 4.3 – Приклад навчальної вибірки

4.4 Підготовка навчальної вибірки для ШНМ2

Навчальна вибірка для ШНМ2 формувалася на основі вибірки для ШНМ1 шляхом вибору різноракурсних знімків одного і того-ж об'єкта і відновлення тривимірного поля точок на основі цих зображень за допомогою Bundler. В результаті прив'язки камери до мети, об'єкт виходить центрованим на проекції і, оскільки зміна ракурсу при зйомці об'єкта здійснюється плавно за допомогою джойстика, досить визначити деякий крок N на масиві кадрів, з яким об'єкт буде відображений з необхідного числа X досить відрізняються ракурсів зйомки. Таким чином, весь процес також автоматизується за допомогою скриптів, а розмір навчальної вибірки для тривимірного випадку зменшується в X разів. В даному випадку розмір навчальної вибірки склав по 400 прикладів на об'єкт.

4.5 Модель Caffe

Елементарною логічною одиницею процесу в Caffe є blob. На рівні представлення blob – це n -мірний масив, що зберігається в пам'яті в C-сумісному стилі. Blob надає уніфікований інтерфейс для зберігання будь-яких даних, необхідних нейромережі: вхідних даних, параметрів моделі, інструментів оптимізації.

Звичайний blob для зберігання вхідних зображень-це чотиривимірний масив, що складається з номера N , каналу K , висоти кадру H і ширини кадру W . Таким чином, в чотиривимірному blob, значення з індексом (n, k, h, w) фізично розташовується за індексом (4.1):

$$\boxed{\begin{matrix} \times \end{matrix}}. \quad (4.1)$$

Канал K – це k -мірний простір ознак, тобто для пікселя RGB-зображення $K = 3$.

Для опису моделей Caffe використовує формат Protocol Buffers. Нижче наведено приклад, що описує згортковий шар, його ім'я, тип, вхід шару, вихід шару і параметри згортки [20]:

```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  learning rate and decay multipliers for the filters
  param { lr mult:1 decay mult: 1 }
  learning rate and decay multipliers for the biases
  param { lr mult: 2 decay mult: 0 }
  convolution param {
    num output: 96
    kernel size:11
    stride: 4
    weight filler {
      type: "gaussian"
      std: 0.01
    }
    bias filler {
      type: "constant"
      value: 0
    }
  }
}
```

Листинг 4.1 – Представление модели слоя в Caffe

4.6 Навчання класифікаторів

Валідаційна вибірка для контролю процесу навчання формувалася виходячи з відношення 1/4 від навчальної вибірки.

Завдяки включенню Dropout-шарів, точність класифікатора ШНМ1 на валідаційній вибірці досягла 91% на 84%, після чого почала знижуватися. Навчання було зупинено на 85 епосі.

Точність класифікатора ШНМ2 на валідаційній вибірці досягла 64% на 26%, після чого почала знижуватися. Навчання було зупинено на 27 епосі. Причини низької точності ШНМ2 на валідаційній вибірці будуть розглянуті далі.

4.7 Висновки за розділом 4

Метою четвертого розділу було висвітлення рішень, щодо розробки ПЗ. У розділі було описано поетапний процес розробки програми: наведено опис технічних засобів та платформ для роботи, підготовка навчальних вибірок даних та навчання класифікаторів на базі ШНМ.

В результаті було розроблено програмну систему для РО на базі МН з використанням гібридних топологій та паралельного підходу. Обрано механізм реалізації відповідних методів, а також елементи керування, з якими користувачу буде зручно використовувати розроблений програмний продукт.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

Метою п'ятого розділу є перевірка роботи розробленої програми:

- надано опис стратегії тестування;
- наведено результати тестування ефективності розробленого ПЗ;
- проведено аналіз отриманих результатів;
- опис стратегій подальшого розвитку та підтримки розробки.

5.1 Результати випробування системи

Контрольна вибірка формувалися виходячи з відношення 1/4 від навчальної вибірки, що з урахуванням додаткових перетворень склало по 500 прикладів на об'єкт для ШНМ1 і 100 прикладів на об'єкт для ШНМ2.

Результати класифікації на контрольній вибірці ШНМ1 виявилися хорошими і цілком передбачуваними. Точність класифікатора склала 91% на контрольній вибірці (рис. 5.1). Даний показник пов'язаний, в першу чергу, з досить якісною навчальною вибіркою і деякою однорідністю середовища, в яку поміщені об'єкти, якщо порівнювати вибірку, представлену в даній роботі, з відомими датасетами, на прикладі яких зазвичай демонструється робота класифікаторів.

Помилка класифікації ШНМ2 для всіх п'яти об'єктів варіювалася в районі 40%, що можна вважати вкрай незадовільним. Даний показник пов'язаний, в першу чергу, з особливостями роботи SFM – методу при обробці зображень, отриманих в умовах слабо вираженої топології, якими є умови зйомки з борту безпілотного літального апарату в польоті.

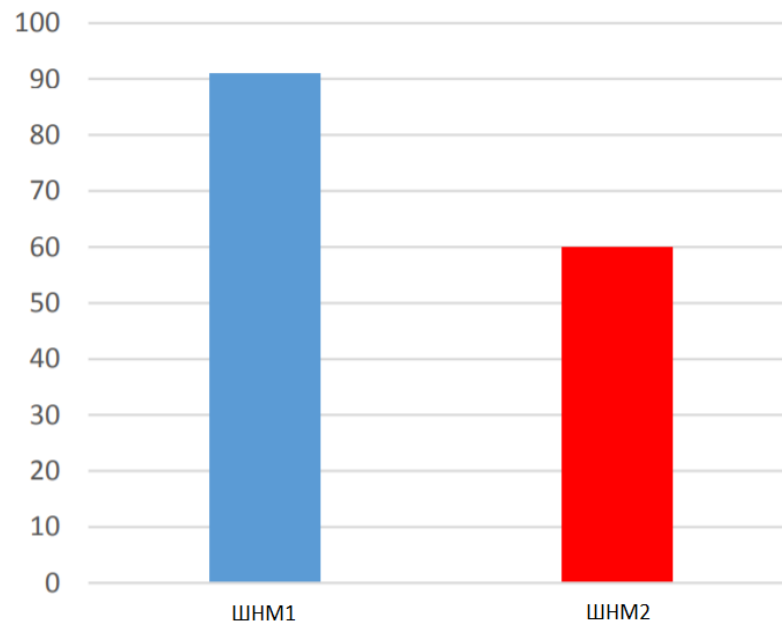


Рисунок 5.1 – Точність розпізнавання на контрольній вибірці

5.2 Аналіз результатів

В ході розробки системи з використанням віртуальних моделей була досягнута точність основного класифікатора, порівнянна з людською точністю. Сучасні класифікатори, що навчаються на базах рукописний символів (MNIST) та інших відомих дата сетах часто перевершують людину в точності і швидкості розпізнавання. Досягнення такого рівня точності мережею з використанням для навчання віртуальних моделей говорить про можливість подальшого успішного розвитку системи.

Нестійкість і низька точність другого класифікатора пов'язана, в першу чергу, з помилками SFM – методу, враховуючи специфіку зображень, одержуваних при зйомці з борту безпілотної літальної апарату. Ця зйомка проходить в умовах слабо вираженої топології, що впливає на результат вже на першому етапі методу - при пошуку корисних для подальшого перетворення Sift – дескрипторів особливих точок. Таким чином, з наступними етапами методу, помилка продовжує прогресивно зростати, тому

Класифікатор не отримує на вхід дійсно адекватної картини даних ні під час навчання, ні на контрольній вибірці.

Подолати зазначену проблему можна двома шляхами: конструктивним і деструктивним.

Конструктивний шлях полягає в тому, щоб використовувати більш підходящий для даного завдання SFM – метод, наприклад, оптичний потік Лукаса-Канаді, що працює з кожним пікселем вихідного зображення, а не тільки зі знайденими особливими точками.

Деструктивний шлях виглядає цікавіше і полягає в тому, щоб повністю змінити існуючий метод: в ролі основного класифікатора може виступати одна нейромережа з кастомізованим інтерфейсом вхідних даних. На основі одержуваної тривимірної структури можна побудувати тривимірний меш, текстурований відповідними цій структурі зображеннями. Після подальших перетворень і нормалізації даних, на вході нейромережі можна отримати реальне зображення, з усіма притаманними йому перевагами точності, що має в кожному пікселі додаткове значення висоти. Подібний підхід може виявитися дуже цікавим, оскільки теоретично володіє високим навчальним потенціалом, завдяки щільності вхідних даних.

5.3 Висновки за результатами тестування

Проведено експериментальну оцінку ефективності методів. Метод РО показав результати на рівні сучасних робіт. Розроблений ПЗ в повній мірі реалізував запропонований метод.

ВИСНОВКИ

Під час виконання дипломної кваліфікаційної роботи магістра, було розглянуто основні методи МН для РО. Для цього було розглянуто та вирішено наступний ряд проблем та задач:

- огляд принципів роботи і сфери застосування сучасних методів МН і РО;

- розробка методу системи на основі ансамблю нейронних мереж. Розробка підсистеми формування тривимірної структури об'єкта на основі детерміністського методу супроводу особливих точок;

- розробка підсистеми розпізнавання на основі двовимірних зображень і тривимірної структури об'єктів з використанням двох навчених штучних багатосарових нейронних мереж згорткового типу, що включає формування навчальних і тестових вибірок, навчання нейромереж.

- проведення випробувань системи розпізнавання на тестових вибірках, дослідження поведінки нейромереж. Представлення експериментальних результатів.

Таким чином, в процесі підготовки дипломної кваліфікаційної роботи магістра була досягнута основна мета роботи і вирішені поставлені завдання

Під час розробки програми та аналізу розробленого програмного комплексу з'явилися ідеї з удосконалення, такі як: використання окремих генеративних ШНМ на базі кластерних систем.

