Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія фільтрації зображень»

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

Керівник: к.т.н., доцент кафедри КН
Барабан С. В.
(прізвище та ініціали)
«15» 12 2022 р.

Опонент: ктн., доцент кафедри КСУ
Юхимчук М. С.
(прізвище та ініціали)
«15» 12 2022 р.

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

Вінниця ВНТУ 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.

14.09 **2022 року**

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Сулима Ярослав Олександрович
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія фільтрації зображень
керівник роботи к.т.н., доцент кафедри КН Барабан С. В.
затвердженні наказом вищого навчального закладу від «14» 09 2022 року №
2. Строк подання студентом роботи 11 листопад 2022 року
3. Вихідні дані до роботи:
вхідні дані – об'єктно орієнтована мова програмування яка підтримує бібліотеку TensorFlow; програма має працювати як на ОС Windows з 32 так і з 64 розрядною системою; програма має підтримувати обробку найпоширеніших форматів зображень, таких як: png, jpg, jpeg, webp; обсяг навчальної вибірки – не менше 100; програма повинна виводити чіткий результат та обробляти помилки; розмір програми не має перевищувати 500Мб.
4. Зміст текстової частини:
Вступ, основні поняття предметної області; вибір засобів реалізації інформаційної технології аналізу зображень; проектування та розробка інформаційної системи, економічна частина, висновки, перелік використаних джерел, додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
UML діаграма класів, діаграма способів використання інформаційної системи, фрагмент схеми алгоритму.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціалита посада консультанта	Підпис, дата	
		завдання видав	виконан прийня
1-3	Барабан С. В., к.т.н., доц. каф. КН	14.09.2022	14.11.202
4	Бурєннікова Н. В., д. е. н., проф. каф. ЕПВМ	19.09.2022	12.12.202

7. Дата видачі завдання 14.09 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапу	Строк виконання роботи	Примітка
1	Основні поняття предметної області	14.09.2022р - 01.10.2022р.	Розділ 1
2	Вибір засобів реалізації	02.10.2022р - 16.10.2022р.	Розділ 2
3	Проектування та розробка системи	17.10.2022-07.11.2022	Розділ 3
4	Підготовка економічної частини	08.11.2022-21.11.2022	Розділ 4
5	Апробація результатів дослідження	23.11.2022р - 01.12.2022р	тези доповіді
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	02.12.2022 - 14.12.2022р.	Підготовка та захист граф. матеріалів, презентація

Студент _____ (підпис) Сулима Я. О.

Керівник роботи _____ (підпис) Барабан С. В.

АНОТАЦІЯ

УДК 004.8

Сулима Я. О. Інформаційна технологія фільтрації зображень. Магістерська кваліфікаційна робота зі спеціальності 122 – комп'ютерні науки, освітня програма - комп'ютерні науки. Вінниця: ВНТУ, 2022. 98 с.

На укр. мові. Бібліогр.: 21 назв; рис.: 2; табл. 16.

В даній магістерській кваліфікаційній роботі було реалізовано інформаційну систему фільтрації зображень з використанням штучного інтелекту, призначену для відсікання небажаного фотографічного контенту (фото зброї, наркотичних речовин, насильства та контенту для дорослих) безпосередньо перед збереженням на сервер, що в свою чергу допоможе економити місце і кошти, а також запобіжить поширенню небажаного контенту. Даний програмний продукт призначений для запуску в хмарі (ASW Lambda functions, Google cloud functions…). Здійснено аналіз предметної області, розглянуто основні поняття та визначення, описано явище та методи фільтрації контенту, наведено парадигми навчання нейронної мережі. Проведено вибір програмних засобів для реалізації інформаційної системи. Наведено особливості мови програмування Python, описано основні бібліотеки для реалізації нейронної мережі.

Розроблено структурну схему та алгоритм функціонування інформаційної системи фільтрації зображень. Розроблено UML діаграму класів та UML діаграму використання.

Проведено економічний аналіз на доцільність розробки даної інформаційної системи, підраховано витрати, в результаті чого зроблено висновок про доцільність розробки даного продукту.

