ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка алгоритмів машинного навчання для використання в інформаційних системах»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126-Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми бакалавра
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Єгор ДЕМЧЕНКО
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. ІСД-41
Єгор ДЕМЧЕНКО
Ім'я, ПРІЗВИЩЕ

Керівник: *PhD* Валентина ДАНИЛЬЧЕНКО
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем
Ступінь вищої освіти бакалавр
Спеціальність Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС
_____ Каміла СТОРЧАК
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
Демченко Єгору Юрійовичу
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Розробка алгоритмів машинного навчання для використання в інформаційних системах

керівник кваліфікаційної роботи Валентина ДАНИЛЬЧЕНКО *PhD*
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «___» _____ 2024 р. № ___

2. Строк подання кваліфікаційної роботи «___» _____ 2024 р.

3. Вихідні дані до кваліфікаційної роботи:
1. Науково-технічна література з теми бакалаврської роботи.
2. Принципи застосування діаграм в інформаційних системах.
3. Принципи та методи використання машинного навчання інформаційних системах.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Визначення інформаційних систем та її складових.
2. Визначення перспектив у застосування UML діаграм в інформаційних системах.
3. Розробка та демонстрація використання алгоритму машинного навчання для обробки та аналізу UML діаграм.

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання: «___» _____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Постановка мети дослідження	27.02-01.03.2024	
2	Аналіз наявної науково-технічної літератури	02.03-14.03.2024	
3	Дослідження та огляд сфери інформаційних систем та її складових	15.03-28.03.2024	
4	Дослідження застосування різних типів діаграм в інформаційних системах	29.04-08.04.2024	
5	Дослідження методів машинного навчання та технології комп'ютерного зору для аналізу діаграм	09.04-23.04.2024	
6	Розробка алгоритму машинного навчання з використанням технології комп'ютерного зору для аналізу UML діаграм класів	24.04-14.05.2024	
7	Оформлення роботи: вступ, висновки, реферат	15.05-21.05.2024	
8	Розробка демонстраційних матеріалів	22.05-24.05.2024	

Здобувач(ка) вищої освіти _____ Єгор ДЕМЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Валентина ДАНИЛЬЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 50 стор., 27 рис., 24 джерел.

Мета роботи – дослідження та розробка алгоритму для обробки і аналізу діаграм у інформаційних системах з метою автоматизації збору даних для максимальної оптимізації процесів.

Об'єкт дослідження – аналіз архітектури інформаційних систем, а саме UML діаграм, які використовуються як інструмент для моделювання у сфері програмної інженерії і розробки різних типів комп'ютеризованих систем.

Предмет дослідження – використання машинного навчання для обробки UML діаграм, а саме технології комп’ютерного зору та згорткових нейронних мереж.

Короткий зміст роботи: В дослідженні було розглянуто перспективи використання машинного навчання і технології комп’ютерного зору для аналізу архітектури UML діаграм. Метою була автоматизація та оптимізація обробки інформації використовуючи алгоритм машинного навчання. Використовуючи матеріали дослідження та засвоєні знання був створений алгоритм машинного навчання на основі технології комп’ютерного зору, який показав гарний результат і підтвердив свою ефективність. В роботі розглядаються існуючі інструменти для зображення архітектури інформаційних систем, їх перспективи, актуальність та обмеження. Особлива увага приділена перспективі використання згорткових нейронних мереж в UML діаграмах.

Проведене дослідження демонструє, що застосування загорткових нейронних мереж для аналізу UML діаграм автоматизує і значно підвищує ефективність обробки інформації, оптимізуючи обробку інформації. Використання алгоритму машинного навчання та технології комп’ютерного зору дозволяє оптимізувати обробку даних та має перспективи на подальше вдосконалення.

Ключові слова: ІНФОРМАЦІЙНІ СИСТЕМИ, UML ДІАГРАМИ, МАШИННЕ НАВЧАННЯ, ГЛИБОКЕ НАВЧАНННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, ТЕХНОЛОГІЯ КОМП’ЮТЕРНОГО ЗОРУ.

ABSTRACT

The text part of the qualification work for the bachelor's degree: 50 pages, 27 figures, 24 sources.

The purpose of the work – research and develop an algorithm for processing and analyzing diagrams in information systems in order to automate data collection for maximum process optimization.

Object of research – the analysis of the architecture of information systems, namely UML diagrams, which are used as a tool for modeling in the field of software engineering and the development of various types of computerized systems.

Subject of research - the use of machine learning for processing UML diagrams, namely computer vision and convolutional neural network technologies.

Summary of the work: The study examined the prospects of using machine learning and computer vision technology to analyze the architecture of UML diagrams. The goal was to automate and optimize information processing using a machine learning algorithm. Using the research materials and the acquired knowledge, a machine learning algorithm based on computer vision technology was created, which showed a good result and confirmed its effectiveness. The paper discusses the existing tools for depicting the architecture of information systems, their prospects, relevance and limitations. Particular attention is paid to the prospect of using convolutional neural networks in UML diagrams.

The study demonstrates that the use of convolutional neural networks for analyzing UML diagrams automates and significantly increases the efficiency of information processing by optimizing information processing. The use of a machine learning algorithm and computer vision technology allows optimizing data processing and has prospects for further improvement.

KEYWORDS: INFORMATION SYSTEMS, UML DIAGRAMS, MACHINE LEARNING, DEEP LEARNING, CONVOLUTIONAL NEURAL NETWORK, COMPUTER VISION TECHNOLOGY.

ЗМІСТ

ВСТУП

Актуальність теми. Представлений проект вирішує актуальну проблему в області інформаційних систем та розробки програмного забезпечення, сприяючи створенню ефективних інструментів для аналізу вимог та реалізації процесів розробки програмного забезпечення з використанням технологій машинного навчання та комп'ютерного зору.

Мета і завдання дослідження. Метою цієї роботи є створення ефективного інструменту для аналізу вимог до програмного забезпечення та підвищення продуктивності процесів розробки. Впровадження передових технологій машинного навчання та комп'ютерного зору дозволить автоматизувати багато рутинних завдань, що звільнить час та ресурси для більш глибокого аналізу та творчої роботи.

завдання:

1. Дослідження передових технологій комп'ютерного зору та нейронних мереж і їх застосування для аналізу діаграм класів UML.

2. Розробка моделей машинного навчання для ефективного перетворення намальованих діаграм класів в цифровий формат та виділення специфікацій вимог.

3. Використання навчених нейронних мереж для розпізнавання шаблонів у діаграмах та автоматизованого виділення важливої інформації.

4. Проведення експериментів для оцінки ефективності та точності розроблених моделей та алгоритмів.

5. Підготовка звіту про результати дослідження та розробки, включаючи опис методології, отримані результати, аналіз та висновки.

Об'єкт дослідження. Аналіз архітектури інформаційних систем з використанням машинного навчання та технології комп’ютерного зору.

Предмет дослідження. Машинне навчання для обробки UML діаграм, а саме діаграм класів та архітектури системи, яку вона може відображати

Методи дослідження. У цій роботі буде розглянута можливість застосування технологій комп'ютерного зору та нейронних мереж для автоматизації процесу аналізу діаграм класів UML. Моделі машинного навчання можуть

використовуватися для ефективного перетворення у цифровий формат та виділення специфікацій вимог, закладених у цих діаграмах. Це дозволить нам удосконалити процес аналізу вимог до програмного забезпечення шляхом автоматичного виділення та перевірки повноти цих вимог.

Практичне значення одержаних результатів. Планується використання навчених нейронних мереж для розпізнавання шаблонів у діаграмах та автоматизованого виділення важливої інформації. Цей підхід може значно полегшити роботу аналітиків та розробників, дозволяючи їм швидше та ефективніше перевіряти відповідність програмного забезпечення вимогам. Крім того, впровадження такого рішення сприятиме автоматизації процесів аналізу та підвищенню продуктивності розробки програмного забезпечення.

Апробація:

І Всеукраїнська науково-технічна конференція "Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу " , 28 листопада 2023 року, ДУІКТ – «Застосування штучного інтелекту для підвищення безпеки дорожнього руху»

РОЗДІЛ 1. ОГЛЯД ОСНОВНИХ ПОЛОЖЕНЬ ІНФОРМАЦІЙНИХ СИСТЕМ І МАШИННОГО НАВЧАННЯ

1.1 Основні положення і способи застосування інформаційних систем

1.1.1 Основні положення інформаційних систем

Інформаційні системи – це взаємопов’язані компоненти, які взаємодіють для збору, обробки, зберігання та поширення інформації, підтримуючи прийняття рішень, координацію, контроль, аналіз та візуалізацію в організації.

Інформаційні системи можна розглядати як такі, що мають п'ять основних компонентів: апаратне забезпечення, програмне забезпечення, дані, люди та процеси. До компонентів інформаційних систем входить апаратне забезпечення, програмне забезпечення і дані. Також вони підпадають під категорію технологій. Технологія може розглядатися як застосування наукових знань для практичних цілей. Люди і процеси, відокремлюють ідею інформаційних систем від більш технічних галузей, таких як комп'ютерні науки.

Апаратне забезпечення - це конкретні, матеріальні складові інформаційної системи, які можна фізично відчути або доторкнутися до них. Прикладами такого обладнання є комп'ютери, клавіатури, дискові та флеш-накопичувачі.

Програмне забезпечення складається з набору інструкцій, які керують роботою апаратного забезпечення, але саме за собою не мають фізичної форми. Програмісти створюють програмне забезпечення, розробляючи послідовність інструкцій, які вказують апаратному забезпеченню, як виконувати певні завдання. Існують дві основні категорії програмного забезпечення: операційні системи та прикладне програмне забезпечення. Операційні системи, такі як Microsoft Windows та Ubuntu, Linux для персональних комп'ютерів або Google, Android та Apple iOS для мобільних пристроїв, забезпечують інтерфейс між апаратним забезпеченням та програмними додатками. Прикладне програмне забезпечення, навпаки, дозволяє користувачам виконувати конкретні завдання, такі як створення документів, редагування електронних таблиць.

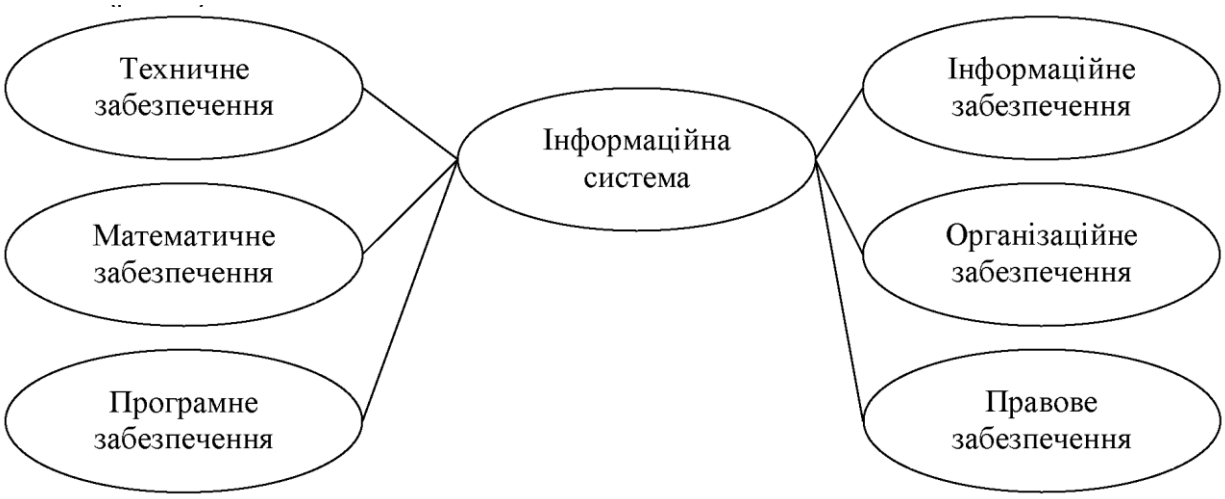


Рис. 1.1 – Структура і підсистеми інформаційної системи [5]

Третім компонентом технології є дані, які можна розглядати як набір фактів, таких як адреса, номер телефону або дані облікового запису у соціальній мережі. Подібно до програмного забезпечення, дані також є нематеріальними і не можуть бути сприйняті в їхньому природному вигляді. Розділені та неорганізовані дані мають обмежену корисність, але коли вони агреговані, індексовані та організовані в базі даних, вони стають потужним інструментом для бізнесу. Організації збирають різноманітні дані і використовують їх для прийняття рішень, які потім можна проаналізувати для оцінки їхньої ефективності. Аналіз даних використовується для покращення продуктивності та розвитку бізнесу.