Нові можливості зможуть розширити коло вирішення практичних задач, адже останні роботи засвідчили високий рівень роботи таких нейронних мереж в задачах розпізнавання, прогнозування та аналізу за рахунок пришвидшення роботи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Becker P. W. Recognition of patterns [Text] : using the frequencies of occurrence of binary words / by Peter W. Becker. – København : Polyteknisk Forlag, 1968. – XIII, 216 p.
2. Cakmakov D. Feature selection for pattern recognition [Text] / D. Cakmakov, Y. Bennani. – Scopje : Informa, 2002. – VII, 164 p.
3. Kvetnyy R. N. Textured patterns recognition based on eigen harmonic decomposition [Текст] / R. N. Kvetnyy, O. YU. Sofina, YU. A. Bunyak // Оптико-електронні інформаційно-енергетичні технології. – 2010. – № 1. – С. 90-96.
4. Oliinyk A. O. Parallel multiagent method of big data reduction for pattern recognition [Текст] / A. O. Oliinyk, S. Yu. Skrupsky, V. V. Shkarupylo, O. Yu. Blagodariov // Радіоелектроніка, інформатика, управління. – 2017. – № 2. – С. 82-92.
5. Dmitriev S. Gas turbine engines diagnosing using the methods of pattern recognition [Текст] / S. Dmitriev, O. Popov, O. Yakushenko, V. Potapov, O. Pashchuk // Авиационно-космическая техника и технология. – 2017. – № 8. – С. 115–120.
6. Kapkan O. O. Pattern Recognition: Physical Exercises Modes During Motor Skills Development in Girls Aged 14 [Текст] / O. O. Kapkan, O. M. Khudolii, P. Bartík // Теорія та методика фізичного виховання. – 2018. – Т. 18, № 4. – С. 167-174.
7. Povhan I. General scheme for constructing the most complex logical tree of classification in pattern recognition discrete objects [Текст] / I. Povhan // Електроніка та інформаційні технології. – 2019. – Вип. 11. – С. 74-81.
8. Majdi H. S. Analysis of fault diagnosis of dc motors by power consumption pattern recognition [Текст] / H. S. Majdi, S. S. Shijer, A. O. Hanfesh, L. J. Habeeb, A. H. Sabry // Eastern-european journal of enterprise technologies. – 2021. – № 5(5). – С. 14-20.

9. Вінцюк Т. К. Образний комп'ютер: концепції, методологія, підходи / Т. К. Вінцюк // Оптико-електрон. інформ.-енерг. технології. – 2001. – № 1. – С. 125-138.
10. Gurov S. I. The algorithm for preliminary discrete pattern recognition / S. I. Gurov // Тавр. вестн. інформатики и математики. – 2005. – № 2. – С. 45-50.
11. Glushchenko V. E. Pattern recognition in biological systems of personality authentication / V. E. Glushchenko, Yu. V. Glushchenko // Искусств. интеллект. – 2008. – № 4. – С. 215-219.
12. Тимченко Л. И. Методы обучения параллельно-иерархической сети и их применение для распознавания образов / Л. И. Тимченко, В. В. Мельников, Н. И. Кокряцкая // Кибернетика и систем. анализ. – 2011. – 47, № 6. – С. 175-184.
13. Карпенко О. М. Застосування нейромережевих технологій при інтерпретації даних геофізичних досліджень свердловин / О. М. Карпенко, О. В. Булмасов // Геоінформатика. – 2005. – № 1. – С. 71-79.
14. Geche F. The recognition of discrete patterns and signals in the neural basis / F. Geche, V. Kotsovsky, A. Batiuk // Вісн. Нац. ун-ту "Львів. політехніка". – 2011. – № 719. – С. 269-277.
15. Яровий А. А. Розпізнавання мімічних мікровиразів обличчя людини на основі комбінування Time Delay Neural Network та Deep Belief Network [Текст] / А. А. Яровий, С. Г. Кашубін, О. О. Кулик // Оптико-електронні інформаційно-енергетичні технології. – 2015. – № 1. – С. 76–83.
16. Chumachenko O. I. Deep Learning Classifier Based on NEFPROX Neural Network [Текст] / O. I. Chumachenko // Electronics and control systems. – 2016. – № 4. – С. 63-66.
17. Petrenko A. Detection of human respiration patterns using deep convolution neural networks [Текст] / A. Petrenko, R. Kyslyi, I. Pysmennyi // Восточно-Европейский журнал передовых технологий. – 2018. – № 4(9). – С. 6-13.

18. Tymchenko B. I. Segmentation of Cloud Organization Patterns From Satellite Images using Deep Neural Networks [Текст] / B. I. Tymchenko, Philip O. Marchenko, D. V. Spodarets // Herald of advanced information technology. – 2020. – Vol. 3, no 1. – С. 352-361.

19. Slyusar V. Improvement of the object recognition model on aerophotos using deep convolutional neural network [Текст] / V. Slyusar, M. Protsenko, A. Chernukha, P. Kovalov, P. Borodych, S. Shevchenko, O. Chernikov, S. Vazhynskyi, O. Bogatov, K. Khrustalev // Eastern-european journal of enterprise technologies. – 2021. – № 5(2). – С. 6-21.

20. Паулін О. М. Порівняльний аналіз ефективності методів моделювання розв'язання систем лінійних алгебраїчних рівнянь при використанні кластера та технології NVidia CUDA / О. М. Паулін, І. В. Кривошاپ // Електротехн. и компьютер. системы. – 2012. – № 8. – С. 61-65.

21. Яровий А. А. Комп'ютерне моделювання процесу паралельного оброблення зображень на основі технологій Openmp та Nvidia Cuda [Текст] / А. А. Яровий, О. О. Кулик, І. Р. Арсенюк // Оптико-електронні інформаційно-енергетичні технології. – 2015. – № 2. – С. 33-39.

22. Зори С. А. Синтез изображений рельефов и ландшафтов методом ROAM на кластерной и CUDA- архитектурах вычислительных систем [Текст] / С. А. Зори, В. В. Лисеенко // Наукові праці Донецького національного технічного університету. Серія : Інформатика, кібернетика та обчислювальна техніка. – 2013. – № 1. – С. 57-61.

23. Caffe [Electronic resource]. – Access mode: <https://github.com/Microsoft/caffe>.

24. The Caffe Deep Learning Framework: An Interview with the Core Developers [Electronic resource]. – Access mode: <https://www.edge-ai-vision.com/2016/01/the-caffe-deep-learning-framework-an-interview-with-the-core-developers/>.

25. C++ Language [Electronic resource]. – Access mode:
<https://www.cplusplus.com/doc/tutorial/>.

26. Tutorial 01: Say Hello to CUDA [Electronic resource]. – Access mode:
<https://cuda-tutorial.readthedocs.io/en/latest/tutorials/tutorial01/>.

27. Arma 3 [Electronic resource]. – Access mode:
https://store.steampowered.com/app/107410/Arma_3/.

ДОДАТОК А
ТЕХНІЧНЕ ЗАВДАННЯ

A.1 Вступ

A.1.2 Призначення та область застосування

ПЗ для розпізнавання тривимірних об'єктів на графічних зображеннях призначена для аналізу та дослідження графічних зображень з метою пошуку та розпізнавання складних тривимірних зображень. Існуюча програма реалізована з використанням мови програмування C++, та бібліотеки Caffe.

A.2 Підстави для розробки

Підставою для розробки є завдання на магістерську роботу, яке було визначене за темою «Дослідження та програмна реалізація методів розпізнавання тривимірних об'єктів на основі поля оптичного потоку» і затверджене наказом № 435 від 05 листопада 2021 р. за Національним університетом «Запорізька політехніка».

A.3 Призначення розробки

Програма призначена для розпізнавання тривимірних об'єктів на графічних зображеннях призначена для аналізу та дослідження графічних зображень з метою пошуку та розпізнавання складних тривимірних зображень.

A.4 Вимоги до програми

Програма повинна вирішувати наступні завдання:

- завантаження вибірки зображень;
- виконання обробки вибірки;
- реалізація методу навчання нейромоделі;

- запуск та виконання процесів класифікації та розпізнавання тривимірних об'єктів за зображеннями;
- використання паралельних обчислювальних систем для виконання навчання класифікаторів та розпізнавання об'єктів.

A.4.1 Вимоги до функціональних характеристик

Розробка всіх основних компонентів системи, реалізація класифікаторів, навчання, тестування і валідація класифікаторів проводилася на високо-продуктивному комп'ютері під управлінням операційної системи ОС Windows.

З особливостей обчислювального процесу при використанні нейромережових класифікаторів, можна виділити вимоги до швидкості багаторазового виконання двомірної згортки, швидкості матричних обчислень і гнучкості паралельних обчислень. Тому явною перевагою при реалізації системи буде використання графічного процесора, зокрема підтримує технологію CUDA. Для навчання, тестування і перевірки використовувався графічний прискорювач на базі Nvidia GPU GeForce GTX 980 ti потужністю 5.6 терафлопсів.

A.4.2 Вимоги до надійності

Для сталого функціонування програмного забезпечення необхідно:

- наявність безперебійних блоків живлення персонального комп'ютера.

Програма повинна видавати відмову доступу до комп'ютера при спробі несанкціонованого доступу.

A.4.3 Вимоги до технічних та програмних засобів

Технічна характеристика:

- ПК з ОС Microsoft Windows 10 та старше;
- відео карта починаючи від GeForce GTX 980 Ti з підтримкою технології CUDA;
- процесор – Intel (лінійка починаючи з Core i5) або AMD (лінійка Ryzen) з тактовою частотою 3.0 МГц та вище, ядер від: 2, логічних процесорів від: 4.

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- опис проектування;
- опис тестування;
- опис програми.

A.6 Стадії і етапи розробки

A.6.1 Стадії розробки

Розробка повинна бути проведена в три стадії:

- розробка технічного завдання;
- робоче проектування;
- впровадження.

A.6.2 Етапи розробки

На стадії розробки технічного завдання повинен бути виконаний етап розробки, погодження та затвердження справжнього технічного завдання.

На стадії робочого проєктування повинні бути виконані перераховані нижче етапи робіт [20], [21]:

- розробка програми;
- розробка програмної документації;
- випробування програми.