Ключові слова: штучний інтелект, нейронні мережі, патерн, штучний інтелект, клієнт–сервер, TensorFlow, Keras, структурна схема, алгоритм, UML–діаграма.

ABSTRACT

Sulyma Y. O. Information technology of image filtering. Master's thesis in the specialty 122 - computer sciences, educational program - computer science. Vinnytsia: VNTU, 2022. 98 p.

In Ukrainian language. Bibliographer: 21 titles; fig .: 2; table 16.

In this master, work was developed an image-filtering information system using artificial intelligence, designed to reject unwanted photographic content (photos of weapons, drugs, violence and adult content) before saving to the server, which in turn will help save space in storage and money, and will prevent the spread of unwanted content. This software product is designed to run in the cloud (ASW Lambda functions, Google cloud functions...). An analysis of the subject area was carried out, the main concepts and definitions were considered, the phenomenon and methods of content filtering were described, and neural network training paradigms were given. A selection of software tools for the implementation of the information system was made. The features of the Python programming language are presented, the main libraries for implementing a neural network are described, as a result of which the TensorFlow and Keras libraries were chosen.

The structure, scheme and algorithm of functioning of the image filtering information system have been developed. Developed UML class diagram and UML usage diagram.

An economic analysis was conducted on the feasibility of developing this information system, the costs were calculated, as a result of which a conclusion was drawn on the feasibility of developing this product.

Keywords: artificial intelligence, neural networks, pattern, artificial intelligence, client-server, TensorFlow, Keras, structural diagram, algorithm, UML-diagram.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Завдяки онлайн-технологіям люди отримали величезні можливості для спілкування, набуття нових навичок, творчості та участі у створенні кращого суспільства.

В сучасному світі все більш активно набирає обертів процес діджиталізації і інтегрування все більшої кількості програмних продуктів в життя рядового члену соціуму. Кожного дня з’являються нові додатки, які допомагають людям генерувати терабайти контенту, такі як TikTok, Instagram, Telegram, Linked In, тощо. Проте, зі збільшенням кількості контенту збільшується і кількість матеріалів, що порушують правила платформи або закони країни, громадянином якої є користувач. Кожного дня в соціальних мережах з’являються фото і відеоматеріали, на яких присутні насильство, паління, алкоголь, зброя, пряма або непряма пропаганда наркотиків, тощо.

Виходячи з усього вищесказаного можна зробити висновок про високу актуальність створення різного роду інструментів для автоматичної фільтрації контенту.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання досліджень. Метою магістерської кваліфікаційної роботи є підвищення ефективності ідентифікації даних типу шкідливий контент, за допомогою розробки інформаційної технології фільтрації зображень.

Для досягнення поставленої задачі необхідне вирішити наступні завдання:

1. Огляд та аналіз існуючих рішень;

2. Дослідження вимог, методів і алгоритмів вирішення поставленого завдання;

3. Модифікація існуючих методів аналізу зображень на вміст шкідливого контенту та визначення наявності обличчя на фото та порівняння з існуючими;

4. Розроблення структури програмного забезпечення;

5. Створення прикладної програми – інформаційної системи аналізу зображень для соціальних мереж.

Об'єкт дослідження процес аналізу зображень на вміст шкідливого контенту.

Предметом дослідження методи та засоби аналізу зображень за допомогою технологій комп'ютерного зору.

Методи досліджень. Для створення моделі аналізу та проектування структури програмного засобу було використано методи об'єктно-орієнтованого аналізу та проектування.

Наукова новизна роботи полягає в розробці інформаційної технології фільтрації зображень із використанням згорткової нейронної мережі, що дозволяє підвищити точність ідентифікації даних на наявність шкідливого контенту.

Практичне значення одержаних результатів полягає у створенні інформаційної системи, яка дозволить аналізувати завантажувані користувачем зображенням, перед їм збереженням у БД, а також аналізувати уже наявні зображення в БД, вести аналітику, та блокувати користувачів, на основі багатократних порушень політики безпеки зображень соціальної мережі.

Апробація результатів роботи. Результати роботи були апробовані на конференції «Молодь в науці: дослідження, проблеми, перспективи-2023» (м. Вінниця, Україна, 2023 р.) [1],

Публікації. За результатами досліджень опубліковано тези доповіді на конференції.