Поміж вже зазначених компонентів, таких як апаратне забезпечення, програмне забезпечення та дані, було запропоновано додати ще один - комунікацію. Раніше інформаційні системи могли функціонувати самостійно, без можливості спілкування, наприклад, перші персональні комп'ютери були автономними, не підключеними до Інтернету. Однак у сучасному світі, який є вкрай зв'язаним, комп'ютери нечасто працюють в ізоляції, вони майже завжди з'єднані з іншими пристроями або мережами. Хоча з технічної точки зору мережевий комунікаційний компонент складається з апаратного та програмного забезпечення, він є настільки важливим для сучасних інформаційних систем, що став окремою категорією.

При розгляді інформаційних систем легко відволіктись на технологічних компонентах і втратити з поля зору їхню інтеграцію в організацію. Проте важливо

вийти за межі цих інструментів, щоб повністю зрозуміти, як вони взаємодіють з організацією. Сконцентрування на людях, які працюють з інформаційними системами, є наступним кроком. Від співробітників служби підтримки користувачів і системних аналітиків до розробників та директорів - всі вони є важливими учасниками, пов'язаними з інформаційними системами.

Останній компонент інформаційних систем - це процес. Процес визначається як послідовність кроків, спрямованих на досягнення конкретного результату або мети. Інформаційні системи стають все більш інтегрованими з організаційними процесами, що призводить до підвищення продуктивності і кращого контролю над ними. Проте автоматизація діяльності за допомогою технологій - лише початок. Компанії, що прагнуть ефективно використовувати інформаційні системи, повинні зосередитися на вдосконаленні процесів, як внутрішніх, так і зовнішніх, з метою покращення взаємодії з постачальниками та клієнтами. Такі поняття, як "реінжиніринг бізнес-процесів", "управління бізнес-процесами" та "планування ресурсів підприємства", пов'язані з постійним вдосконаленням цих бізнес-процесів і їхньою інтеграцією з технологіями. Компанії, які мають намір зберегти конкурентну перевагу, приділяють значну увагу цьому компоненту інформаційних систем.

1.1.2 Моделювання процесів в інформаційних системах

Чим ефективніші процеси, тим більш успішним є бізнес. Деякі компанії розглядають свої процеси як можливість отримати конкурентну перевагу. Процес, який досягає своєї мети унікальним способом, може виділити компанію серед інших. Також, процес, що оптимізує витрати, дозволяє компанії знизити ціни або збільшити прибуток. У контексті інформаційних систем, бізнес-процес представляє собою набір дій, які здійснюються людьми або інформаційною системою з метою досягнення певного результату.

Процес - це послідовність дій, які виконуються для досягнення певної цілі. У бізнесі це означає, що бізнес-процес спрямований на досягнення певної мети в рамках підприємства. Процеси представляють собою те, через що підприємство проходить щодня для виконання своєї місії.

Інструмент діаграмування для документування бізнес-процесів представляє собою стандартизовану візуальну мову, яка дозволяє системним аналітикам чітко описувати бізнес-процеси, зобразити їх для кращого розуміння та комунікації для успішного управління. Природні мови, такі як англійська, не завжди можуть належним чином пояснити складні бізнес-процеси. Діаграми використовуються як засіб моделювання бізнес-процесів у сфері інформаційних систем. Існує різноманітні типи інструментів, кожен з яких має свої особливості, стиль та синтаксис для досягнення конкретних цілей. Найпоширенішими серед них є Business Process Modeling Notation (BPMN), діаграми потоків даних (DFD) та Уніфікована мова моделювання (UML).

BPMN – містить умовні символи та їх опис у форматі XML для представлення бізнес-процесів у вигляді діаграм. BPMN спрямована на аудиторію, що складається як з технічних спеціалістів, так і з бізнес-користувачів. BPMN розширює традиційний підхід блок-схем, додаючи більше елементів для діаграм для опису бізнес-процесів і маючи за мету підтримувати документування цих процесів шляхом використання інтуїтивно зрозумілих нотацій для бізнес-правил. Діаграми в стилі блок-схем в рамках BPMN можуть дати детальні специфікації бізнес-процесів від початку до кінця. Однак, BPMN не може декомпонувати систему для великих інформаційних систем.

DFD – діаграма потоків даних, графічно відображає потоки даних у системі. Ця діаграма дозволяє зобразити процеси обробки даних, що відбуваються у інформаційній системі. Розробнику зазвичай зручно спочатку створювати діаграму потоків даних рівня контексту, яка демонструє взаємодію системи з зовнішніми модулями. Подальша робота над діаграмою полягає у деталізації процесів та потоків даних для представлення всієї системи в розвинутому вигляді. Основна концепція DFD полягає в підході "зверху вниз" до розуміння системи. Цей підхід узгоджується з концепцією системного підходу, яка розглядає систему в цілому та стосується розуміння системи через вивчення її компонентів та їх взаємодії. Під час опису бізнес-процесу за допомогою DFD визначаються сховища даних, які використовуються в процесі, та потоки даних, що генеруються в процесі.

UML - це уніфікована мова моделювання, стандарт моделювання програмних процесів за допомогою різних наборів діаграм, що допомагає розробникам програмного забезпечення та системним інженерам визначати, зобразити та реалізовувати процеси, в основному в об'єктно-орієнтованій парадигмі розробки. Уніфікована мова моделювання (UML) є загальновизнаним інструментом моделювання у сфері програмної інженерії для розробки різних типів комп'ютеризованих систем. UML включає набір різних типів діаграм із різними елементами моделювання та різноманітними графічними стилями. Різноманітні діаграми в UML можуть надавати детальні специфікації для програмної інженерії з різних перспектив побудови інформаційних систем. Однак вони можуть бути занадто складними для документування бізнес-процесів з точки зору управління цими процесами.

Коли компанії починають документувати свої процеси, вони відчувають потребу у систематичному відстеженні цих процесів, оскільки вони постійно змінюються і вдосконалюються. Важливо мати уявлення про те, які саме процеси є найсвіжішими і актуальними, а також забезпечити легку можливість оновлення цих процесів. Це сприяє ефективному управлінню процесом з метою забезпечення його актуальності та легкості оновлення. Потреба в управлінні документацією процесу була однією з ключових дійових сил у створенні системи управління документацією. Система управління документами забезпечує зберігання та відстеження документів, а також підтримує ряд інших функцій.

- В системі документообігу будуть зберігатися різні версії документів, а кожній версії буде присвоєно мітку часу. Остання версія документа буде легко визначатися і вважатиметься активною версією за замовчуванням.
- Управління документами та робочими процесами означатиме, що при необхідності внесення змін до процесу система буде керувати як доступом до редагування документів, так і маршрутизацією документів для затвердження.
- Також важливою є комунікація. Під час змін у процесі впровадження, важливо, щоб особи, які відповідають за це, були повідомлені про будь-які

зміни. Система управління документами повідомить відповідним особам про затвердження змін у документі.

Звичайно, системи управління документами використовуються не лише для керування бізнес-процесами, але й для управління різними типами документів, такими як юридичні документи або проектна документація. Організації, які намагаються вдосконалювати бізнес-процеси, також зазвичай створюють структури для управління бізнес-процесами.

Рис. 1.2 – Класифікація інформаційних систем за структурованістю розв’язування завдань [5]

Управління бізнес-процесами (BPM) – це цілеспрямоване планування, документування, впровадження та розподілення бізнес-процесів організації за допомогою інформаційних систем та технологій. Управління бізнес-процесами - це більше, ніж просто автоматизація деяких простих кроків. Хоча автоматизація може зробити бізнес більш ефективним, вона не може бути використана для забезпечення конкурентної переваги. З іншого боку, управління бізнес-процесами може бути невід'ємною частиною створення такої переваги. Не всіма процесами в організації слід керувати таким чином. Організація повинна шукати процеси, які є важливими для функціонування бізнесу, а також ті, які можуть бути використані

для отримання конкурентних переваг. Найкраще звернути увагу на ті процеси, в яких задіяні працівники з різних відділів, ті, що вимагають прийняття рішень, які нелегко автоматизувати, а також процеси, які змінюються залежно від обставин.

Правильне управління бізнес-процесами може принести організації кілька важливих переваг, які можуть бути використані для формування конкурентних переваг. Ймовірні переваги включають:

- Розширення можливостей співробітників стає можливим, коли бізнес-процес правильно спроектований і підтримується інформаційними технологіями. У такому випадку працівники мають можливість самостійно впроваджувати процеси згідно своїх потреб. Наприклад, в контексті політики повернення товарів працівник може приймати рішення про повернення, які здійснюються протягом перших чотирнадцяти днів, або використовувати систему для визначення умов повернення після цього періоду.
- Інтегрована звітність дозволяє організації відстежувати ключові показники щодо їхніх процесів, вбудовуючи вимірювання безпосередньо в програмне забезпечення.
- Впровадження найкращих практик означає використання процесів, які підтримуються інформаційними системами, для реалізації найефективніших підходів до класу бізнес-процесів. У цьому контексті організація може встановити вимогу, згідно з якою всі клієнти, що повертають товар без чеку, повинні пред'являти документ, що підтверджує їх особу. Ця вимога може бути впроваджена в систему таким чином, що повернення не буде оброблене, якщо не буде введений дійсний ідентифікаційний номер.
- Забезпечення послідовності означає створення процесу та його виконання за допомогою інформаційних технологій для забезпечення єдності та однаковості в усій організації. Наприклад, усі магазини роздрібної мережі можуть застосовувати однакову політику повернення товарів. Якщо ця політика зазнає змін, вони автоматично впроваджуються у всіх магазинах мережі.

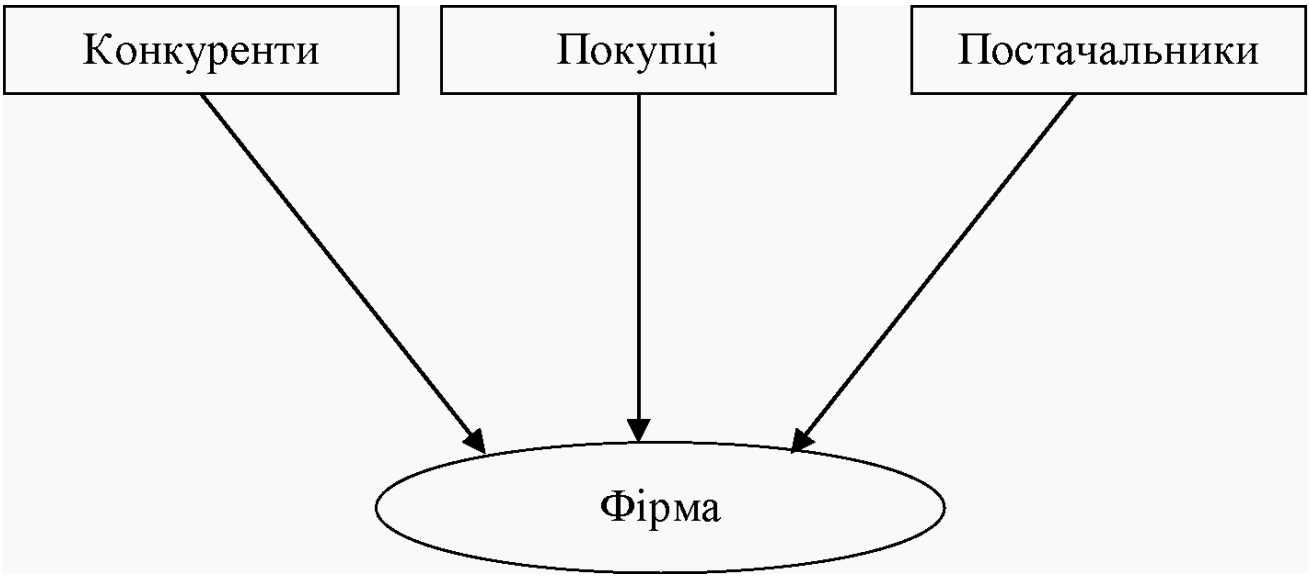


Рис. 1.3 – Фактори що впливають на підприємство [5]

Прагнення організацій керувати своїми процесами для здобуття конкурентної переваги вимагає розуміння того, що існуючі методи ведення справ можуть бути не найбільш ефективними або продуктивними. Навіть якщо процес був розроблений вже досить давно, його просте оновлення технологіями не робить його автоматично оптимальним.

Багато зі структур та процесів у наших робочих місцях були сформовані у минулому, коли конкурентне середовище було інше, а комп'ютери не використовувалися. Вони спрямовані на досягнення більшої ефективності та контролю. Однак сучасні запити на інновації, сервіс та якість вимагають нових підходів.

Реінжиніринг бізнес-процесів (BPR) – це не тільки про автоматизацію існуючого процесу, а і про повне розуміння мети процесу, ґрунтовне перетворення його з нуля для досягнення значного підвищення продуктивності та якості. Більшість людей зазвичай зосереджуються на невеликих, локальних поліпшеннях в процесі. Адже повний редизайн вимагає багато часу.