На стадії впровадження повинен бути виконаний етап розробки підготовка і передача програми [20], [21].

A.6.3 Зміст робіт по етапах

На етапі розробки технічного завдання повинні бути виконані перераховані нижче роботи [20], [21]:

- постановка задачі;
- визначення і уточнення вимог до технічних засобів;
- визначення вимог до програми;
- визначення стадій, етапів і термінів розробки програми та документації на неї [20], [21];
- узгодження і затвердження технічного завдання.

На етапі розробки програми повинна бути виконана робота з програмування (кодування) і налагодженні програми [20], [21].

На етапі розробки програмної документації повинна бути виконана розробка програмних документів відповідно до вимог до складу документації [20], [21].

На етапі випробувань програми повинні бути виконані перераховані нижче види робіт [20], [21]:

- розробка, узгодження та затвердження та методики випробувань;
- проведення приймально-здавальних випробувань;
- коректування програми і програмної документації за результатами випробувань [20], [21].

ДОДАТОК Б
ОПИС ПРОГРАМИ

Б.1 Загальні свідчення

ПЗ для розпізнавання тривимірних об'єктів на графічних зображеннях призначена для аналізу та дослідження графічних зображень з метою пошуку та розпізнавання складних тривимірних зображень. Існуюча програма реалізована з використанням мови програмування C++, та бібліотеки Caffe.

Б.2 Функціональне призначення

Розробка всіх основних компонентів системи, реалізація класифікаторів, навчання, тестування і валідація класифікаторів проводилася на високо-продуктивному комп'ютері під управлінням операційної системи ОС Windows.

З особливостей обчислювального процесу при використанні нейромережевих класифікаторів, можна виділити вимоги до швидкості багаторазового виконання двомірної згортки, швидкості матричних обчислень і гнучкості паралельних обчислень. Тому явною перевагою при реалізації системи буде використання графічного процесора, зокрема підтримує технологію CUDA. Для навчання, тестування і перевірки використовувався графічний прискорювач на базі Nvidia GPU GeForce GTX 980 ті потужністю 5.6 терафлопсів.

Програма повинна вирішувати наступні завдання:

- завантаження вибірки зображень;
- виконання обробки вибірки;
- реалізація методу навчання нейромоделі;
- запуск та виконання процесів класифікації та розпізнавання тривимірних об'єктів за зображеннями;
- використання паралельних обчислювальних систем для виконання навчання класифікаторів та розпізнавання об'єктів.

Б.3 Опис логічної структури

На рис. Б.1 наведено принцип загальної організації розроблюваного застосунку із зазначенням рівнів взаємодії системи.

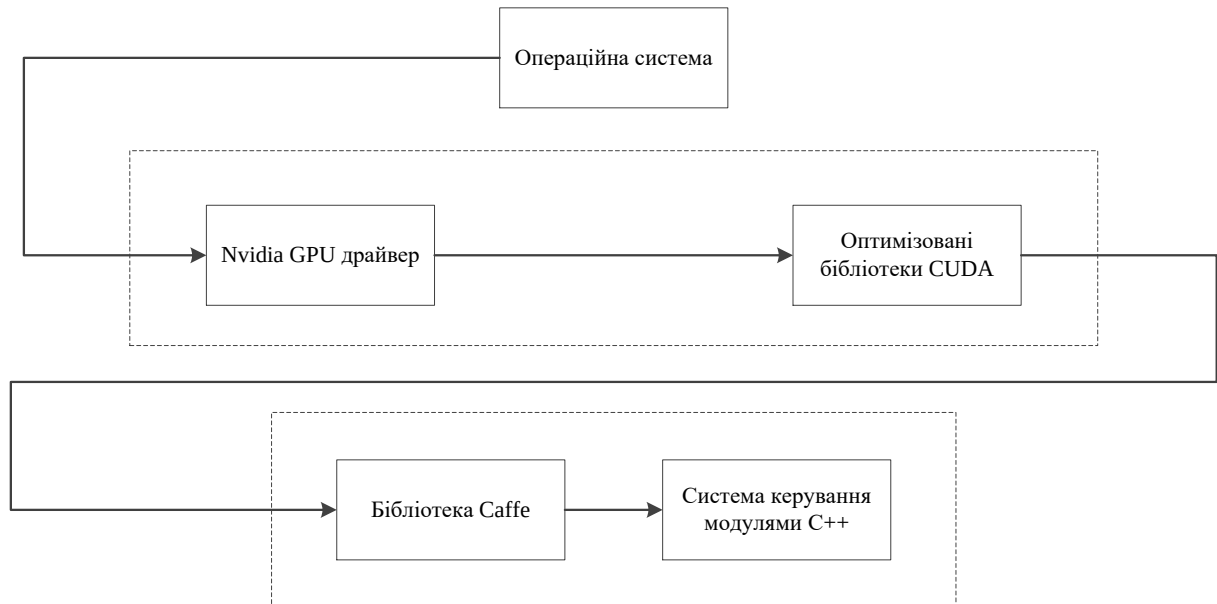


Рисунок Б.1 – Загальна структура організації розроблюваного застосунку

Оскільки планується задіювати паралельні обчислювальні потоки GPU в роботі будуть використовуватися різні ланки подібних систем: починаючи від загальної організації ОС процесів, до використання бібліотек CUDA для спеціалізованого використання потоків GPU.

Б.4 Використані технічні засоби

Технічна характеристика:

- ПК з ОС Microsoft Windows 10 та старше;
- відео карта починаючи від GeForce GTX 980 Ti з підтримкою технології CUDA;
- процесор – Intel (лінійка починаючи з Core i5) або AMD (лінійка Ryzen) з тактовою частотою 3.0 МГц та вище, ядер від: 2, логічних процесорів від: 4.

ДОДАТОК В
ТЕКСТ ПРОГРАМИ

Сирцевий код розробленого застосунку:

```

namespace FW_mod_exec_caffe {

temp <typename Dtype>
SolutResMod<Dtype>::SolutResMod(const SolutResModfeatureter&
featur)
    : modl_() {
    Init(featur);
}

temp <typename Dtype>
SolutResMod<Dtype>::SolutResMod(const string& featur_file)
    : modl_() {
    SolutResModfeatureter featur;
    LoadPhoto_from(featur_file, &featur);
    Init(featur);
}

temp <typename Dtype>
void SolutResMod<Dtype>::Init(const SolutResModfeatureter&
featur) {
    LOG(INFO) << "Initializing SolutResMod from featureters: " <<
std::endl
        << featur.DebugString();
    featur_ = featur;
    CHECK_GE(featur_.average_miss(), 1) << "average_miss should be
non-negative.";
    if (featur_.random_seed() >= 0) {
        FW_mod_exec_caffe::set_random_seed(featur_.random_seed());
    }
#ifdef USE_CUDNN
    FW_mod_exec_caffe::set_cudnn_mem_richness(featur_.richness());
#endif
    InitModTrain();
    InitModest();
    LOG(INFO) << "SolutResMod scaffolding done.";
    iter_ = 0;
    moment_step_ = 0;
}

temp <typename Dtype>
void SolutResMod<Dtype>::InitModTrain() {
    const int num_train_modls = featur_.has_modl() +
featur_.has_modl_featur() +
featur_.has_train_modl() +
featur_.has_train_modl_featur();
    const string& atrib_names = "modl, modl_featur, train_modl,
train_modl_featur";
    CHECK_GE(num_train_modls, 1) << "SolutResModfeatureter must
specify a train modl "
        << "using one of these fields: " << atrib_names;
    CHECK_LE(num_train_modls, 1) << "SolutResModfeatureter must
not contain more than "

```

```

        << "one of these fields specifying a train_modl: " <<
atrib_names;
    modlfeaturer modl_featur;
    if (featur_.has_train_modl_featur()) {
        LOG(INFO) << "Creating training modl specified in
train_modl_featur.";
        modl_featur.dublFrom(featur_.train_modl_featur());
    } else if (featur_.has_train_modl()) {
        LOG(INFO) << "Creating training modl from train_modl file: "
        << featur_.train_modl();
        Read_photo_from(featur_.train_modl(), &modl_featur);
    }
    if (featur_.has_modl_featur()) {
        LOG(INFO) << "Creating training modl specified in
modl_featur.";
        modl_featur.dublFrom(featur_.modl_featur());
    }
    if (featur_.has_modl()) {
        LOG(INFO) << "Creating training modl from modl file: " <<
featur_.modl();
        Read_photo_from(featur_.modl(), &modl_featur);
    }
    modlState modl_state;
    modl_state.set_phase(TRAIN);
    modl_state.MergeFrom(modl_featur.state());
    modl_state.MergeFrom(featur_.train_state());
    modl_featur.change_state()->dublFrom(modl_state);
    modl_.reset(new modl<Dtype>(modl_featur));
}

temp <typename Dtype>
void SolutResMod<Dtype>::InitModest() {
    const bool has_modl_featur = featur_.has_modl_featur();
    const bool has_modl_file = featur_.has_modl();
    const int num_GInputs_modl = has_modl_featur + has_modl_file;
    CHECK_LE(num_GInputs_modl, 1)
        << "Both modl_featur and modl_file may not be specified.";
    const int num_test_modlmodl_featur =
featur_.test_modlmodl_featur_size();
    const int num_test_modlmodl_files =
featur_.test_modlmodl_files_size();
    const int num_test_modlmodls = num_test_modlmodl_featur +
num_test_modlmodl_files;
    if (num_GInputs_modl) {
        CHECK_GE(featur_.test_iter_size(), num_test_modlmodls)
            << "test_iter must be specified for each test
modlwork.";
    } else {
        CHECK_EQ(featur_.test_iter_size(), num_test_modlmodls)
            << "test_iter must be specified for each test
modlwork.";
    }
    const int num_generic_modl_instances =
featur_.test_iter_size() - num_test_modlmodls;
    const int num_test_modlmodl_instances = num_test_modlmodls +
num_generic_modl_instances;
    if (featur_.test_state_size()) {