1 ОСНОВНІ ПОНЯТТЯ ПРЕДМЕТНОЇ ОБЛАСТІ ФІЛЬТРАЦІЇ ЗОБРАЖЕНЬ

1.1 Явище фільтрації контенту

Інтернет-фільтр — це програмне забезпечення, яке обмежує або контролює вміст, до якого може отримати доступ користувач Інтернету, особливо коли він використовується для обмеження матеріалів, що доставляються через соціальні мережі, електронну пошту чи іншими засобами. Програмне забезпечення для контролю вмісту визначає, який вміст буде доступним або заблокованим.

Такі обмеження можуть застосовуватися на різних рівнях: уряд може спробувати застосувати їх по всій країні (див. Інтернет-цензура), або вони можуть, наприклад, застосовуватися постачальником послуг Інтернету до своїх клієнтів, роботодавцем до свого персоналу, школою для своїх учнів, бібліотекою для відвідувачів, батьками до комп'ютера дитини або окремими користувачами до власних комп'ютерів.

Мотивом часто є запобігання доступу до вмісту, який власник(и) комп'ютера або інші органи влади можуть вважати небажаними. Якщо контроль вмісту вводиться без згоди користувача, його можна охарактеризувати як форму інтернет-цензури. Деяке програмне забезпечення для контролю вмісту містить функції контролю часу, які дають змогу батькам визначати кількість часу, який дитина може витрачати на доступ до Інтернету, ігри чи іншу діяльність за комп'ютером.

У деяких країнах таке програмне забезпечення є повсюдним. На Кубі, якщо користувач комп'ютера в підконтрольному уряду Інтернет-кафе вводить певні слова, текстовий процесор або веб-браузер автоматично закриваються, і видається попередження про «державну безпеку».

Термін «контроль вмісту» час від часу вживають CNN, журнал Playboy, San Francisco Chronicle і The New York Times [2]. Проте кілька інших термінів, зокрема «програмне забезпечення для фільтрації вмісту», «проксі-сервери фільтрації», «захищені веб-шлюзи», «цензурне програмне забезпечення», «безпека та контроль вмісту», «програмне забезпечення для фільтрації веб-сайтів», «програмне забезпечення для цензури вмісту» та часто використовуються "програмне забезпечення для блокування вмісту". «Няня» також використовувалася як у маркетингу продукції, так і в ЗМІ. Галузева дослідницька компанія Gartner використовує «безпечний веб-шлюз» (SWG) для опису сегмента ринку.

Компанії, які виробляють продукти, які вибірково блокують веб-сайти, не називають ці продукти цензурним програмним забезпеченням і віддають перевагу таким термінам, як «Інтернет-фільтр» або «URL-фільтр»; у спеціальному випадку програмного забезпечення, спеціально розробленого для того, щоб дозволити батькам контролювати та обмежувати доступ своїх дітей, також використовується «програмне забезпечення для батьківського контролю». Деякі продукти реєструють усі сайти, до яких відвідує користувач, і оцінюють їх на основі типу вмісту для звітування «партнеру з підзвітності» за вибором особи, і використовується термін програмне забезпечення для підзвітності. Інтернет-фільтри, програмне забезпечення батьківського контролю та/або програмне забезпечення звітності також можна об’єднати в один продукт.

Однак ті, хто критикує таке програмне забезпечення, вільно використовують термін «цензурне програмне забезпечення»: розглянемо, наприклад, проект Censorware. Використання терміну «цензурне програмне забезпечення» в редакційних статтях, які критикують виробників такого програмного забезпечення, є широко поширеним і охоплює багато різних різновидів і застосувань: Ксені Джардін використала цей термін у редакційній статті в The New York Times від 9 березня 2006 року, коли обговорювалася використання американської фільтрації. програмне забезпечення для

придушення вмісту в Китаї; того ж місяця старшокласник використав цей термін, щоб обговорити розгортання такого програмного забезпечення в своєму шкільному окрузі.