На жаль, в багатьох організаціях реінжиніринг бізнес-процесів має погану репутацію. Це сталося через те, що його часто використовували як засіб для зменшення витрат, що насправді не мало відношення до самого реінжинірингу бізнес-процесів. Наприклад, багато компаній використовували його як привід для

звільнення частини свого персоналу. Однак сьогодні багато принципів реінжинірингу бізнес-процесів інтегровані в бізнес та розглядаються як необхідна частина ефективного управління бізнес-процесами.

1.1.3 Поняття даних в інформаційних системах

Дані – це факти, які можуть бути позбавлені контексту або конкретного наміру. Наприклад, список замовлень на продаж комп'ютерів - це приклад даних. Дані можуть бути як кількісними, так і якісними. Кількісні дані включають числа або результати вимірювань, підрахунків чи інших математичних операцій, тоді як якісні дані можуть описувати, наприклад, колір або інші атрибути.

Інформація - це результат обробки даних, який має визначений контекст, важливість та призначення. Наприклад, якщо щоденні дані про продажі за минулий рік обробляються для розрахунку місячних продажів, то отримані дані вже можна вважати інформацією. Інформація зазвичай включає в себе аналіз та обробку неструктурованих даних з метою виявлення певних відмінностей, тенденцій або закономірностей для досягнення певної цілі.

Знання – це людські переконання або уявлення про взаємозв'язки між фактами чи поняттями, що стосуються певної сфери. Наприклад, розуміння зв'язку між якістю товару та його продажами - це знання. Знання можна вважати інформацією, яка сприяє прийняттю дій. Після того, як ми розмістили наші дані в контексті, агрегували та проаналізували їх, ми можемо використовувати їх для прийняття рішень у нашій організації. Ці знання можуть бути застосовані для прийняття рішень, розробки політики та навіть створення інновацій.

Явні знання - це знання, яке легко пояснити і яке доступне для інших. Такі знання легко передаються іншим людям. Більшість форм явних знань можуть зберігатися на різних типах носіїв інформації. Для їх отримання не потрібні постійні тренування, як у випадку з неявними знаннями. Людина може самостійно навчитися таким знанням, використовуючи зрозумілі та чітко сформульовані правила. Явні знання зазвичай визначаються як ті, що можуть бути виражені словами або числами. На відміну від них, неявні знання включають інтуїцію, які важко передати іншій людині за допомогою простих засобів комунікації.

Очевидно, що коли інформація або явні знання фіксуються і зберігаються в комп'ютері, вони перетворюються на дані, якщо позбавлені контексту або наміру.

Великі дані - це об'ємні масиви даних, які настільки великі, що звичайні методи обробки даних не можуть їх ефективно обробити, зазвичай засновані на рішеннях класу бізнес-аналітики та системах управління базами даних, не можуть бути застосовані до них. Зберігання та обробка такого обсягу даних виходить за рамки можливостей традиційних інструментів управління даними. Розробка ефективних інструментів і методів для управління та аналізу цих об'ємних масивів даних є складним завданням. Багато інформаційних систем спрямовані на те, щоб перетворювати дані в інформацію, яка може бути використана для прийняття рішень. Для досягнення цієї мети система повинна бути здатна приймати дані, допомагати користувачеві встановлювати контекст для цих даних та надавати інструменти для їх агрегації та аналізу. База даних призначена саме для цієї мети.

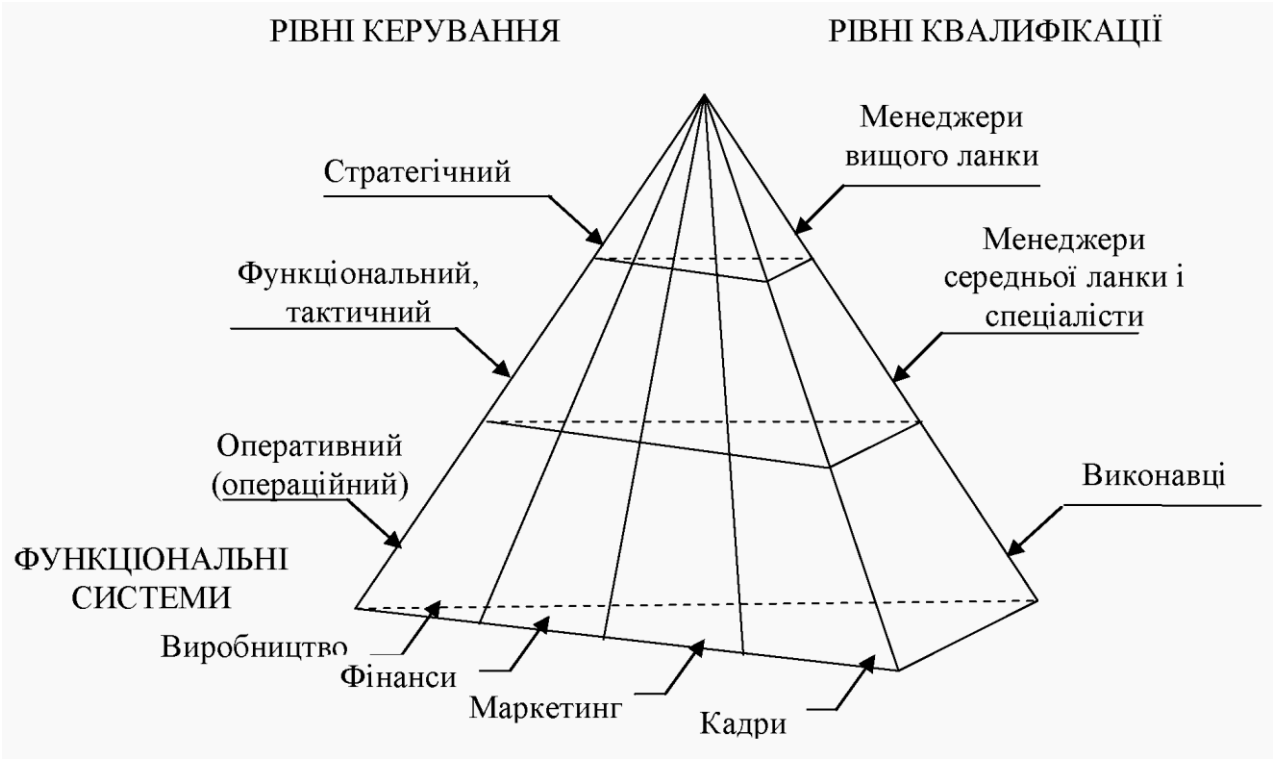


Рис. 1.4 – Різні типи інформаційних систем залежно від функціональної ознаки та рівнем кваліфікації персоналу [5]

Дані в організації є важливим ресурсом, але використання таких інструментів, як електронні таблиці Excel або документи Word, для зберігання та

обробки бізнес-даних не є ефективним рішенням. Це може призвести до недоліків у даних, таких як надлишковість, неузгодженість, неточність і пошкодження. У невеликих обсягах це може бути допустимо, але для великих організацій ці помилки можуть мати серйозні наслідки. Часті помилки в управлінні даними пояснюються такими факторами:

- Відсутність контролю надлишкових даних. Люди часто зберігають більше даних, ніж необхідно, для зручності. Надмірні дані можуть призвести до непослідовності у наборі даних. Наприклад якщо одні і ті самі дані зберігаються декілька разів. Хоча це може полегшувати доступ до інформації, такий підхід може мати негативні наслідки. У малих системах це може здатися допустимим, але у великій системи можуть виникнути проблеми зі зміною всіх надлишкових даних, що може призвести до пошкодження всього набору даних.
- Порушення цілісності даних. Цілісність даних вказує на консистентність між збереженими даними. Якщо ми встановимо правило цілісності даних, наприклад, тоді порушення цілісності даних буде неможливим.
- Розрахунок на людську пам’ять для зберігання та пошуку необхідних даних. Поширена помилка в управлінні ресурсами даних полягає у надмірному використанні людської пам'яті для пошуку даних. Людина може запам'ятати, де зберігаються певні дані, але також може помилятися. Якщо певна частина даних знаходиться в місці, яке людина не може згадати, то ці дані фактично стають втраченими. Покладання на людську пам'ять для зберігання та пошуку даних з часом призводить до дезорганізації всього набору даних.

База даних - це організована колекція пов'язаних даних. Це організована колекція, тому що в базі даних всі дані описані і пов'язані з іншими даними. Щоб уникнути вищезгаданих типових недоліків в управлінні ресурсами даних, необхідно застосовувати технологію баз даних. У цьому контексті ми розглядатимемо лише комп'ютеризовані бази даних. Хоча електронні таблиці не можуть замінити бази даних, вони можуть бути ідеальним інструментом для аналізу даних, що зберігаються в базі даних. Пакет електронних таблиць можна

підключити до певної таблиці або запиту в базі даних і використовувати для створення діаграм або аналізу цих даних.

Бази даних можуть бути структуровані різними способами за допомогою різних моделей. Модель даних бази даних визначає логічну структуру елементів даних та їх взаємозв'язків. Існує кілька моделей даних. В наш час реляційні системи баз даних широко використовуються в бізнес-організаціях, за рідкісними винятками. Реляційну модель даних легко зрозуміти і використовувати. У реляційній базі даних дані організовані в таблиці. Кожна таблиця має набір полів, які визначають структуру даних, що зберігаються в таблиці. Запис - це один екземпляр набору полів у таблиці. Рядки можна уявити рядками таблиці, а поля - як стовпці таблиці. Спеціальне поле або комбінація полів, які визначають унікальний запис, називається ключем. Зазвичай ключ - це унікальний ідентифікаційний номер запису.

З появою великих даних та численних нових інструментів і методик компанії навчаються ефективно використовувати інформацію для своєї користі. Термін "бізнес-аналітика" описує процес, у якому організації аналізують зібрані дані, щоб отримати конкурентну перевагу. Окрім використання власних даних, що зберігаються у сховищах даних, компанії часто купують інформацію у брокерів даних, щоб отримати загальне уявлення про свою галузь та економічні умови. Результати цього аналізу можуть впливати на організаційні стратегії та забезпечувати конкурентні переваги.

Візуалізація даних - це спосіб графічного відображення інформації та даних, що дозволяє швидко узагальнити дані у більш зрозумілій формі. Це можуть бути діаграми, графіки або карти, які допомагають спростити інформацію та сприяють виникненню нових ідей та розумінню. Часто візуалізація даних стає першим кроком у вивченні та аналізі даних організації. Прикладами програм для візуалізації даних є Tableau та Google Data Studio.

Оскільки організації все більше використовують бази даних як основний інструмент своєї роботи, вони все частіше розуміють важливість повного використання та розуміння зібраних даних. Проте безпосередній аналіз даних,

необхідних для щоденних операцій, може перевантажити роботу компанії. Крім того, організації хочуть мати можливість аналізувати дані в історичному контексті, порівнюючи їх з минулими періодами. Ці потреби вибудовали концепцію сховища даних. Суть концепції сховища даних полягає в тому, щоб витягти дані з одного або кількох джерел баз даних організації та зберігати їх для подальшого аналізу у спеціальному сховищі даних, що, в суті, є ще однією базою даних. Однак реалізація цієї концепції не така проста, оскільки сховище даних повинне відповідати наступним критеріям:

- Використання неоперативних даних. Це означає, що сховище даних отримує копію інформації з активних баз даних, які використовуються в повсякденній діяльності компанії. Отже, сховище даних повинно регулярно та за розкладом оновлювати дані з існуючих баз даних.
- Використання нестаціонарної системи. Це означає, що кожен раз, коли дані переносяться до сховища даних, вони отримують позначку часу, що дає змогу порівнювати їх між різними моментами часу.
- Дані в сховищі даних підлягають стандартизації. Оскільки вони часто надходять з різних джерел, може виникнути ситуація, коли вони мають різні визначення або одиниці виміру. Наприклад, формати дат можуть відрізнятися між різними базами даних. Щоб забезпечити узгодженість, необхідно встановити стандартний формат для дат і перетворити всі дані у цей формат. Цей процес відомий як етап видобування-трансформації-завантаження (ETL).

Інтелектуальний аналіз даних - це процес виявлення раніше невідомих тенденцій, закономірностей та асоціацій у наборах даних з метою підтримки прийняття рішень. Зазвичай цей аналіз виконується за допомогою автоматизованих засобів на основі великих обсягів даних, таких як сховища даних.

Один з методів інтелектуального аналізу даних, який організація може використовувати для проведення такого аналізу, називається машинним навчанням. Машинне навчання використовується для аналізу даних і побудови

моделей, не будучи явно запрограмованим на це. Існує дві основні гілки машинного навчання: контрольоване навчання та неконтрольоване навчання.