```

```

        CHECK_EQ(featur_.test_state_size(),
num_test_modlmodl_instances)
        << "test_state must be unspecified or specified once per
test modl.";
    }
    if (num_test_modlmodl_instances) {
        CHECK_GT(featur_.test_interval(), 0);
    }
    int test_modlmodl_id = 0;
    vector<string> sources(num_test_modlmodl_instances);
    vector<modlfeatureter>
modmodl_featur(num_test_modlmodl_instances);
    for (int i = 0; i < num_test_modmodl_featur; ++i,
++test_modlmodl_id) {
        sources[test_modlmodl_id] = "test_modlmodl_featur";

modmodl_featur[test_modlmodl_id].dublFrom(featur_.test_modlmodl_featur
(i));
    }
    for (int i = 0; i < num_test_modlmodl_files; ++i,
++test_modlmodl_id) {
        sources[test_modlmodl_id] = "test_modlmodl file: " +
featur_.test_modlmodl(i);
        Read_photo_from(featur_.test_modlmodl(i),
&modmodl_featur[test_modlmodl_id]);
    }
    const int remaining_test_modlmodls = featur_.test_iter_size()
- test_modlmodl_id;
    if (has_modl_featur) {
        for (int i = 0; i < remaining_test_modlmodls; ++i,
++test_modlmodl_id) {
            sources[test_modlmodl_id] = "modl_featur";

modmodl_featur[test_modlmodl_id].dublFrom(featur_.modl_featur());
        }
    }
    if (has_modl_file) {
        for (int i = 0; i < remaining_test_modlmodls; ++i,
++test_modlmodl_id) {
            sources[test_modlmodl_id] = "modl file: " +
featur_.modl();
            Read_photo_from(featur_.modl(),
&modmodl_featur[test_modlmodl_id]);
        }
    }
    test_modlmodls_.resize(num_test_modlmodl_instances);
    for (int i = 0; i < num_test_modlmodl_instances; ++i) {
        modlState modl_state;
        modl_state.set_phase(TEST);
        modl_state.MergeFrom(modmodl_featur[i].state());
        if (featur_.test_state_size()) {
            modl_state.MergeFrom(featur_.test_state(i));
        }
        modmodl_featur[i].change_state()->dublFrom(modl_state);
        LOG(INFO)
        << "Creating test modl (#" << i << ") specified by " <<
sources[i];

```

```

        test_modlmodls_[i].reset(new
modl<Dtype>(modmodl_featur[i]));
        test_modlmodls_[i]->set_debug_info(featur_.debug_info());
    }
}

temp <typename Dtype>
void SolutResMod<Dtype>::Step(int iters) {
    vector<Blob<Dtype>*> bottom_vec;
    const int start_iter = iter_;
    const int stop_iter = iter_ + iters;
    int average_miss = this->featur_.average_miss();
    vector<Dtype> misses;
    Dtype smoothed_miss = 0;

    while (iter_ < stop_iter) {
        for (int i = 0; i < modl_->features().size(); ++i) {
            shared_ptr<Blob<Dtype> > blob = modl_->features()[i];
            switch (FW_mod_exec_caffe::mode()) {
                case FW_mod_exec_caffe::CPU:
                    FW_mod_exec_caffe_set(blob->num(),
static_cast<Dtype>(0),
                    blob->change_cpu_diff());
                    break;
                case FW_mod_exec_caffe::GPU:
#ifdef CPU_ONLY
                    FW_mod_exec_caffe_gpu_set(blob->num(),
static_cast<Dtype>(0),
                    blob->change_gpu_diff());
#else
                    NO_GPU;
#endif
                    break;
            }
        }

        if (featur_.test_interval() && iter_ %
featur_.test_interval() == 0
            && (iter_ > 0 || featur_.test_initialization())) {
#ifdef USE_MPI
            if
(FW_mod_exec_caffe::parallel_mode()==FW_mod_exec_caffe::MPI){
                SyncData();
            }
#endif
            TestAll();
        }

        const bool display = featur_.display() && iter_ %
featur_.display() == 0;
        modl_->set_debug_info(display && featur_.debug_info());
        Dtype miss = 0;
        for (int i = 0; i < featur_.iter_size(); ++i) {
#ifdef USE_MPI
FW_mod_exec_caffe::set_remaining_sub_iter(featur_.iter_size() - i -
1);
#endif

```

```

        miss += modl_->ForwardBackward(bottom_vec);
    }

    #ifdef USE_MPI
    if (FW_mod_exec_caffe::parallel_mode() ==
FW_mod_exec_caffe::MPI) {
        DLOG(INFO) << "Communication";

        SyncGradient();

        SyncOutput(this->modl_);

        miss = Syncmiss(miss);
    }
    #endif

    miss /= featur_.iter_size();
    if (misses.size() < average_miss) {
        misses.push_back(miss);
        int size = misses.size();
        smoothed_miss = (smoothed_miss * (size - 1) + miss) /
size;
    } else {
        int idx = (iter_ - start_iter) % average_miss;
        smoothed_miss += (miss - misses[idx]) / average_miss;
        misses[idx] = miss;
    }
    if (display) {
        LOG(INFO) << "Iteration " << iter_ << ", miss = " <<
smoothed_miss;
        const vector<Blob<Dtype>*>& result = modl_-
>output_blobs();
        int score_index = 0;
        for (int j = 0; j < result.size(); ++j) {
            const Dtype* res_vector = result[j]->cpu_data();
            const string& output_name =
modl_->blob_names()[modl_-
>output_blob_indices()[j]];
            const Dtype miss_weight =
modl_->blob_miss_weights()[modl_-
>output_blob_indices()[j]];
            for (int k = 0; k < result[j]->num(); ++k) {
                ostream miss_msg_stream;
                if (miss_weight) {
                    miss_msg_stream << " (* " << miss_weight
<< " = " << miss_weight *
res_vector[k] << " miss)";
                }
                LOG(INFO) << "    Train modl output #"
<< score_index++ << ": " << output_name << " = "
<< res_vector[k] << miss_msg_stream.str();
            }
        }
    }
    ApplyUpdate();

    ++iter_;

```

```

        if (featur_.PhotoLand() && iter_ % featur_.PhotoLand() == 0)
        {
            PhotoLand();
            #ifdef USE_MPI
                if (FW_mod_exec_caffe::parallel_mode() ==
FW_mod_exec_caffe::MPI) {
                    MPIComm::Synchrhize();
                    MPI_Barrier(MPI_COMM_WORLD);
                }
            #endif

        }
    }

    #ifdef USE_MPI
        temp <typename Dtype>
        void SolutResMod<Dtype>::SyncGradient() {

            const vector<int>& featur_owners = this->modl_-
>featur_owners();
            const vector<shared_ptr<Blob<Dtype> > >& modmodl_featur =
this->modl_->featur();
            shared_ptr<Layer<Dtype> > layer;
            double t1, t2;
            t1 = MPI_Wtime();

            mpi_force_synchronize();
            for (int featuretr_id = 0; featuretr_id <
modmodl_featur.size(); ++featuretr_id) {
                int featur_owner = featur_owners[featuretr_id];

                bool self_state = (featur_owner == -1);
                layer = this->modl_->layer_by_featur(featuretr_id);
                bool req_synchr = layer->req_synchr();
                if (self_state && req_synchr) {

                    #ifndef CPU_ONLY
                        FW_mod_exec_caffe_gpu_scal(modmodl_featur[featuretr_id]-
>num(),
Dtype(1.)/Dtype(FW_mod_exec_caffe::MPI_all_rank()),
                        modmodl_featur[featuretr_id]-
>change_gpu_diff());
                    #else
                        FW_mod_exec_caffe_scal(modmodl_featur[featuretr_id]-
>num(),
Dtype(1.)/Dtype(FW_mod_exec_caffe::MPI_all_rank()),
                        modmodl_featur[featuretr_id]-
>change_cpu_diff());
                    #endif
                }
            }
            t2 = MPI_Wtime();
            DLOG(INFO) << "Communication time " << t2-t1 << " second";
        }
    }

```

```

temp <typename Dtype>
void SolutResMod<Dtype>::SyncData() {

    const vector<int>& featur_owners = this->modl_-
>featur_owners();
    const vector<shared_ptr<Blob<Dtype> > >& modmodl_featur =
this->modl_->featur();
    double t1, t2;
    t1 = MPI_Wtime();
    for (int featuretr_id = 0; featuretr_id <
modmodl_featur.size(); ++featuretr_id) {
        int featur_owner = featur_owners[featuretr_id];
        bool self_state = (featur_owner == -1);
        if (self_state){
            Dtype* data = modmodl_featur[featuretr_id]-
>change_cpu_data();
            FW_mod_exec_caffe_ibcast(data,
modmodl_featur[featuretr_id]->num());
        }
    }
    mpi_force_synchronize();
    t2 = MPI_Wtime();
    LOG(INFO) << "Model Synchronization Communication time " << t2-
t1 << " second";
}
temp <typename Dtype>
void SolutResMod<Dtype>::SyncOutput(shared_ptr<modl<Dtype> >
modl) {
    const vector<Blob<Dtype>*>& result = modl->output_blobs();
    for (int j = 0; j < result.size(); ++j) {
        FW_mod_exec_caffe_iallreduce<Dtype>(result[j]-
>change_cpu_data(),
            result[j]->num());
    }
    mpi_force_synchronize();
    for (int j = 0; j < result.size(); ++j){
        FW_mod_exec_caffe_scal(result[j]->num(),

Dtype(1.)/Dtype(FW_mod_exec_caffe::MPI_all_rank()),
            result[j]->change_cpu_data());
    }

}
temp <typename Dtype>
Dtype SolutResMod<Dtype>::Syncmiss(Dtype miss){
    Dtype sum_miss;
    FW_mod_exec_caffe_iallreduce<Dtype>(&miss, &sum_miss, 1);
    mpi_force_synchronize();
    return sum_miss / FW_mod_exec_caffe::MPI_all_rank();
}
#endif

temp <typename Dtype>
void SolutResMod<Dtype>::Solve(const char* resume_file) {
    LOG(INFO) << "Solving " << modl->name();
    LOG(INFO) << "Learning Rate Policy: " << featur_.lr_policy();

    if (resume_file) {