Загалом, крім редакційних сторінок, як описано вище, традиційні газети не використовують термін «цензурне програмне забезпечення» у своїх репортажах, віддаючи перевагу використанню менш відверто суперечливих термінів, таких як «фільтр вмісту», «контроль вмісту» або «веб-фільтрація». ; New York Times і The Wall Street Journal, здається, дотримуються цієї практики. З іншого боку, веб-газети, такі як CNET, використовують цей термін як у редакційному, так і в журналістському контекстах, наприклад «Windows Live to Get Censorware» [2].

1.2 Методи фільтрації контенту

Фільтри можуть бути реалізовані багатьма різними способами: за допомогою програмного забезпечення на персональному комп’ютері, через мережеву інфраструктуру, таку як проксі-сервери, DNS-сервери або брандмауери, які забезпечують доступ до Інтернету. Жодне рішення не забезпечує повного охоплення, тому більшість компаній розгортають поєднання технологій для досягнення належного контролю вмісту відповідно до своєї політики.

Браузерні фільтри.

Рішення для фільтрації вмісту на основі веб-переглядача є найпростішим рішенням для фільтрування вмісту та реалізується через стороннє розширення для браузера.

Фільтри електронної пошти

Фільтри електронної пошти діють на інформацію, що міститься в тілі листа, у заголовках листів, як-от відправник і тема, а також на вкладення електронної пошти, щоб класифікувати, приймати чи відхиляти повідомлення.

Зазвичай використовуються фільтри Байєса, тип статистичного фільтра. Доступні як клієнтські, так і серверні фільтри.

Клієнтські фільтри.

Цей тип фільтра встановлюється як програмне забезпечення на кожному комп'ютері, де потрібна фільтрація. Зазвичай цим фільтром може керувати, вимикати чи видаляти кожен, хто має права адміністратора в системі. Клієнтський фільтр на основі DNS мав би налаштувати DNS Sinkhole, наприклад Pi-Hole.

Інтернет-провайдери з обмеженим (або відфільтрованим) вмістом.

Інтернет-провайдери з обмеженим (або фільтрованим) вмістом — це постачальники Інтернет-послуг, які пропонують доступ лише до певної частини Інтернет-вмісту за згодою або в обов'язковому порядку. Кожен, хто підписався на цей тип послуг, підпадає під обмеження. Тип фільтрів може бути використаний для здійснення державного регуляторного[15] або батьківського контролю над абонентами.

Мережева фільтрація.

Цей тип фільтра реалізується на транспортному рівні як прозорий проксі або на прикладному рівні як веб-проксі. Програмне забезпечення для фільтрації може містити функцію запобігання втраті даних для фільтрації вихідної та вхідної інформації. Усі користувачі підлягають політиці доступу, визначеній установою. Фільтрування можна налаштувати, тому бібліотека середньої школи шкільного округу може мати інший профіль фільтрації, ніж бібліотека молодшої школи округу.

Фільтрація на основі DNS.

Цей тип фільтрації реалізовано на рівні DNS і намагається запобігти пошуку доменів, які не відповідають набору політик (або батьківського контролю, або правил компанії). Кілька безкоштовних публічних служб DNS пропонують параметри фільтрації як частину своїх послуг. DNS Sinkholes, наприклад Pi-Hole, також можна використовувати для цієї мети, але лише на стороні клієнта.

Фільтри пошукових систем.

Багато пошукових систем, таких як Google і Bing, пропонують користувачам можливість увімкнути фільтр безпеки. Коли цей фільтр безпеки активовано, він відфільтровує невідповідні посилання з усіх результатів пошуку. Якщо користувачі знають фактичну URL-адресу веб-сайту, який містить відвертий або дорослий вміст, вони мають можливість отримати доступ до цього вмісту без використання пошукової системи. Деякі провайдери пропонують орієнтовані на дітей версії своїх двигунів, які дозволяють лише веб-сайти, орієнтовані на дітей [2].

1.3 Поняття нейронної мережі

Штучні нейронні мережі (ANN), які зазвичай просто називають нейронними мережами (NN), - це обчислювальні системи, нечітко натхненні біологічними нейронними мережами, які складають мозок тварин [2].