1.2 Загальні положення машинного навчання

Машинне навчання – галузь знань, яка дає комп'ютерам можливість навчатися без очевидного втручання і програмування. У машинному навчанні дані поділяються на навчальні та тестові.

Машинне навчання охоплює багато алгоритмів, що ґрунтуються на статистичних методах, і вибір відповідного алгоритму чи їх комбінації є постійним завданням для фахівців у цій галузі. Існує три основні категорії машинного навчання. Ці категорії включають контрольоване навчання, неконтрольоване навчання та підкріплення.

Контрольоване навчання зосереджується на вивченні закономірностей шляхом встановлення зв'язку між змінними та відомими результатами і роботи з маркованими наборами даних. Контрольоване навчання працює шляхом подачі машині вибірки даних з різними ознаками (позначеними як "X") і правильним виведенням даних (позначеним як "y"). Той факт, що вихідні значення і значення ознак відомі, кваліфікує набір даних як "маркований". Потім алгоритм розшифровує закономірності, що існують у даних, і створює модель, яка може відтворювати ті ж самі основні правила з новими даними

Контрольоване навчання застосовується, коли організація володіє даними про свою минулу діяльність та бажає використати їх для досягнення певних результатів. Наприклад, плануючи запуск нової маркетингової кампанії для певної лінійки продуктів, компанія може аналізувати дані з попередніх маркетингових ініціатив, щоб визначити, які групи споживачів найбільш позитивно реагували на них. Після аналізу може бути створена модель машинного навчання, яка допоможе ідентифікувати ці нові клієнти. Цей підхід відомий як контрольоване навчання, оскільки аналіз спрямований на досягнення певних результатів. Методи контрольованого навчання включають дерева рішень, нейронні мережі, класифікатори та логістичну регресію. Приклади алгоритмів

керованого навчання включають k-найближчих сусідів, регресійний аналіз, дерева рішень, нейронні мережі та машини опорних векторів.

У неконтрольованому навчанні не всі змінні або шаблони даних класифікуються наперед. Натомість машина повинна виявляти приховані шаблони та створювати мітки за допомогою відповідних алгоритмів. Один з популярних прикладів алгоритму неконтрольованого навчання - це алгоритм кластеризації k-середніх.

У випадку неконтрольованого навчання не всі вхідні змінні та шаблони даних класифікуються. Замість цього, система має виявити та розпізнати приховані шаблони, використовуючи алгоритми неконтрольованого навчання. Один з популярних прикладів такого навчання – алгоритм кластеризації k-середніх.

Навчання з підкріпленням представляє третю та найбільш складну категорію алгоритмів у машинному навчанні. У відміну від контрольованого та неконтрольованого навчання, навчання з підкріпленням постійно вдосконалює свою модель, використовуючи зворотний зв'язок з попередніх ітерацій. Це відрізняється від контрольованого та неконтрольованого навчання, де модель досягає певної кінцевої точки після формулювання на основі навчальних та тестових сегментів даних. Навчання з підкріпленням подібне до навчання, де алгоритми встановлюються для навчання моделі шляхом постійного покращення. Типова модель навчання з підкріпленням має вимірювані критерії ефективності, де результати не просто фіксуються, а оцінюються.

Приклад навчання з підкріпленням – це Q-навчання. Машина вивчатиме, як вибирати дії для конкретного стану, такі, які генерують або підтримують найвищий рівень Q. Спочатку вона пройде через фазу випадкових дій у різних станах. Машина буде записувати свої результати, включаючи нагороди та покарання, і спостерігатиме, як вони впливають на рівень Q. Ці дані

зберігатимуться для подальшого використання у вирішенні та оптимізації майбутніх дій.

Набори даних майже завжди потребують спеціальної попередньої обробки та втручання людини перед тим, як вони будуть готові для подальшого використання. Існує великий різноманітний спектр методів. Скрабінг (Scrubbing) – це технічний процес вдосконалення набору даних, щоб забезпечити оптимальність для подальшої обробки. Це може включати корекцію, іноді видалення неповних, неправильно відформатованих, неактуальних або дубльованих даних. Крім того, це може передбачати перетворення текстових даних у числові значення та аналіз характеристик.

Щоб отримати найкращі результати даних, важливо спочатку визначити змінні, які найбільше відповідають гіпотезі. На практиці це означає, що варто бути вибірковими щодо змінних, які обираних для побудови моделі. Це означає, що потрібно проявити обережність до вибору змінних, які включаються до моделі.

1.3 Загальні положення нейронних мереж

Нейронні мережі, також відомі як штучні нейронні мережі (ANN), є підмножиною алгоритмів машинного навчання, які імітують роботу людського мозку. Ці алгоритми потужні у розпізнаванні основних взаємозв'язків у наборі даних. Сьогодні нейронні мережі широко використовуються для класифікації (бінарної класифікації, багатокласової класифікації, а також регресії) та кластеризації даних.

Нейронна мережа - це мережа з декількох нейронів, складених разом у кілька шарів. Мережа складається з одного вхідного шару, одного або декількох вихідних шарів і декількох прихованих шарів. Якщо кількість прихованих шарів перевищує 3, що зазвичай трапляється в сучасних програмах для вирішення конкретних завдань, такі мережі називаються глибокими нейронними мережами (Deep Neural Networks). У цьому типі нейронної мережі вихід кожного шару є входом для

наступних шарів. Навчання нейронної мережі означає безперервне коригування ваг та зміщень вхідних ознак, зазвичай за допомогою зворотного поширення.

Нейрон - будівельний блок штучної нейронної мережі - це математична функція, яка приймає набір вхідних даних і множить їх на відповідні ваги. Потім ці добутки вагових коефіцієнтів підсумовуються, і кінцеве значення пропускається через деяку нелінійну функцію активації. Метою використання нелінійної функції активації є апроксимація комплексної функції.

Приклад чотиришарова нейронної мережі на Рисунку 1. Крайній лівий шар цієї мережі називається вхідним, а нейрони в ньому - вхідними нейронами. Крайній правий шар, або вихідний шар, містить вихідні нейрони або, як у цьому випадку, один вихідний нейрон. Середній шар називається прихованим, оскільки нейрони в ньому не є ні вхідними, ні вихідними. Термін "прихований" може здатися трохи загадковим - коли я вперше почув цей термін, то подумав, що він має якесь глибоке філософське або математичне значення - але насправді він просто означає, що нейрони не є ані входом, ані виходом. Наведена вище мережа має лише один прихований шар, але деякі мережі мають кілька прихованих шарів. Наприклад, наступна чотиришарова мережа має два прихованих шари.

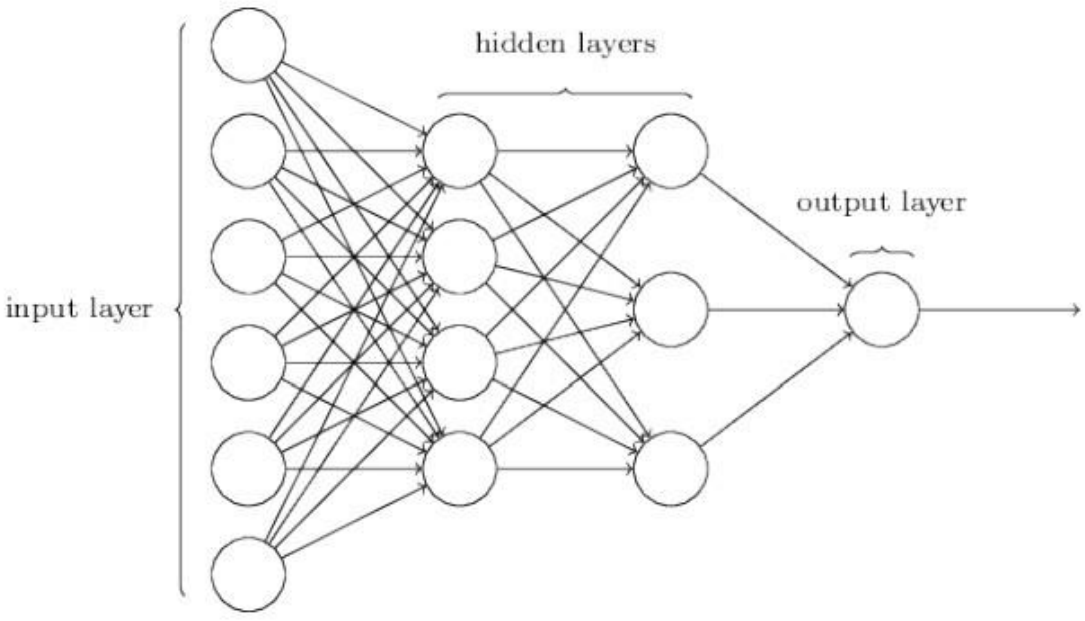


Рис. 1.5 – Чотиришарова нейронна мережа [10]

Для декількох дійсних вхідних даних (навчальних прикладів), вхідні дані можуть бути розміщені у двовимірній матриці, де кожен стовпчик представляє один вхід з n ознаками, а всього є m стовпчиків для всіх навчальних прикладів. Наприклад

Висновок до розділу 1

У розділі було значено, що інформаційні системи є сукупністю компонентів, що взаємодіють для збору, обробки, зберігання та поширення інформації. Важливою складовою є також людський фактор та процеси, що впливають на ефективність використання інформаційних систем. Було розглянуто різницю між даними, інформацією та знаннями. Також було розглянуто важливість розуміння явних та неявних знань для успішного використання інформаційних систем. Крім того, було відзначено роль великих даних у сучасному світі та їх вплив на управління та аналіз інформації. У розділі про моделювання процесів було досліджено концепцію процесів як послідовності дій, спрямованих на досягнення певної цілі.

Були розглянуті різні інструменти для моделювання бізнес-процесів, такі як BPMN, DFD та UML, та їх застосування в практичних сценаріях. Моделювання бізнес-процесів допомагає підприємствам підвищити ефективність, оптимізувати витрати та досягати конкурентних переваг.

Також зазначено, що одним з аспектів розвитку інформаційних систем у сучасному світі є машинне навчання. Ця технологія дозволяє системам автоматизувати процеси аналізу та вивчення даних, що дає змогу підприємствам отримати цінну інформації з великих обсягів інформації. Використання машинного навчання в інформаційних системах дозволяє підприємствам приймати кращі стратегічні рішення, прогнозувати тенденції та вдосконалювати свої процеси.

РОЗДІЛ 2. ПОРІВНЯННЯ ДОСТУПНИХ ВИРІШЕНЬ ЗАДАЧІ АНАЛІЗУ ІНФОРМАЦІЙНИХ СИСТЕМ

2.1 Розгляд існуючих алгоритмів машинного навчання

LSTM є важливим типом рекурентної нейронної мережі, яка може навчатися і зберігати інформацію про довгострокові залежності. Ця можливість запам'ятовувати попередні дані протягом тривалих періодів корисна для аналізу часових рядів та інших завдань. Ланцюжкова структура LSTM містить чотири взаємодіючі шари, які працюють разом для ефективного аналізу даних. Крім аналізу часових рядів, LSTM також успішно використовуються для різноманітних завдань, таких як розпізнавання мови та музичного складання.

Генеративні адверсаріальні мережі (GAN) є іншим типом глибокого навчання, які використовуються для створення нових прикладів даних, подібних до навчальних даних. GAN складаються з двох основних компонентів: генератора і дискримінатора, які співпрацюють для створення реалістичних прикладів даних. Застосування генеративних алгоритмів глибокого навчання стало широко поширеним з часом і зараз вони використовуються для різноманітних завдань, включаючи покращення зображень та створення анімацій та 3D-моделей.

Згорткові нейронні мережі (CNN) використовуються для класифікації та завдань комп'ютерного зору. До появи згорткових нейронних мереж застосовувалися ручні, трудомісткі методи виділення ознак для ідентифікації об'єктів на зображеннях. Проте зараз згорткові нейронні мережі забезпечують більш масштабований підхід до класифікації зображень і розпізнавання об'єктів, використовуючи принципи лінійної алгебри, зокрема множення матриць, для виявлення шаблонів на зображенні. Однак, вони можуть бути вимогливими до обчислювальних ресурсів, тому для тренування моделей потрібні графічні процесори (GPU).

CNN, також відомі як ConvNets, або згорткові нейронні мережі складаються з кількох шарів і головним чином використовуються для обробки зображень та розпізнавання об'єктів. Програми CNN широко використовуються для

ідентифікації супутникових знімків, обробки медичних зображень, прогнозування часових рядів і виявлення аномалій.