```

```

        LOG(INFO) << "Restoring previous SolutResMod status from "
<< resume_file;
        Restore(resume_file);
    }
    #ifdef USE_MPI
    else{
        SyncData();
    }
    #endif

    Step(featur_.max_iter() - iter_);
    if (featur_.PhotoLand_after_train()
        && (!featur_.PhotoLand() || iter_ % featur_.PhotoLand() !=
0)) {

        PhotoLand();
        #ifdef USE_MPI
        if (FW_mod_exec_caffe::parallel_mode() ==
FW_mod_exec_caffe::MPI){
            MPIComm::Synchrhonize();
            MPI_Barrier(MPI_COMM_WORLD);
        }
        #endif
    }
    if (featur_.display() && iter_ % featur_.display() == 0) {
        Dtype miss;
        modl_->ForwardPrefilled(&miss);
        LOG(INFO) << "Iteration " << iter_ << ", miss = " << miss;
    }
    if (featur_.test_interval() && iter_ % featur_.test_interval()
== 0) {
        TestAll();
    }
    LOG(INFO) << "Optimization Done.";
}

temp <typename Dtype>
void SolutResMod<Dtype>::TestAll() {
    for (int test_modlmodl_id = 0; test_modlmodl_id <
test_modlmodls_.size(); ++test_modlmodl_id) {
        Test(test_modlmodl_id);
    }
}

temp <typename Dtype>
void SolutResMod<Dtype>::Test(const int test_modlmodl_id) {
    LOG(INFO) << "Iteration " << iter_
        << ", Testing modl (" << test_modlmodl_id << ")";
    CHECK_NOTNULL(test_modlmodls_[test_modlmodl_id].get())->
        ShareTrainedLayersWith(modl_.get());
    vector<Dtype> test_amount;
    vector<int> test_amount_output_id;
    vector<Blob<Dtype>*> bottom_vec;
    const shared_ptr<modl<Dtype> >& test_modlmodl =
test_modlmodls_[test_modlmodl_id];
    Dtype miss = 0;

```

```

        for (int i = 0; i < featur_.test_iter(test_modlmodl_id); ++i)
        {
            Dtype iter_miss;
            const vector<Blob<Dtype>*>& result =
                test_modlmodl->Forward(bottom_vec, &iter_miss);
#ifdef USE_MPI
                if (FW_mod_exec_caffe::parallel_mode() ==
FW_mod_exec_caffe::MPI) {
                    SyncOutput(test_modlmodl);
                    iter_miss = Syncmiss(iter_miss);
                }
#endif
            if (featur_.test_compute_miss()) {
                miss += iter_miss;
            }
            if (i == 0) {
                for (int j = 0; j < result.size(); ++j) {
                    const Dtype* res_vector = result[j]->cpu_data();
                    for (int k = 0; k < result[j]->num(); ++k) {
                        test_amount.push_back(res_vector[k]);
                        test_amount_output_id.push_back(j);
                    }
                }
            } else {
                int idx = 0;
                for (int j = 0; j < result.size(); ++j) {
                    const Dtype* res_vector = result[j]->cpu_data();
                    for (int k = 0; k < result[j]->num(); ++k) {
                        test_amount[idx++] += res_vector[k];
                    }
                }
            }
            if (featur_.test_compute_miss()) {
                miss /= featur_.test_iter(test_modlmodl_id);
                LOG(INFO) << "Test miss: " << miss;
            }

            for (int i = 0; i < test_amount.size(); ++i) {
                const int output_blob_index =
                    test_modlmodl-
>output_blob_indices()[test_amount_output_id[i]];
                const string& output_name = test_modlmodl-
>blob_names()[output_blob_index];
                const Dtype miss_weight = test_modlmodl-
>blob_miss_weights()[output_blob_index];
                ostringstream miss_msg_stream;
                Dtype mean_score = test_amount[i] /
featur_.test_iter(test_modlmodl_id);

                if (miss_weight) {
                    miss_msg_stream << " (* " << miss_weight
                        << " = " << miss_weight * mean_score << "
miss)";
                }
                LOG(INFO) << "      Test modl output #" << i << ": " <<
output_name << " = "
                    << mean_score << miss_msg_stream.str();
            }
        }
    }

```

```

    }
}

temp <typename Dtype>
void SolutResMod<Dtype>::PhotoLand() {
#ifdef USE_MPI
    if (FW_mod_exec_caffe::MPI_my_rank() != 0) return;
#endif
    modlfeatureter modl_featur;
    modl_->ToProto(&modl_featur, featur_.PhotoLand_diff());
    string filename(featur_.PhotoLand_prefix());
    string model_filename, PhotoLand_filename;
    const int kBufferSize = 20;
    char iter_str_buffer[kBufferSize];
    snprintf(iter_str_buffer, kBufferSize, "_iter_%d", iter_);
    filename += iter_str_buffer;
    model_filename = filename + ".FW_mod_exec_caffemodel";
    LOG(INFO) << "PhotoLandting to " << model_filename;
    WriteProtoToBinaryFile(modl_featur, model_filename.c_str());
    SolutState state;
    PhotoLandSolutState(&state);
    state.set_iter(iter_);
    state.set_learned_modl(model_filename);
    state.set_moment_step(moment_step_);
    PhotoLand_filename = filename + ".SolutState";
    LOG(INFO) << "PhotoLandting SolutResMod state to " <<
PhotoLand_filename;
    WriteProtoToBinaryFile(state, PhotoLand_filename.c_str());
}

temp <typename Dtype>
void SolutResMod<Dtype>::Restore(const char* state_file) {
    SolutState state;
    modlfeatureter modl_featur;
    ReadProtoFromBinaryFile(state_file, &state);
    if (state.has_learned_modl()) {
        ReadmodlfeaturFromBinaryFileOrDie(state.learned_modl().c_str(),
        &modl_featur);
        modl_->dublTrainedLayersFrom(modl_featur);
    }
    iter_ = state.iter();
    moment_step_ = state.moment_step();
    RestoreSolutState(state);
}

temp <typename Dtype>
Dtype SGDSolutResMod<Dtype>::GetLearningRate() {
    Dtype rate;
    const string& lr_policy = this->featur_.lr_policy();
    if (lr_policy == "fixed") {
        rate = this->featur_.base_lr();
    } else if (lr_policy == "step") {
        this->moment_step_ = this->iter_ / this->featur_.stepsize();
        rate = this->featur_.base_lr() *
            pow(this->featur_.gamma(), this->moment_step_);
    } else if (lr_policy == "exp") {

```

```

        rate = this->featur_.base_lr() * pow(this->featur_.gamma(),
this->iter_);
    } else if (lr_policy == "inv") {
        rate = this->featur_.base_lr() *
            pow(Dtype(1) + this->featur_.gamma() * this->iter_,
                - this->featur_.power());
    } else if (lr_policy == "multistep") {
        if (this->moment_step_ < this->featur_.stepvalue_size() &&
            this->iter_ >= this->featur_.stepvalue(this-
>moment_step_)) {
            this->moment_step_++;
            LOG(INFO) << "MultiStep Status: Iteration " <<
                this->iter_ << ", step = " << this->moment_step_;
        }
        rate = this->featur_.base_lr() *
            pow(this->featur_.gamma(), this->moment_step_);
    } else if (lr_policy == "poly") {
        rate = this->featur_.base_lr() * pow(Dtype(1.) -
            (Dtype(this->iter_) / Dtype(this->featur_.max_iter()))),
            this->featur_.power());
    } else if (lr_policy == "sigmoid") {
        rate = this->featur_.base_lr() * (Dtype(1.) /
            (Dtype(1.) + exp(-this->featur_.gamma() * (Dtype(this-
>iter_) -
                Dtype(this->featur_.stepsize())))));
    } else if (lr_policy == "exp10"){
        rate = this->featur_.base_lr()
            * pow(Dtype(10.), Dtype(-1 * this->iter_) /
Dtype(this->featur_.stepsize()));
    } else {
        LOG(FATAL) << "Unknown learning rate policy: " << lr_policy;
    }
    return rate;
}

temp <typename Dtype>
void SGDSolutResMod<Dtype>::PreSolve() {
    // Initialize the history
    const vector<shared_ptr<Blob<Dtype> > >& modmodl_featur =
this->modl_->featur();
    history_.clear();
    update_.clear();
    temp_.clear();
    for (int i = 0; i < modmodl_featur.size(); ++i) {
        const vector<int>& shape = modmodl_featur[i]->shape();
        history_.push_back(shared_ptr<Blob<Dtype> >(new
Blob<Dtype>(shape)));
        update_.push_back(shared_ptr<Blob<Dtype> >(new
Blob<Dtype>(shape)));
        temp_.push_back(shared_ptr<Blob<Dtype> >(new
Blob<Dtype>(shape)));
    }
}

temp <typename Dtype>
void SGDSolutResMod<Dtype>::ClipGradients() {
    const Dtype clip_gradients = this->featur_.clip_gradients();
    if (clip_gradients < 0) { return; }

```

```

        const vector<shared_ptr<Blob<Dtype> > >& modmodl_featur =
this->modl_->featur();
        Dtype sumsq_diff = 0;
        for (int i = 0; i < modmodl_featur.size(); ++i) {
            if (this->modl_->featur_owners()[i] < 0) {
                sumsq_diff += modmodl_featur[i]->sumsq_diff();
            }
        }

        temp <typename Dtype>
        void SGDSolutResMod<Dtype>::ApplyUpdate() {
            Dtype rate = GetLearningRate();
            if (this->featur_.display() && this->iter_ % this-
>featur_.display() == 0) {
                LOG(INFO) << "Iteration " << this->iter_ << ", lr = " <<
rate;
            }
            ClipGradients();
            for (int featuretr_id = 0; featuretr_id < this->modl_-
>featur().size(); ++featuretr_id) {
                Normalize(featuretr_id);
                Regularize(featuretr_id);
                ComputeUpdateValue(featuretr_id, rate);
            }
            this->modl_->Update();
        }

        temp <typename Dtype>
        void SGDSolutResMod<Dtype>::Normalize(int featuretr_id) {
            if (this->featur_.iter_size() == 1) { return; }
            const vector<shared_ptr<Blob<Dtype> > >& modmodl_featur =
this->modl_->featur();
            const Dtype accum_normalization = Dtype(1.) / this-
>featur_.iter_size();
            switch (FW_mod_exec_caffe::mode()) {
                case FW_mod_exec_caffe::CPU: {
                    FW_mod_exec_caffe_scal(modmodl_featur[featuretr_id]->num(),
accum_normalization,
                    modmodl_featur[featuretr_id]->change_cpu_diff());
                    break;
                }
                case FW_mod_exec_caffe::GPU: {
#ifdef CPU_ONLY
                    FW_mod_exec_caffe_gpu_scal(modmodl_featur[featuretr_id]-
>num(), accum_normalization,
                    modmodl_featur[featuretr_id]->change_gpu_diff());
#else
                    NO_GPU;
#endif
                    break;
                }
                default:
                    LOG(FATAL) << "Unknown FW_mod_exec_caffe mode: " <<
FW_mod_exec_caffe::mode();
            }
        }

        temp <typename Dtype>

```

```

void SGDSolutResMod<Dtype>::Regularize(int featuretr_id) {
    const vector<shared_ptr<Blob<Dtype> > >& modmodl_featur =
this->modl_->featur();
    const vector<float>& modmodl_featur_weight_decay =
        this->modl_->featur_weight_decay();
    Dtype weight_decay = this->featur_.weight_decay();
    string regularization_type = this->featur_.regularization_type();
    Dtype local_decay = weight_decay *
modmodl_featur_weight_decay[featuretr_id];
    switch (FW_mod_exec_caffe::mode()) {
    case FW_mod_exec_caffe::CPU: {
        if (local_decay) {
            if (regularization_type == "L2") {
                // add weight decay
                FW_mod_exec_caffe_axpy(modmodl_featur[featuretr_id]-
>num(),
                    local_decay,
                    modmodl_featur[featuretr_id]->cpu_data(),
                    modmodl_featur[featuretr_id]->change_cpu_diff());
            } else if (regularization_type == "L1") {
                FW_mod_exec_caffe_cpu_sign(modmodl_featur[featuretr_id]-
>num(),
                    modmodl_featur[featuretr_id]->cpu_data(),
                    temp_[featuretr_id]->change_cpu_data());
                FW_mod_exec_caffe_axpy(modmodl_featur[featuretr_id]-
>num(),
                    local_decay,
                    temp_[featuretr_id]->cpu_data(),
                    modmodl_featur[featuretr_id]->change_cpu_diff());
            } else {
                LOG(FATAL) << "Unknown regularization type: " <<
regularization_type;
            }
        }
        break;
    }
    case FW_mod_exec_caffe::GPU: {
#ifdef CPU_ONLY
        if (local_decay) {
            if (regularization_type == "L2") {
                // add weight decay
                FW_mod_exec_caffe_gpu_axpy(modmodl_featur[featuretr_id]-
>num(),
                    local_decay,
                    modmodl_featur[featuretr_id]->GPU_in_data(),
                    modmodl_featur[featuretr_id]->change_gpu_diff());
            } else if (regularization_type == "L1") {
                FW_mod_exec_caffe_gpu_sign(modmodl_featur[featuretr_id]-
>num(),
                    modmodl_featur[featuretr_id]->GPU_in_data(),
                    temp_[featuretr_id]->change_GPU_in_data());
                FW_mod_exec_caffe_gpu_axpy(modmodl_featur[featuretr_id]-
>num(),
                    local_decay,
                    temp_[featuretr_id]->GPU_in_data(),
                    modmodl_featur[featuretr_id]->change_gpu_diff());
            } else {

```

```

        LOG(FATAL) << "Unknown regularization type: " <<
regularization_type;
    }
}
#else
    NO_GPU;
#endif
    break;
}
default:
    LOG(FATAL) << "Unknown FW_mod_exec_caffe mode: " <<
FW_mod_exec_caffe::mode();
}
}

temp <typename Dtype>
void SGDSolutResMod<Dtype>::ComputeUpdateValue(int featuretr_id,
Dtype rate) {
    const vector<shared_ptr<Blob<Dtype> > >& modmodl_featur =
this->modl_->featur();
    const vector<float>& modmodl_featur_lr = this->modl_-
>featur_lr();
    Dtype momentum = this->featur_.momentum();
    Dtype loc_ratinge = rate * modmodl_featur_lr[featuretr_id];
    switch (FW_mod_exec_caffe::mode()) {
    case FW_mod_exec_caffe::CPU: {
        FW_mod_exec_caffe_cpu_axpby(modmodl_featur[featuretr_id]-
>num(), loc_ratinge,
                                modmodl_featur[featuretr_id]->cpu_diff(),
momentum,
                                history_[featuretr_id]->change_cpu_data());
        FW_mod_exec_caffe_dubl(modmodl_featur[featuretr_id]->num(),
                                history_[featuretr_id]->cpu_data(),
                                modmodl_featur[featuretr_id]->change_cpu_diff());
        break;
    }
    case FW_mod_exec_caffe::GPU: {
#ifdef CPU_ONLY
        FW_mod_exec_caffe_gpu_axpby(modmodl_featur[featuretr_id]-
>num(), loc_ratinge,
                                modmodl_featur[featuretr_id]->gpu_diff(),
momentum,
                                history_[featuretr_id]->change_GPU_in_data());
        FW_mod_exec_caffe_dubl(modmodl_featur[featuretr_id]->num(),
                                history_[featuretr_id]->GPU_in_data(),
                                modmodl_featur[featuretr_id]->change_gpu_diff());
#else
        NO_GPU;
#endif
    }
    default:
        LOG(FATAL) << "Unknown FW_mod_exec_caffe mode: " <<
FW_mod_exec_caffe::mode();
    }
}

temp <typename Dtype>

```

```

void          SGDSolutResMod<Dtype>::PhotoLandSolutState(SolutState*
state) {
    state->clear_history();
    for (int i = 0; i < history_.size(); ++i) {
        // Add history
        BlobProto* history_blob = state->add_history();
        history_[i]->ToProto(history_blob);
    }
}

temp <typename Dtype>
void SGDSolutResMod<Dtype>::RestoreSolutState(const SolutState&
state) {
    CHECK_EQ(state.history_size(), history_.size())
        << "Incorrect length of history blobs.";
    LOG(INFO) << "SGDSolutResMod: restoring history";
    for (int i = 0; i < history_.size(); ++i) {
        history_[i]->FromProto(state.history(i));
    }
}

temp <typename Dtype>
void          NesterovSolutResMod<Dtype>::ComputeUpdateValue(int
featuretr_id, Dtype rate) {
    const vector<shared_ptr<Blob<Dtype> > >& modmodl_featur =
this->modl_->featur();
    const vector<float>& modmodl_featur_lr = this->modl_-
>featur_lr();
    Dtype momentum = this->featur_.momentum();
    Dtype loc_ratinge = rate * modmodl_featur_lr[featuretr_id];
    switch (FW_mod_exec_caffe::mode()) {
    case FW_mod_exec_caffe::CPU: {
        FW_mod_exec_caffe_dubl(modmodl_featur[featuretr_id]->num(),
            this->history_[featuretr_id]->cpu_data(),
            this->update_[featuretr_id]->change_cpu_data());

        FW_mod_exec_caffe_cpu_axpby(modmodl_featur[featuretr_id]-
>num(), loc_ratinge,
            modmodl_featur[featuretr_id]->cpu_diff(),
momentum,
            this->history_[featuretr_id]->change_cpu_data());

        FW_mod_exec_caffe_cpu_axpby(modmodl_featur[featuretr_id]-
>num(), Dtype(1) + momentum,
            this->history_[featuretr_id]->cpu_data(), -momentum,
            this->update_[featuretr_id]->change_cpu_data());

        FW_mod_exec_caffe_dubl(modmodl_featur[featuretr_id]->num(),
            this->update_[featuretr_id]->cpu_data(),
            modmodl_featur[featuretr_id]->change_cpu_diff());
        break;
    }
    case FW_mod_exec_caffe::GPU: {
#ifdef CPU_ONLY
        FW_mod_exec_caffe_dubl(modmodl_featur[featuretr_id]->num(),
            this->history_[featuretr_id]->GPU_in_data(),
            this->update_[featuretr_id]->change_GPU_in_data());

```

```

        FW_mod_exec_caffe_gpu_axpby(modmdl_featur[featuretr_id]-
>num(), loc_ratinge,
                                modmdl_featur[featuretr_id]->gpu_diff(),
momentum,
                                this->history_[featuretr_id]-
>change_GPU_in_data());

        FW_mod_exec_caffe_gpu_axpby(modmdl_featur[featuretr_id]-
>num(), Dtype(1) + momentum,
                                this->history_[featuretr_id]->GPU_in_data(), -momentum,
                                this->update_[featuretr_id]->change_GPU_in_data());

        FW_mod_exec_caffe_dubl(modmdl_featur[featuretr_id]->num(),
                                this->update_[featuretr_id]->GPU_in_data(),
                                modmdl_featur[featuretr_id]->change_gpu_diff());
    #else
        NO_GPU;
    #endif
        break;
    }
    default:
        LOG(FATAL) << "Unknown FW_mod_exec_caffe mode: " <<
FW_mod_exec_caffe::mode();
    }
}

temp <typename Dtype>
void AdaGradSolutResMod<Dtype>::ComputeUpdateValue(int
featuretr_id, Dtype rate) {
    const vector<shared_ptr<Blob<Dtype> > >& modmdl_featur =
this->modl_->featur();
    const vector<float>& modmdl_featur_lr = this->modl_-
>featur_lr();
    Dtype delta = this->featur_.delta();
    Dtype loc_ratinge = rate * modmdl_featur_lr[featuretr_id];
    switch (FW_mod_exec_caffe::mode()) {
    case FW_mod_exec_caffe::CPU: {
        FW_mod_exec_caffe_powx(modmdl_featur[featuretr_id]->num(),
                                modmdl_featur[featuretr_id]->cpu_diff(), Dtype(2),
                                this->update_[featuretr_id]->change_cpu_data());

        FW_mod_exec_caffe_add(modmdl_featur[featuretr_id]->num(),
                                this->update_[featuretr_id]->cpu_data(),
                                this->history_[featuretr_id]->cpu_data(),
                                this->history_[featuretr_id]->change_cpu_data());

        FW_mod_exec_caffe_powx(modmdl_featur[featuretr_id]->num(),
                                this->history_[featuretr_id]->cpu_data(),
Dtype(0.5),
                                this->update_[featuretr_id]->change_cpu_data());

        FW_mod_exec_caffe_add_scalar(modmdl_featur[featuretr_id]-
>num(),
                                delta,
                                this->update_[featuretr_id]-
>change_cpu_data());

        FW_mod_exec_caffe_div(modmdl_featur[featuretr_id]->num(),
                                modmdl_featur[featuretr_id]->cpu_diff(),