Виявлення об'єктів існує вже досить давно і може здійснюватися як за допомогою традиційного підходу класифікації з використанням машин опорних векторів (SVM), так і за допомогою сучасної нейромережевої архітектури, що дозволяє автоматично виявляти і вивчати об'єкти на заданому зображенні. Дуже важливо, щоб вхідне зображення не містило шуму і мало чітко окреслені краї для виявлення контурів. Етапи попередньої обробки зображень в обговорюють алгоритми вирівнювання гістограми, виявлення країв, проріджування країв та перетворення. Ці алгоритми для попередньої обробки зображень можуть бути використані з бібліотек з відкритим вихідним кодом, таких як OpenCV, ImageJ. Алгоритм в забезпечує ефективний спосіб виявлення контурів на зображенні шляхом знаходження меншої кількості точок (тобто пікселів), які представляють вхідну криву та апроксимації різного типу полігонів, що представляють інтерес для користувачів, які можуть бути перевірені за допомогою перехресної кореляції або коефіцієнту кореляції. Сучасний спосіб виявлення об'єктів, однак, використовує згорткові нейронні мережі з пропозицією області (Region Proposal Convolutional Neural Networks, RCNN) або швидші RCNN для апроксимації областей, таких як прямокутники, овали або будь-які інші типи полігонів.

Метою операції згортки є виділення високорівневих характеристик, таких як краї, з вхідного зображення. ConvNets не повинні обмежуватися лише одним шаром згортки. Зазвичай, перший ConvLayer відповідає за захоплення низькорівневих характеристик, таких як краї, колір, орієнтація градієнта тощо. З додаванням шарів архітектура адаптується і до високорівневих характеристик, даючи нам мережу, яка має цілісне розуміння зображень у наборі даних, подібно до того, як це робили б ми самі.

Існує два типи результатів операції - один, в якому згорнута функція зменшується в розмірності порівняно з вхідними даними, а інший, в якому розмірність або збільшується, або залишається незмінною. Це досягається

застосуванням параметра Valid Padding у першому випадку, або Same Padding у другому.

Подібно до шару згортки, шар об'єднання відповідає за зменшення просторового розміру згорнутої функції. Це робиться для зменшення обчислювальної потужності, необхідної для обробки даних за рахунок зменшення розмірності. Крім того, він корисний для вилучення домінантних ознак, які є інваріантними щодо обертання та положення, таким чином підтримуючи процес ефективного навчання моделі.

Комп'ютерний зір - це галузь штучного інтелекту (AI), яка допомагає отримувати високорівневе розуміння та змістовну інформацію з зображення та відео. У той час як обробка зображень в основному пов'язана з виконанням перетворень у цифровому зображенні для покращення або спрощення даного зображення (попередня обробка), комп'ютерний зір в першу чергу займається розумінням зображень, що включає широкий спектр застосувань, таких як виявлення об'єктів, 3D-моделювання, спостереження, оптичне розпізнавання символів, взаємодія людини з комп'ютером і так далі. Одне з ключових понять, яке відіграє важливу роль як в обробці зображень, так і в комп'ютерному зорі - це згортка (Convolution).

З точки зору математики, згортка - це оператор, який приймає дві функції f і g і виводить третю функцію (f * g), яка виражає, як форма однієї з них змінюється іншою. Термін згортка відноситься як до функції результату, так і до процесу її обчислення . Згортка для двох функцій f(a) і g(b) в точці c задається рівнянням,
З точки зору комп'ютерного зору, згортка просто означає ковзання невеликої матриці по зображенню для досягнення різних ефектів на зображенні, таких як підвищення різкості, розмиття, виявлення країв.

2.2 Вибір об’єкту аналізу в інформаційних системах

UML використовується для ефективного моделювання великих і складних систем і базується на найкращих інженерних практиках. UML визначає набір діаграм, які можна широко класифікувати на структурні та поведінкові діаграми. Всі ці діаграми складаються з компонентів (тобто класифікаторів), які

представляють концептуальні або фізичні аспекти моделі. Деякі приклади класифікаторів включають класи, інтерфейси, типи асоціацій тощо. Класифікатори представлені суцільними прямокутниками, що містять назву класифікатора, і, за бажанням, з відділеннями, розділеними горизонтальними лініями, що містять ознаки або інші члени класифікаторів.

Діаграма класів - це статична структурна діаграма, яка використовується для концептуального моделювання програми для опису класів та їх взаємозв'язків. Діаграма класів сприяє абстрагуванню - одному з багатьох принципів об'єктно-орієнтованого моделювання та проектування - дозволяючи нам проектувати систему на високому рівні, приховуючи деталі реалізації.

Клас в UML - це класифікатор, який описує набір об'єктів зі спільними ознаками (атрибутами та методами), обмеженнями та семантикою. Приклад класу Project показаний нижче на рисунку 6. Якщо клас представлено у вигляді 3 різних відсіків всередині прямокутника, то у верхньому відсіку міститься ім'я класу - вирівняний по центру текст, виділений жирним шрифтом з великої літери, середній відсік містить атрибути класу з відповідними типами, а нижній відсік містить методи класу з типом вхідних параметрів та відповідними типами повернення цих методів. Кожному атрибуту та методу класу може передувати оператор видимості, який може бути одним з +, - або #, що означає загальнодоступну, приватну та захищену видимість відповідно.

Рис. 2.1 – Приклад UML класу Адміністратора системи

Зв'язок в UML - це абстрактний елемент, який призначений для відображення залежності між елементами UML. Між класами можуть існувати різні типи зв'язків. Крім того, зв'язки також встановлюють асоційовану множинність між класами, які пов'язані один з одним.

Висновок до розділу 2

Використання UML для моделювання складних систем та застосування загорткових нейронних мереж для вирішення завдань комп'ютерного зору ймовірно будуть найкращими інструментами для вирішення поставленої задачі.

UML є потужним інструментом для моделювання великих та складних систем, що базується на найкращих інженерних практиках. Використовуючи структурні та поведінкові діаграми, UML дозволяє розробникам створювати концептуальні та фізичні моделі систем, що забезпечують абстрагування і приховування деталей реалізації. Зокрема, діаграми класів є важливим засобом для опису класів та їх взаємозв'язків у системі, сприяючи розробці системи на високому рівні.

Зважаючи на те що діаграми класів в UML можуть представляти складні системи, то це наштовхує на наступні думки. Розробка алгоритмів машинного

навчання спрямована на аналіз складних UML діаграм може підвищити ефективність обробки великих об’ємів інформації. Оскільки діаграми класів можуть представляти складні системи з купою зав’язків і класів. Саме тому обрати для аналізу діаграм класів в сфері інформаційних систем буде оптимальним рішенням.

У той же час, згорткові нейронні мережі (CNN) зарекомендували себе як ефективний інструмент для класифікації зображень та розпізнавання об'єктів. Використовуючи принципи лінійної алгебри, такі як множення матриць, CNN здатні автоматично виділяти ознаки та виявляти шаблони на зображеннях. Це значно спрощує процес ідентифікації об'єктів у порівнянні з ручними методами виділення ознак, що були поширені до появи CNN. Проте слід враховувати, що навчання CNN потребує значних обчислювальних ресурсів, зокрема графічних процесорів (GPU).

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ВИКОРИСТАННЯ МАШИННОГО НАВЧАННЯ В ІНФОРМАЦІЙНИХ СИСТЕМАХ, А САМЕ В ДІАГРАМАХ КЛАСІВ

3.1 Підготовка даних

Навчання нейронної мережі для розпізнавання користувацьких об'єктів (у нашому випадку - класифікаторів UML) вимагає значних ресурсів у вигляді дискового простору, пам'яті, CPU, GPU та TPU. Також важливо мати можливість інтерактивної перевірки результатів роботи моделі та коду. Для розробки свери розробки використовується на платформі Visual Studio Code (VSC). Python (v. 3.11.1) використовується як мова програмування через його виняткові обчислювальні можливості, оптимізацію коду та широку спільноту з відкритим вихідним кодом, а також наявність багатьох бібліотек та фреймворків.

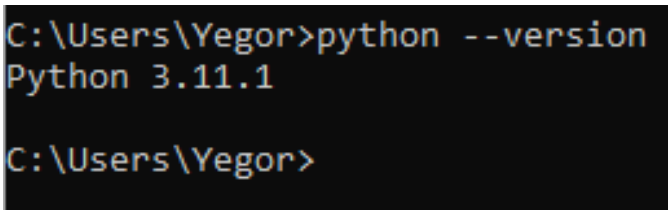


Рис. 3.1 – Вводимо в консоль команду та перевіряємо чи встановлений в нас Python і яка версія встановленна

Був використаний фреймворк detecto для навчання власної моделі Faster-RCNN, для виявлення класів та зв'язків UML. Фреймворк easyocr використовується для розпізнавання та виявлення тексту. Обидва фреймворки написані на Pytorch - фреймворк машинного навчання з відкритим вихідним кодом, заснований на бібліотеці Torch, розроблений компанією Meta-, і сприяє швидкій розробці. Ми використовували бібліотеку Opencv для виконання деяких перетворень зображень та matplotlib для візуалізації результатів.

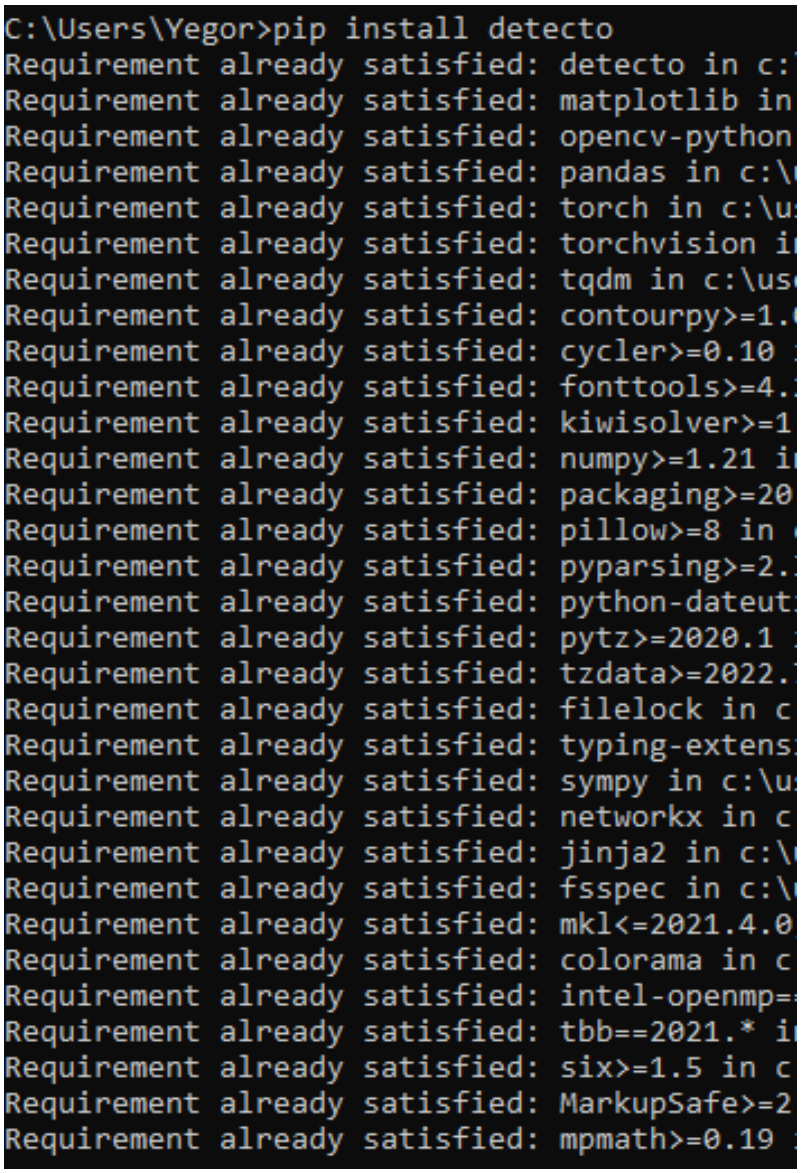


Рис. 3.2 – Вводимо в консоль команду для встановлення фреймворку detecto

Навчальний і тестовий набори даних (зображення) розміщені на Google Drive. Обидва набори даних анотовані відповідними мітками класів за допомогою інструменту анотування labelImg.

Навчальний набір даних складається з п’ятдесяти чотирьох вхідних зображень з відповідними анотаціями у форматі PascalVOC.