```

```

        this->update_[featuretr_id]->cpu_data(),
        this->update_[featuretr_id]->change_cpu_data());

        FW_mod_exec_caffe_cpu_axpby(modmodl_featur[featuretr_id]-
>num(), loc_ratinge,
        this->update_[featuretr_id]->cpu_data(), Dtype(0),
        modmodl_featur[featuretr_id]->change_cpu_diff());
        break;
    }
    case FW_mod_exec_caffe::GPU: {
#ifdef CPU_ONLY
        FW_mod_exec_caffe_gpu_powx(modmodl_featur[featuretr_id]-
>num(),
        modmodl_featur[featuretr_id]->gpu_diff(), Dtype(2),
        this->update_[featuretr_id]->change_GPU_in_data());

        FW_mod_exec_caffe_gpu_add(modmodl_featur[featuretr_id]-
>num(),
        this->update_[featuretr_id]->GPU_in_data(),
        this->history_[featuretr_id]->GPU_in_data(),
        this->history_[featuretr_id]->change_GPU_in_data());

        FW_mod_exec_caffe_gpu_powx(modmodl_featur[featuretr_id]-
>num(),
        this->history_[featuretr_id]->GPU_in_data(),
Dtype(0.5),
        this->update_[featuretr_id]-
>change_GPU_in_data());

        FW_mod_exec_caffe_gpu_add_scalar(modmodl_featur[featuretr_id]->num(),
        delta,
        this->update_[featuretr_id]-
>change_GPU_in_data());

        FW_mod_exec_caffe_gpu_div(modmodl_featur[featuretr_id]-
>num(),
        modmodl_featur[featuretr_id]->gpu_diff(),
        this->update_[featuretr_id]->GPU_in_data(),
        this->update_[featuretr_id]-
>change_GPU_in_data());

        FW_mod_exec_caffe_gpu_axpby(modmodl_featur[featuretr_id]-
>num(), loc_ratinge,
        this->update_[featuretr_id]->GPU_in_data(), Dtype(0),
        modmodl_featur[featuretr_id]->change_gpu_diff());
    #else
        NO_GPU;
    #endif
        break;
    }
    default:
        LOG(FATAL) << "Unknown FW_mod_exec_caffe mode: " <<
FW_mod_exec_caffe::mode();
    }
}

```

ДОДАТОК Д
СЛАЙДИ ПРЕЗЕНТАЦІЇ

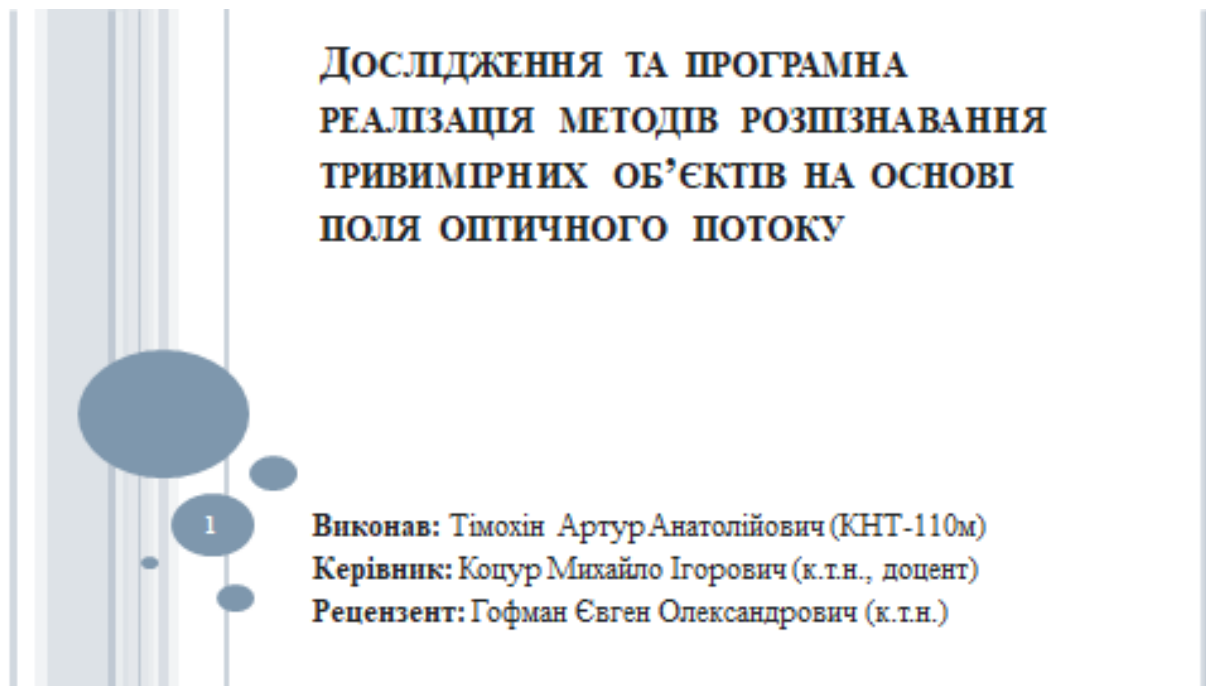


Рисунок Д.1 – Слайд №1

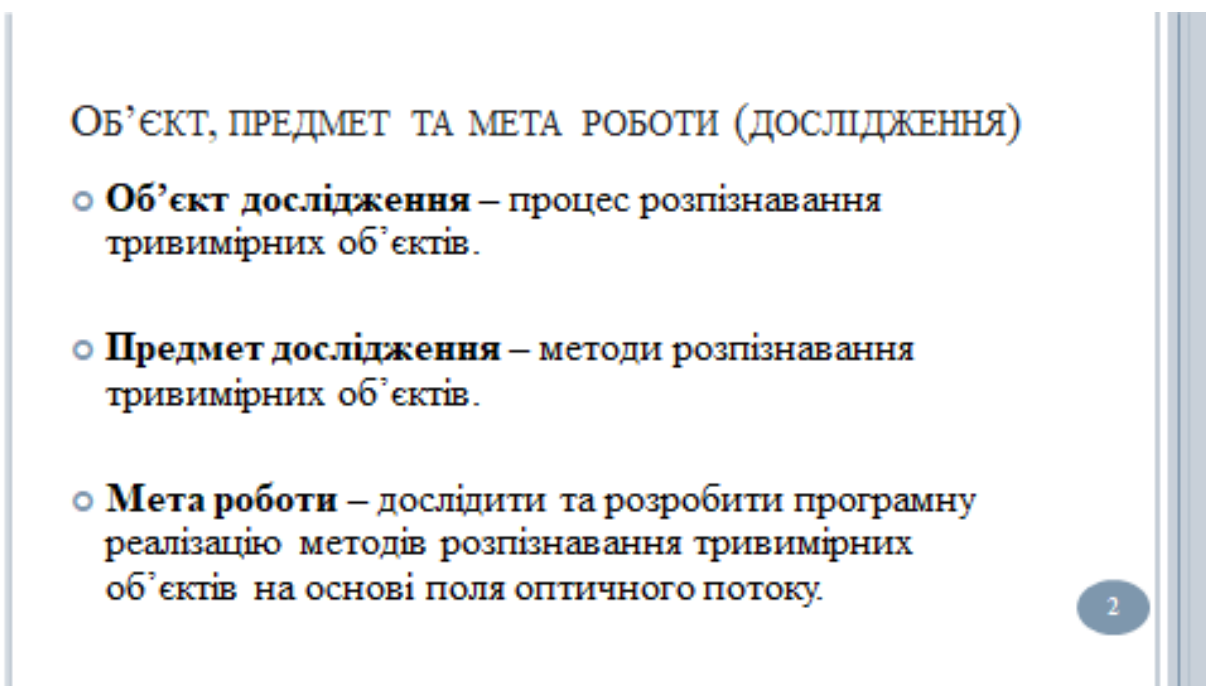


Рисунок Д.2 – Слайд №2

ПОСТАНОВКА ЗАВДАННЯ

- Дослідження та програмна реалізація методів розпізнавання тривимірних об'єктів на основі поля оптичного потоку для інтегрованої автоматизованої гібридної системи виявлення об'єктів при аналізі зображень з безпілотних літаючих пристроїв, наприклад, квадрокоптерів.
- Подібне забезпечення може використовуватися для різних галузей людської діяльності в тому числі для логістики, військової справи та охорони довкілля.

3

Рисунок Д.3 – Слайд №3

ПОРІВНЯННЯ СУЧАСНИХ ТОПОЛОГІЙ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ЗАДАЧ КОМП'ЮТЕРНОГО ЗОРУ

	Багатошарові перцептрони	Рекурентні ШНМ	Глибокі ШНМ	Гібридні види глибоких ШНМ
Швидкість навчання	Висока	Середня	Низька	Середня
Якість розпізнавання	Середня	Висока	Висока	Висока
Методи навчання	Класичні	Класичні	Класичні	Класичні
Підтримка фреймворках МН ^у	Підтримується	Підтримується	Підтримується	Підтримується (за умови)

4

Рисунок Д.4 – Слайд №4

ПОРІВНЯННЯ БІБЛІОТЕК ГЛИБОКОГО НАВЧАННЯ

	Torch	Theano	Caffe
Підтримка мов програмування	C++, Lua	Python	C/C++, Python, Matlab
Підтримка моделей нейронних мереж	Середня	Висока	Найвища
Швидкість навчання моделей	Середня	Середня (для глибоких ШНМ збільшується)	Найвища
Паралелізація обчислень	Можлива	Можлива	Можлива
Підтримка паралельних систем на базі GPU	CUDA	CUDA (наявні обмеження)	CUDA та технологія multi-GPU
Спільнота	Менш активна	Активна	Найбільша та найактивніша

5

Рисунок Д.5 – Слайд №5

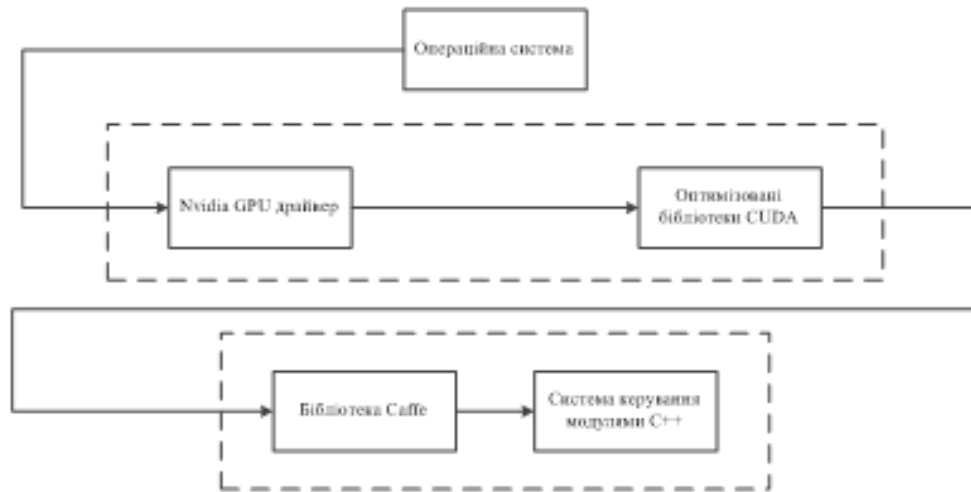
ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ

	Python	Java	C++
Кроссплатформність	Так	Так	Так
Швидкість	Середня	Висока	Найвища
Масштабованість	Середня	Висока	Найвища
Спільнота	Велика	Велика	Велика
Стандартні бібліотеки	Наявні, багато	Наявні	Наявні, багато
Парадигми	багато парадигмальна	Не багато парадигмальна	Багато парадигмальна
Управління пам'яттю	Середнє	Високе	Найбільше

6

Рисунок Д.6 – Слайд №6

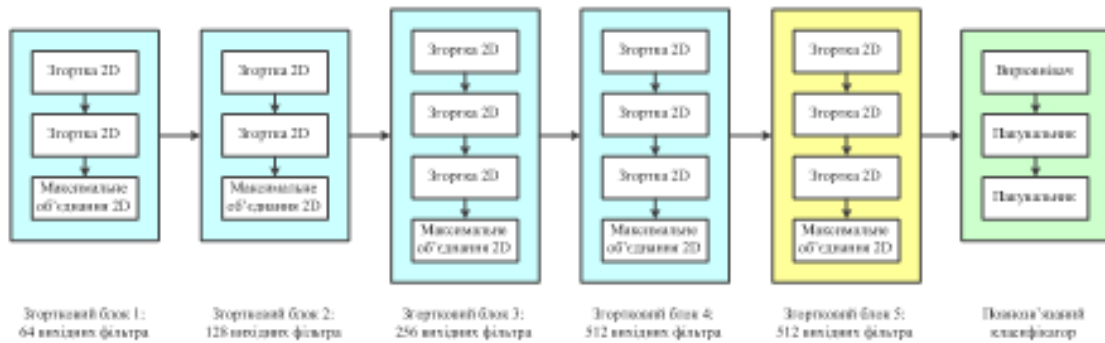
ПРИНЦИП ЗАГАЛЬНОЇ ОРГАНІЗАЦІЇ РОЗРОБЛЮВАНОГО ЗАСТОСУНКУ ІЗ ЗАЗНАЧЕННЯМ РІВНІВ ВЗАЄМОДІЇ СИСТЕМИ



7

Рисунок Д.7 – Слайд №7

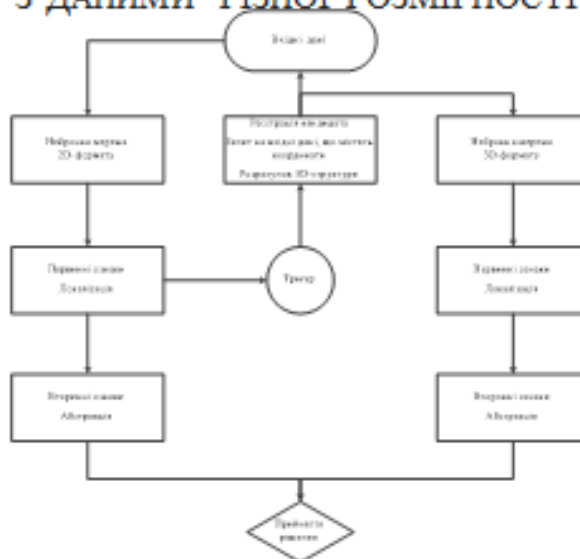
ВИБІР АРХІТЕКТУРИ НЕЙРОМЕРЕЖ



8

Рисунок Д.8 – Слайд №8

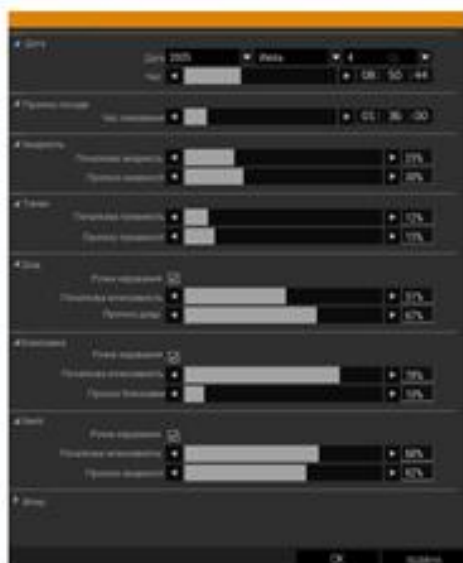
ПАРАЛЕЛЬНИЙ МЕТОД З ДВОХ ШІМ, ЩО ПРАЦЮЮТЬ З ДАНИМИ РІЗНОЇ РОЗМІРНОСТІ



9

Рисунок Д.9 – Слайд №9

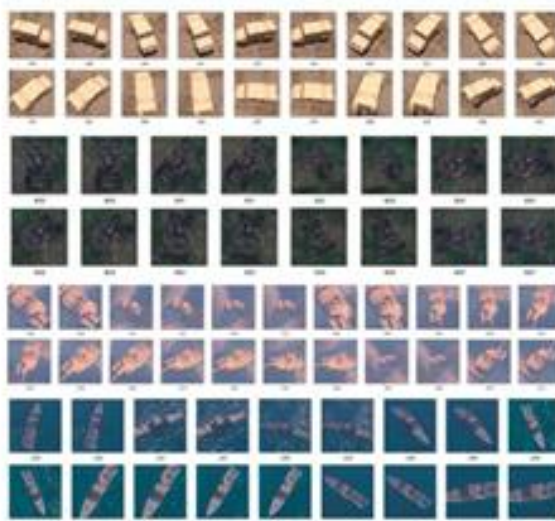
ПІДГОТОВКА ВХІДНИХ ДАНИХ



10

Рисунок Д.10 – Слайд №10

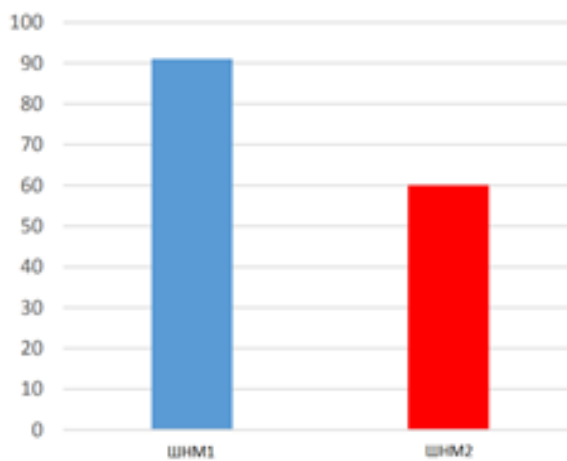
ПРИКЛАД НАВЧАЛЬНОЇ ВИБІРКИ



11

Рисунок Д.11 – Слайд №11

ТОЧНІСТЬ РОЗПІЗНАВАННЯ НА КОНТРОЛЬНІЙ ВИБІРЦІ



12

Рисунок Д.12 – Слайд №12

ВИСНОВКИ

Під час виконання дипломної кваліфікаційної роботи магістра, було розглянуто основні методи машинного навчання для розпізнавання образів. Для цього було розглянуто та вирішено наступний ряд проблем та задач:

- огляд принципів роботи і сфери застосування сучасних методів машинного навчання і розпізнавання образів;
- розробка методу системи на основі ансамблю нейронних мереж. Розробка підсистеми формування тривимірної структури об'єкта на основі детерміністського методу супроводу особливих точок;
- розробка підсистеми розпізнавання на основі двовимірних зображень і тривимірної структури об'єктів з використанням двох навчених штучних багаторішарових нейронних мереж згорткового типу, що включає формування навчальних і тестових вибірок, навчання нейромереж.
- проведення випробувань системи розпізнавання на тестових вибірках, дослідження поведінки нейромереж. Представлення експериментальних результатів.

13

Рисунок Д.13 – Слайд №13

ДЯКУЮ ЗА УВАГУ!

14

Рисунок Д.14 – Слайд №14