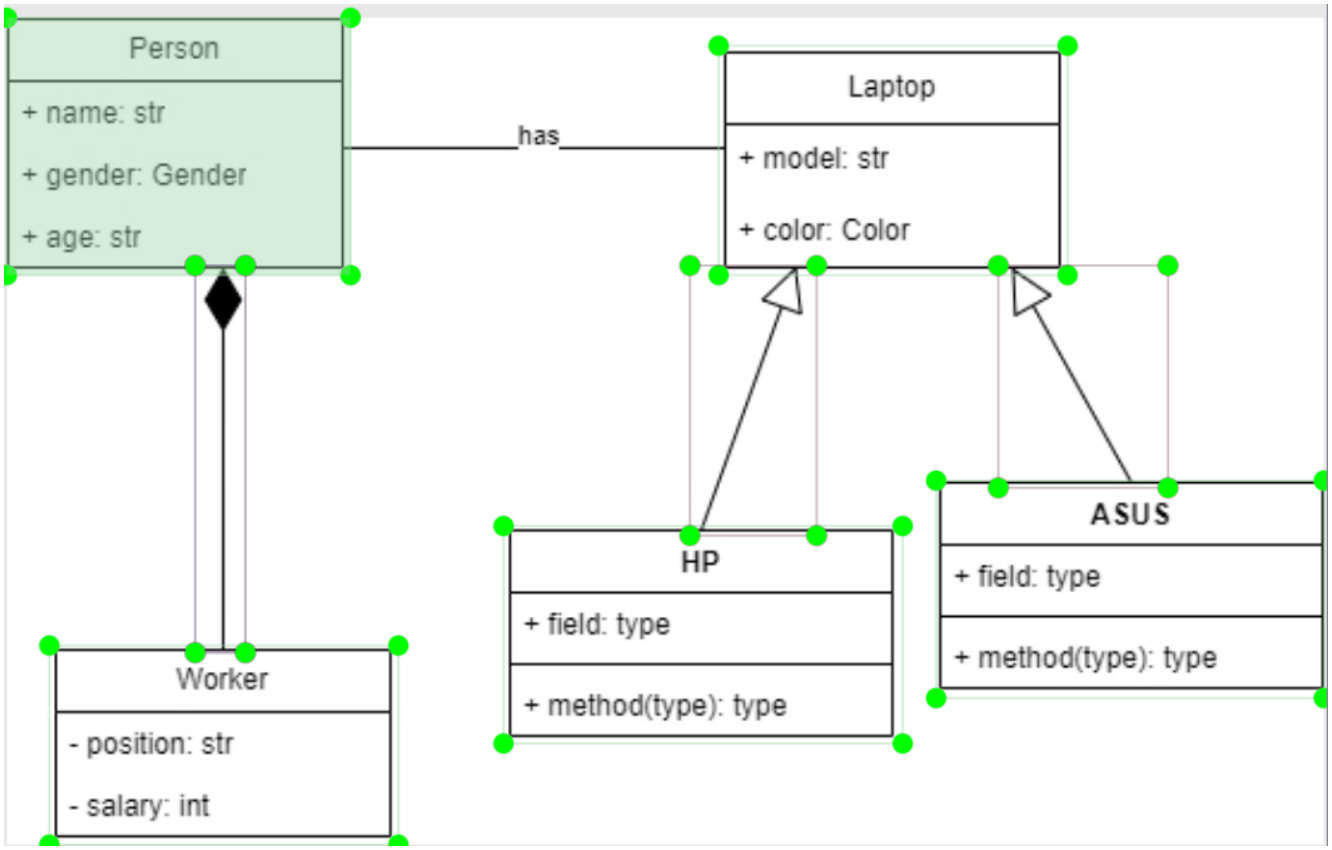


Рис. 3.3 – Зразок набору навчальних даних

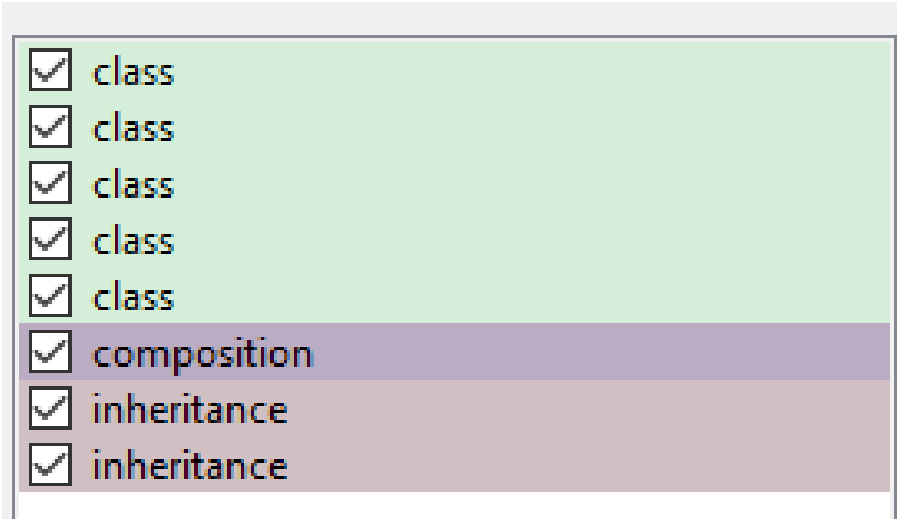


Рис. 3.4 – Анотації до навчальних данних

3.1 Розробка системи

Тестовий набір даних складається з двох схожих по структурі діаграм класів UML.

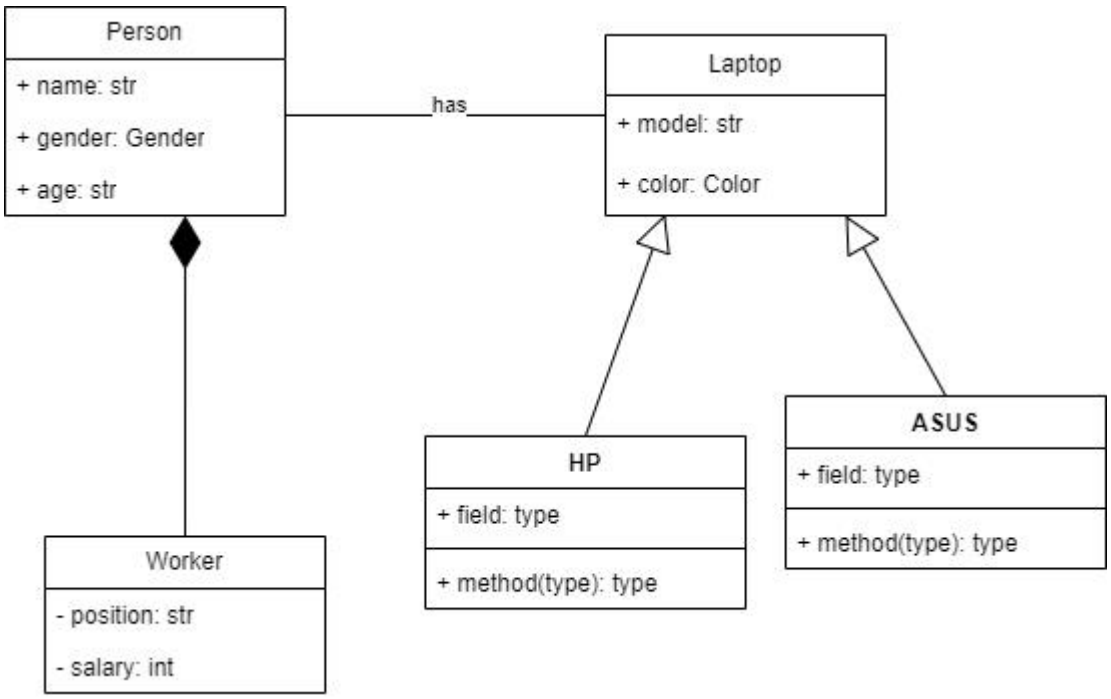


Рис. 3.5 – приклад зображення діаграми класів для тестування

Модель виявлення об'єктів навчалась за допомогою фреймворку detecto, який використовує Faster RCNN для навчання розпізнавання об'єктів, з різними мітками класів. Ми не використовували валідаційний набір для порівняння продуктивності між різними моделями, навченими з різними гіперпараметрами. Значення за замовчуванням для гіперпараметрів, які використовувалися для навчання моделі, наведені нижче.

```
epochs  10
learning rate    0.005
momentum         0.9
weight_decay     0.0005
gamma    0.1
lr_step_size     3
```

Рис. 3.6 – Гіперпараметри для навчання моделі

```
from detecto import core, visualize, utils
from detecto.core import Dataset, DataLoader
train_dataset = core.Dataset('/content/drive/My Drive/Train')
test_dataset = Dataset('/content/drive/My Drive/Test')

loader = core.DataLoader(train_dataset, batch_size= 5, shuffle= True)

labels_map = { 'class': 1,
'association': 2,
'aggregation': 3,
'realization': 4,
'dependency': 5,
'u-association': 6,
'inheritance': 7,
'composition': 8,
'agg-depy-agg': 9,
'aggregation-association': 10,
'box': 11,
'oval': 12
}
model = core.Model([*labels_map.keys()])
model.fit(train_dataset)

model.save("uml_weights.pth")
```

Рис. 3.7 – Модель виявлення об’єктів частина №1

Вихідні прогнози навченої моделі були відфільтровані з використанням значення accuracy_score > 0.25. Початкові прогнози, а також відфільтровані за допомогою оцінки точності, показані на Рисунку 3.6 та Рисунку 3.7.

```
import numpy as np
image=utils.read_image('/content/drive/My Drive/Test/test_1.jpg')
print("Image Shape: ", image.shape)
predictions = model.predict(image)
labels, boxes, scores = predictions

print("Initial predictions, w/o supression of overlapping bbox and accuracy filter")
print("Total number of preictions: ", len(boxes), "\n")
visualize.show_labeled_image(image, boxes, labels)
print("\n\n")

acc_threshold = 0.25
confident_idx = [idx for idx, entry in enumerate(scores) if entry >= acc_threshold]
labels = [labels[idx] for idx in confident_idx]
boxes, scores = boxes[confident_idx], scores[confident_idx]
print("Filtered predictions after threshodling, i.e, accuracy > 0.25.")
print("Total number of preictions: ", len(boxes), "\n")
visualize.show_labeled_image(image, boxes, labels)
print("\n")
for category, pred, confidence in zip(labels, boxes, scores):
print(pred, confidence, category)
```

Рис. 3.8 – Модель виявлення об’єктів, частина №2

```
Image Shape:(2368, 2932, 3)
Initial predictions, w/o supression of overlapping bbox and filter
Total number of predictions: 44
```

Рис. 3.9 – Нефільтровані прогнози для діаграми.

```
Filtered predictions after threshodling, i.e, accuracy > 0.25.
Total number of preictions: 10
```

Рис. 3.10 – Відфільтрований прогноз для діаграми з урахуванням оцінки точності

Для усунення перекриттів між областями з однаковими або різними мітками прогнозу ми використали алгоритм не максимального придушення - обчислює площу перекриття між областями, що перекриваються, і об'єднує їх в одну область, якщо кількість перекриттів перевищує задане порогове значення - з порогом перетину над об'єднанням (Intersection over Union, IOU), рівним 0.5. Результати після придушення блоків, що перетинаються, показано на Рисунку 3.10.

```
import torchvision
import torch
from pprint import pprint
from torchmetrics.detection.map import MAP

int_labels = torch.tensor(list(map(labels_map.get, labels)), dtype= torch.int)
keep = torchvision.ops.nms(boxes, scores, iou_threshold= 0.5)
filtered_boxes, filtered_scores = boxes[keep], scores[keep]
filtered_labels = [labels[idx] for idx in keep]
print("Filtered predictions after non-max suppression of overlapping bboxes, IOU > 0.5")
print("Total number of preictions: ", len(filtered_boxes), "\n")
visualize.show_labeled_image(image, filtered_boxes, filtered_labels)
print("\n")
for fb, fl, fs in zip(filtered_boxes, filtered_labels, filtered_scores):
print(fb, fl, fs)
```

Рис. 3.11 – Реалізація фільтрування прогнозів

```
tensor ([272.7581,159.6465, 1138.8734, 937.0354]) class tensor (0.9977)
tensor ([1795.3662, 1257.4847, 2824.4614, 1996.1605]) class tensor (0.9977)
tensor([ 32.4517, 1409.9215, 866.4893, 2108.2705]) class tensor (0.9974)
tensor ([ 896.1554, 1372.4366, 1720.2419, 2032.8591]) class tensor (0.9965)
tensor ([1821.7087, 163.7551, 2496.3604, 776.5922]) class tensor (0.9962)
tensor ([2102.9976, 700.2303, 2320.7363, 1251.2357]) inheritance tensor (0.3909)
tensor ([ 476.1547, 865.1815, 688.2950, 1376.3400]) inheritance tensor (0.3409)
tensor ([ 900.6353, 880.7171, 1155.8413, 1365.6949]) inheritance tensor (0.2913)
```

Рис. 3.12 – Відфільтровані прогнози після не максимального перекривання обмежувальних рамок.

Остаточні прогнози, отримані після порогового значення точності та не максимального придушення обмежувальних рамок, використовуються для обчислення середньої точності (mAP) та середнього відгуку (mAR) для навченої моделі виявлення об'єктів. Середня точність (AP) обчислюється шляхом порівняння значень істинних даних з результатами прогнозування для різних порогових діапазонів IOU (наприклад, mAP_25, mAP_50 і т.д.) для всіх окремих класів. Середня точність для всіх класів потім усереднюється для обчислення mAP. Розрахована метрика для тестування продуктивності нашої моделі виявлення об'єктів виглядає так, як показано на рисунку 3.13.

```
preds = [
dict(
boxes= filtered_boxes,
scores= filtered_scores,
labels= torch.tensor(list(map(lambda x: labels_map.get(x), filtered_labels)), dtype= torch.int)
)
]

imnum = 1
_, ground_truth = test_dataset[imnum]
gt_boxes, gt_labels = ground_truth["boxes"], ground_truth["labels"]
target = [
dict(
boxes= gt_boxes,
labels= torch.tensor(list(map(labels_map.get, gt_labels)), dtype= torch.int)
)
]

metric = MAP(class_metrics= True)
metric.update(preds, target)
pprint(metric.compute())
```

Рис. 3.13 – Реалізація тестування продуктивності

```
{map: tensor (0.2547),
map_50: tensor (0.4167),
map_75: tensor (0.2500),
map_large: tensor (0.2547),
map_medium: tensor (-1.),
map_per_class: tensor ([0.8762, 0.0000, 0.1424, 0.0000]),
map_small: tensor (-1.),
mar_1: tensor (0.0450),
mar_10: tensor (0.2825),
mar_100: tensor (0.2825),
mar_100_per_class: tensor ([0.8800, 0.0000, 0.2500, 0.0000]),
mar_large: tensor (0.2825),
mar_medium: tensor (-1.),
mar_small: tensor (-1.)}
```

Рис. 3.14 – Метрика продуктивності моделі виявлення об'єктів для діаграми

Кожна прогнозована обмежувальна область, позначена як клас, далі сегментується на три частини, які містять ім'я класу, атрибути та методи цього

UML-класу. Використовуючи морфологічні операції для виявлення всіх розділень (прямих горизонтальних ліній, що проходять через клас UML) та об'єднавши область, обмежену двома розділеннями в різних позиціях по осі y, в один відсоток. Щоб виявити лінії, які є не ідеально прямими, варто визначити ширину бажаної прямої лінії (str_element_width), яка дорівнює 7% від загальної ширини UML класу. Таке зменшення ширини структурного елемента може призвести до того, що для одного горизонтального розділення буде виявлено декілька ліній. Якщо для одного горизонтального розділення виявлено декілька прямих ліній, вони об'єднуються в одну лінію за умови, що відстань між будь-якими 2 лініями не перевищує вертикальний поріг (vertical_threshold) у 30%.

```
import cv2
import matplotlib.pyplot as plt
import time
def segment_uml_class(image, coordinates):
xmin, ymin, xmax, ymax = map(int, coordinates)
img = image.copy()[ymin: ymax, xmin: xmax]
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
thresh = cv2.threshold(img_gray, 127, 255,
cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]

temp = int(0.5 * len(img[0]))
str_element_width = int(0.07 * len(img[0]))
str_element_height = 1

horizontal_kernel = cv2.getStructuringElement(cv2.MORPH_RECT,
(str_element_width, str_element_height))
detect_horizontal = cv2.morphologyEx(thresh, cv2.MORPH_OPEN,
horizontal_kernel, iterations= 1)
cnts = cv2.findContours(detect_horizontal, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
```

Рис. 3.15 – Реалізація розпізнавання тексту частина №1

```
vertical_threshold = .3
count = 0
reduced_cnts = []
for c in cnts:
cv2.drawContours(img, [c], -1, (12, 255, 36), 1)
x, y, w, h = cv2.boundingRect(c)
y = y + (h // 2)
if not reduced_cnts:
reduced_cnts.append(y)
else:
if abs(reduced_cnts[-1] - y) / 100 <= vertical_threshold:
reduced_cnts[-1] = (reduced_cnts[-1] + y) // 2
else:
reduced_cnts.append(y)
imgplot = plt.imshow(img)
plt.show()
print(reduced_cnts)
return reduced_cnts
```

Рис. 3.16 – Реалізація розпізнавання тексту частина №2

```
bbox_segments = {}
captured_reln = []
for fb, fl in zip(filtered_boxes, filtered_labels):
if fl == 'class':

key = (int(fb[0]), int(fb[2]))
y_diff = int(fb[1])

y_pts = list(map(lambda x : x + y_diff, segment_uml_class(image, fb)))
class_segment = []
for i in range(1, len(y_pts)):
class_segment.append([y_pts[i-1], y_pts[i]])
bbox_segments[key] = class_segment
else:
captured_reln.append(fl)

for k, v in bbox_segments.items():
fig, axs = plt.subplots(1, 3)
try:
for i in range(len(v)):
axs[i].imshow(
image[v[i][1]: v[i][0], k[0]: k[1]]
)
except IndexError:
print("Exception occured in parsing UML class. \n \
Reasons could be, \n \
[wrong identification,\n \
threshold for merging contours,\n \
missing attributes or methods for class]\n \
")
pass
```

Рис. 3.17 – Реалізація розпізнавання тексту частина №3

Усі сегменти класів UML пропускаються через фреймворк для вилучення тексту easyocr, який використовує CRAFT для виявлення текстових областей у кожному з цих сегментів і CRNN для розпізнавання тексту, що міститься всередині текстових областей. Для розпізнавання текстів ми використовували попередньо навчену модель розпізнавання тексту, яка автоматично завантажується при виклику методу Reader. Вихідний текст, показаний на рисунку 3.18, знаходиться в порядку методів класу, атрибутів класу та назви класу для кожного класу UML. Крім того, всі виявлені зв'язки між класами UML також зображено на Рисунку 3.18. Розпізнаний текст для всіх сегментів класу кожного виявленого UML-класу

```
import easyocr
from torchmetrics.text.wer import WER
from torchmetrics.text.cer import CharErrorRate
text_ground_truth = ['+ play(): bool + bark(): bool',
'+ name: str + breed: str + color: Color',
'Dog', '# getOccupation(): Job # setOccupation(Job): bool',
'+ name: str - age: int + gender: Enum',
'Person',
'+ chewClothes(): int',
'+ retrievalSpeed: int',
'Retriever',
'+ gotoDuty(): Time',
'+ aggresiveScore: int', 'Shepherd',
'+ address: Address - members: int',
'Family']

reader = easyocr.Reader(['en'])
print("Text-prediction in detected objects: \n")
text_predictions = []
for k, v in bbox_segments.items():
text_in_each_class = []
for i in range(len(v)):
temp = reader.readtext(image[v[i][1]: v[i][0], k[0]: k[1]],
detail = 0, x_ths = 10, paragraph = True)
print(temp)
text_in_each_class.extend(temp)
print("\n")
text_predictions.extend(text_in_each_class)
print("Captured Relationships: ", captured_reln, "\n\n")
```

Рис. 3.18 – Реалізація обчислення коефіцієнту помилок символів

```
Text-prediction in detected objects:

[tPlag () ; boo | tbark () : boo/]
[t name : Sfy breed : stv tcoloy Coloy]
[Pod]

[#qetOccpation (): Job #sel occpotion (Job): booll]
[tname : Str age iot + gender Enum]
[Person]

[Chew Clothes () : in&]
[tvetrievlS peed : inl]
["Retriever"]

[tgoto Att ( ; Time]
[todgresivescove :int]
[Shepherd]

[]
[taddyess Address ~members : it]
[Family]

Captured Relationships: [inheritance, inheritance, inheritance]
```

Рис. 3.19 – Обчислення коефіцієнту помилок символів

Обчислення коефіцієнту помилок символів (CER) і коефіцієнту помилок слів (WER) для всього розпізнаного тексту, порівнюючи його з оригінальним текстом на вхідному зображенні. Результати показано на рисунку 3.19.

```python
word_metric = WER()
word_error = word_metric(text_predictions, text_ground_truth)
print("Word error rate: ", word_error)
char_metric = CharErrorRate()
char_error = char_metric(text_predictions, text_ground_truth)
print("Character error rate: ", char_error)
```

Рис. 3.20 – Реалізація коду для відображення частоти помилок

```
Word error rate: tensor(0.0337)
Character error rate: tensor(0.3043)
```

Рис. 3.21 – Частота помилок у словах і символах

Частота помилок у словах і частота помилок у символах для виявленого та розпізнаного тексту зображено на рисунку 3.21.

Висновок до розділу 3

Розробка системи для аналізу UML діаграм за допомогою згорткових нейронних мереж (CNN) продемонструвала гарні результати автоматизації процесів у галузі програмного забезпечення. В рамках цього дослідження було успішно показано, що застосування сучасних методів машинного навчання до аналізу UML діаграм може значно підвищити ефективність і точність роботи з великими та складними системами. Було розроблено систему для аналізу UML діаграм з використанням згорткових нейронних мереж (CNN). Проведене дослідження та експериментальні результати дозволяють зробити наступні висновки:

- Ефективність Аналізу UML Діаграм за допомогою CNN. Використання згорткових нейронних мереж для аналізу UML діаграм показало гарну ефективність та точність. Завдяки здатності CNN автоматично виявляти й вивчати суттєві ознаки на зображеннях, система змогла успішно класифікувати та інтерпретувати різні типи діаграм, такі як діаграми класів, діаграми послідовностей та інші.
- Зменшення Трудомісткості. Використання CNN значно зменшило потребу у ручній роботі для виділення ознак на діаграмах. Традиційні методи аналізу UML діаграм, що вимагали ручного втручання, були замінені автоматизованими підходами, що суттєво підвищило ефективність та швидкість обробки.
- Попередня Обробка Зображень: Для забезпечення високої точності розпізнавання UML діаграм, були використані різні методи попередньої обробки зображень, такі як вирівнювання гістограми, виявлення країв та проріджування країв. Ці методи допомогли покращити якість вхідних зображень та забезпечили більш точне розпізнавання контурів і елементів діаграм.

- Виявлення Об'єктів CNN. Методи виявлення об'єктів, такі як CNN, були успішно інтегровані в систему для апроксимації областей UML діаграм. Це дозволило точно ідентифікувати різні елементи діаграм, такі як класи, асоціації та інші класифікатори, навіть у складних випадках.
- Перспективи Подальших Досліджень. Отримані результати відкривають перспективи для подальших досліджень у напрямку вдосконалення системи. Можливі покращення включають використання більш складних архітектур нейронних мереж, застосування додаткових методів попередньої обробки зображень та інтеграцію інших видів діаграм для більш комплексного аналізу.

Таким чином, розробка системи для аналізу UML діаграм (діаграми класів) з використанням згорткових нейронних мереж показала гарний результат, продемонструвавши перспективність такого підходу.

ВИСНОВОК

В процесі роботи над дипломною роботою було досліджено різні аспекти використання UML (Unified Modeling Language) для моделювання великих і складних систем. UML визначає набір діаграм, які можна класифікувати на структурні та поведінкові, що дозволяє ефективно представляти концептуальні або фізичні аспекти моделей за допомогою класифікаторів, таких як класи, інтерфейси та типи асоціацій.

Особлива увага була приділена діаграмам класів, які є статичними структурними діаграмами, що використовуються для концептуального моделювання програм і опису класів та їх взаємозв'язків. Класи в UML описують набір об'єктів з загальними ознаками, обмеженнями та семантикою. Діаграма класів дозволяє здійснювати абстракцію, приховуючи деталі реалізації та сприяючи високорівневому проектуванню системи.

З іншого боку, розглянуто різні алгоритми машинного навчання, зокрема згорткові нейронні мережі (Convolutional Neural Networks, CNN), які широко використовуються для завдань комп'ютерного зору. CNN дозволяють автоматично виділяти ознаки зображень та використовують принципи лінійної алгебри, зокрема множення матриць, для виявлення шаблонів на зображенні. Незважаючи на вимогливість до обчислювальних ресурсів, що потребує використання графічних процесорів (GPU), CNN забезпечують більш масштабований підхід до класифікації зображень та розпізнавання об'єктів.

Комп'ютерний зір, як галузь штучного інтелекту, спрямований на отримання високорівневого розуміння та змістовної інформації з зображень та відео. Використовуючи методи згортки, комп'ютерний зір дозволяє досягти різних ефектів на зображеннях, таких як підвищення різкості, розмиття та виявлення країв. Традиційні методи виявлення об'єктів, такі як машини опорних векторів (SVM), поступаються місцем сучасним нейромережевим архітектурам, таким як Region Proposal Convolutional Neural Networks (RCNN) та Faster RCNN, що дозволяють автоматично виявляти і вивчати об'єкти на зображеннях.

У ході дослідження була розроблена система, завданням якої був аналіз UML діаграм з використанням згорткових нейронних мереж. Результати показали високу ефективність цього підходу. CNN змогли автоматично виділяти важливі ознаки UML діаграм та класифікувати їх з високою точністю, що підтвердило потенціал використання CNN для аналізу складних візуальних моделей в області інформаційних систем.

Таким чином, поєднання UML для моделювання систем та згорткових нейронних мереж для аналізу діаграм дозволяє створювати більш ефективні та автоматизовані системи проектування і аналізу програмного забезпечення. Результати цього дослідження можуть стати основою для подальших розробок у цій галузі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Beynon-Davies P. Business Information Systems. Red Globe Press, 2019. 510 с.
2. Rumpe B. Modeling with UML. Cham : Springer International Publishing, 2016. URL: https://doi.org/10.1007/978-3-319-33933-7 (дата звернення: 24.03.2024).
3. Sundaramoorthy S. Uml Diagramming. Taylor & Francis Group, 2022.
4. Farinha J., da Silva A. R. UML Templates Distilled. IEEE Access. 2022. Т. 10. С. 8709–8727. URL: https://doi.org/10.1109/access.2022.3143898 (дата звернення: 27.03.2024).
5. Конспект лекцій (опорний конспект лекцій). StudFiles. URL: https://studfile.net/preview/5064248/ (дата звернення: 27.03.2024).
6. Introducing Types of UML Diagrams | Lucidchart Blog. *Intelligent Diagramming Lucidchart.* URL: https://www.lucidchart.com/blog/types-of-UML-diagrams (дата звернення: 15.04.2024).
7. Biswal A. Top 10 Deep Learning Algorithms You Should Know in 2024. *Simplilearn.com*. URL: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm (дата звернення: 17.04.2024).
8. Blog - Create UML class diagrams. *draw.io*. URL: https://www.drawio.com/blog/uml-class-diagrams (дата звернення: 17.04.2024).
9. Friedland G. Information-Driven Machine Learning. Cham : Springer International Publishing, 2024. URL: https://doi.org/10.1007/978-3-031-39477-5 (дата звернення: 19.04.2024).
10. Planning for Machine Learning. Machine Learning. Indianapolis, IN, USA, 2015. С. 17–44. URL: https://doi.org/10.1002/9781119183464.ch2 (дата звернення: 19.04.2024).
11. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Incorporated, 2022.
12. Analyzing Types of Neural Networks in Deep Learning. *Analytics Vidhya*. URL: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-

analyzing-3-types-of-neural-networks-in-deep-learning/ (дата звернення: 19.04.2024).

13. El-Amir H., Hamdy M. Convolutional Neural Network. Deep Learning Pipeline. Berkeley, CA, 2019. С. 367–413. URL: https://doi.org/10.1007/978-1-4842-5349-6_11 (дата звернення: 20.04.2024).
14. Kanade V. What is machine learning? Understanding types & applications - Spiceworks. *Spiceworks Inc*. URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/ (дата звернення: 20.04.2024).
15. Moreno V. Automatic Classification of Web Images as UML Static Diagrams Using Machine Learning Techniques. *MDPI*. URL: https://www.mdpi.com/2076-3417/10/7/2406 (дата звернення: 23.04.2024).
16. Saha S. A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way. *https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53*. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (дата звернення: 23.04.2024).
17. Sundaramoorthy S. Uml Diagramming. Taylor & Francis Group, 2022.
18. Theobald O. Machine Learning for Absolute Beginners: A Plain English Introduction. Independently Published, 2021.
19. Milosevic N. Exposing Neural Network Models. Introduction to Convolutional Neural Networks. Berkeley, CA, 2020. URL: https://doi.org/10.1007/978-1-4842-5648-0_16 (дата звернення: 24.04.2024).
20. Milosevic N. Let’s Build a Neural Network!. Introduction to Convolutional Neural Networks. Berkeley, CA, 2020. URL: https://doi.org/10.1007/978-1-4842-5648-0_5 (дата звернення: 24.04.2024).
21. CHEN H.-P., Integrating Convolutional Neural Network and Recurrent Neural Network for Automatic Text Classification : thesis. 2019. URL: http://ndltd.ncl.edu.tw/handle/4jqh8z (дата звернення: 26.04.2024).

22. UML Diagram Types | Learn About All 14 Types of UML Diagrams. *Creately Blog*. URL: https://creately.com/blog/diagrams/uml-diagram-types-examples/ (дата звернення: 04.05.2024).
23. What is a Neural Network? | IBM. *IBM in Deutschland, Österreich und der Schweiz*. URL: https://www.ibm.com/topics/neural-networks (дата звернення: 04.05.2024).
24. What is Machine Learning? Types & Uses | Google Cloud. *Google Cloud*. URL: https://cloud.google.com/learn/what-is-machine-learning (дата звернення: 7.05.2024).

Додаток А

КОПІЇ ДЕМОНСТРАЦІЙНИХ МАТЕРІАЛІВ

Державний університет інформаційно-комунікаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Розробка алгоритмів машинного навчання для використання в інформаційних системах»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

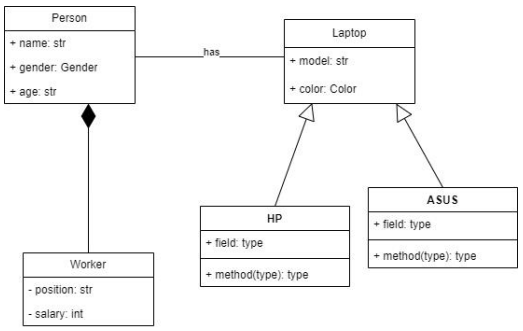
Виконав: Демченко Є.Ю, ІСД-41
Науковий керівник роботи:
Данильченко В.М.

Київ - 2024

- ***Актуальність теми:*** в наш час компанії і підприємства часто стикаються з великими обсягами інформації, які необхідно ефективно обробляти. Машинне навчання один із доступних способів аналізу даних, використовуючи який потенційно можна збільшити ефективність обробки інформації.
- ***Наукова новизна:*** Розробка та огляд ефективності використання згорткових нейронних мереж для аналізу UML діаграм в інформаційних системах.
- ***Об`єкт дослідження:*** Аналіз архітектури інформаційних систем.
- ***Предмет дослідження:*** Машинне навчання для обробки UML діаграм.
- ***Мета дослідження:*** Метою даного дослідження є розробка алгоритму для обробки і аналізу діаграм у інформаційних системах.
- ***Завдання дослідження:***
  - **1. Визначити що таке інформаційні системи і що в них входить.**
  - **2. Розглянути методи машинного навчання, які можна використати в моделюванні інформаційних систем.**
  - **3. Розробити модель машинного навчання для аналізу інформаційних систем і оцінити перспективи розвитку.**

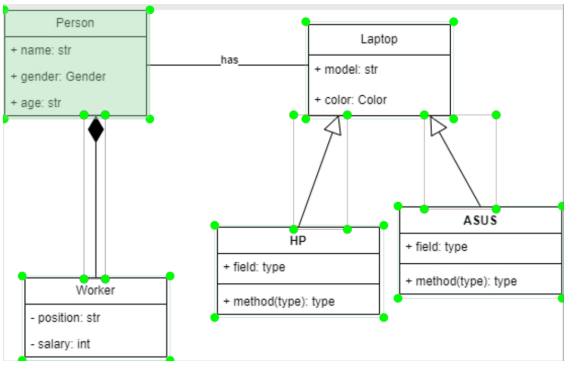
Огляд потенційних об’єктів дослідження та розробки в інформаційних системах

За результатами дослідження було прийнято рішення обрати за об’єкт дослідження UML діаграми, а саме діаграми класів. Оскільки ці діаграми використовуються для концептуального моделювання програми для опису класів та їх взаємозв'язків, дозволяючи проектувати систему на високому рівні. Також UML діаграми часто використовуються у сфері програмної інженерії та комп'ютеризованих систем.



Підготовка даних для навчання моделі

Для навчання були використанні анотацій в форматі xml для більш ефективного навчання моделі.



Навчання моделі розпізнавання об’єктів

Модель розпізнавання об'єктів навчалась за допомогою фреймворку detecto, який використовує Faster RCNN для навчання.

```
from detecto import core, visualize, utils
from detecto.core import Dataset, DataLoader
train_dataset = core.Dataset('/content/drive/My Drive/Train')
test_dataset = Dataset('/content/drive/My Drive/Test')

loader = core.DataLoader(train_dataset, batch_size= 5, shuffle= True)

labels_map = { 'class': 1,
'association': 2,
'aggregation': 3,
'realization': 4,
'dependency': 5,
'u-association': 6,
'inheritance': 7,
'composition': 8,
'agg-depy-agg': 9,
'aggregation-association': 10,
'box': 11,
'oval': 12
}
model = core.Model([*labels_map.keys()])
model.fit(train_dataset)

model.save("uml_weights.pth")
```


Реалізація розпізнавання тексту

Кожна прогнозована обмежувальна область, позначена як клас, далі сегментується на три частини, які містять ім'я класу, атрибути та методи цього UML-класу.

```
import cv2
import matplotlib.pyplot as plt
import time
def segment_uml_class(image, coordinates):
xmin, ymin, xmax, ymax = map(int, coordinates)
img = image.copy()[ymin: ymax, xmin: xmax]
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
thresh = cv2.threshold(img_gray, 127, 255,
cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1])

temp = int(0.5 * len(img[0]))
str_element_width = int(0.07 * len(img[0]))
str_element_height = 1

horizontal_kernel = cv2.getStructuringElement(cv2.MORPH_RECT,
(str_element_width, str_element_height))
detect_horizontal = cv2.morphologyEx(thresh, cv2.MORPH_OPEN,
horizontal_kernel, iterations= 1)
cnts = cv2.findContours(detect_horizontal, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
```

Обчислення коефіцієнту помилок

```
Text-prediction in detected objects:

[tPlag () ; boo | tbark () : boo/]
[t name : Sfy breed : stv tcoloy Coloy]
[Pod]

[#qetOccpation (): Job #sel occpotion (Job): booll]
[tname : Str age iot + gender Enum]
[Person]

[Chew Clothes () : in&]
[tvetrievlS peed : inl]
["Retriever"]

[tgoto Att ( ; Time]
[todgresivescove :int]
[Shepherd]

[]
[taddyess Address ~members : it]
[Family]

Captured Relationships: [inheritance, inheritance, inheritance]
```

```
import easyocr
from torchmetrics.text.wer import WER
from torchmetrics.text.cer import CharErrorRate
text_ground_truth = ['+ play(): bool + bark(): bool',
'+ name: str + breed: str + color: Color',
'Dog', '# getOccupation(): Job # setOccupation(Job): bool',
'+ name: str - age: int + gender: Enum',
'Person',
'+ chewClothes(): int',
'+ retrievalSpeed: int',
'Retriever',
'+ gotoDuty(): Time',
'+ aggresiveScore: int', 'Shepherd',
'+ address: Address - members: int',
'Family']

reader = easyocr.Reader(['en'])
print("Text-prediction in detected objects: \n")
text_predictions = []
for k, v in bbox_segments.items():
text_in_each_class = []
for i in range(len(v)):
temp = reader.readtext(image[v[i][1]: v[i][0], k[0]: k[1]],
detail = 0, x_ths = 10, paragraph = True)
print(temp)
text_in_each_class.extend(temp)
print("\n")
text_predictions.extend(text_in_each_class)
print("Captured Relationships: ", captured_reln, "\n\n")
```


Результати обчислення коефіцієнту помилок

Обчислення коефіцієнту помилок символів (CER) і коефіцієнту помилок слів (WER) для всього розпізнаного тексту, порівнюючи його з оригінальним текстом на вхідному зображенні.

На виході отримуємо наступний результат:

- Коефіцієнт помилок символів ≈ 30%
- Коефіцієнт помилок слів ≈ 5%

```
word_metric = WER()
word_error = word_metric(text_predictions, text_ground_truth)
print("Word error rate: ", word_error)
char_metric = CharErrorRate()
char_error = char_metric(text_predictions, text_ground_truth)
print("Character error rate: ", char_error)
```

```
Word error rate: tensor(0.0337)
Character error rate: tensor(0.3043)
```

Висновки

- Поєднання UML для моделювання систем та згорткових нейронних мереж для аналізу діаграм класів дозволяє досить ефективно збирати дані спираючись на результати дослідження.

- Апробація:

  Всеукраїнська науково-технічна конференція “Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу”, 28 листопада 2024 року ДУІКТ – “Використання Machine Learning для аналізу UML діаграм”