ANN базується на колекції з’єднаних одиниць або вузлів, званих штучними нейронами, які вільно моделюють нейрони біологічного мозку. Кожне з'єднання, як синапси в біологічному мозку, може передавати сигнал іншим нейронам. Штучний нейрон, який отримує сигнал, потім обробляє його і може сигналізувати про нейрони, підключені до нього. "Сигнал" у зв'язку є дійсним числом, і вихід кожного нейрона обчислюється якоюсь нелінійною функцією суми його входів. З'єднання називаються ребрами. Нейрони та краї зазвичай мають вагу, яка регулюється в процесі навчання. Вага збільшує або зменшує силу сигналу при з'єднанні. Нейрони можуть мати такий поріг, що сигнал надсилається лише в тому випадку, якщо сукупний сигнал перетинає цей поріг. Як правило, нейрони агрегуються в шари. Різні шари можуть виконувати різні перетворення на своїх входах. Сигнали рухаються від першого шару (вхідного шару) до останнього шару (вихідного шару), можливо, після багаторазового обходу шарів [2].

Уоррен Маккалок і Уолтер Піттс (2) (1943) почали цю тему зі створення обчислювальних моделей для нейронних мереж. Наприкінці 1940-х років Д. О. Хебб запропонував гіпотезу навчання, засновану на механізмі нейропластичності, а саме навчання Хебба. Фарлі та Уеслі А. Кларк [6] (1954) вперше використали комп'ютери (пізніше названі «калькуляторами») для моделювання мережі Хеббі. Розенблат (1958) створив персептрон. Івахненко та Лапа опублікували першу багатошарову функціональну мережу в 1965 році як груповий метод обробки даних. Основні знання про безперервне зворотне поширення походять з теорії керування Келлі у 1960 році та Брайсона у 1961 році, використовуючи принципи динамічного програмування [2].

У 1970 році Сеппо Лінненмаа опублікував загальний метод автоматичного диференціювання (AD) дискретних зв'язаних мереж вкладених диференціальних функцій [2]. У 1973 році Дрейфус використав зворотне поширення для адаптації параметрів контролера пропорційно градієнту помилки [3]. Алгоритм зворотного поширення Вербоса (1975) дозволяє практичне навчання багаторівневим мережам. У 1982 році він застосував метод А. Д. Лінненмаа до нейронних мереж, який широко використовувався [4]. Після Мінського та Паперта (1969) [5] дослідження припинилися, і вони виявили, що базовий персептрон не може обробляти схему XOR, а комп'ютер не має можливості обробляти корисні нейронні мережі.

Розвиток великомасштабної інтеграції металооксидних напівпровідників (MOS) у вигляді додаткової технології MOS (CMOS) збільшив кількість MOSFET в цифровій електроніці. Це забезпечило більшу обчислювальну потужність для розробки практичних штучних нейронних мереж у 1980-х роках [6]. У 1992 році було введено максимальне витягування, щоб допомогти мінімальній інваріантності зміщень і толерантності до деформації, щоб допомогти розпізнавати об'єкти в 3D [7]. Шмідхубер прийняв багаторівневу мережеву ієрархію (1992), підготовлену заздалегідь на одному рівні за допомогою неконтрольованого навчання та вдосконалену шляхом зворотного поширення.

Джеффрі Хінтон чекав. (2006) запропонували використовувати безперервні шари двійкових або реальних прихованих змінних і кінцеву машину Больцмана [8] для моделювання кожного шару для вивчення високорівневих уявлень. У 2012 році Нг і Дін створили мережу, яка вчиться розпізнавати поняття вищого рівня, такі як кішки, лише дивлячись на зображення без міток. Неконтрольоване попереднє навчання та збільшення обчислювальної потужності від графічних процесорів і розподілених обчислень дозволяють використовувати більші мережі, особливо для розпізнавання зображень і проблем із зором, що називається «глибоким навчанням» [9].

Ciresan та його колеги (2010) [10] показали, що, незважаючи на проблему зникаючих градієнтів, графічні процесори дозволяють зворотне поширення прямого зв'язку багатошарових нейронних мереж [11]. З 2009 по 2012 рр. Академія наук почала займати призові місця в конкурсах Академії наук Виконання різноманітних завдань, близьких до людського, спочатку було у розпізнаванні образів та машинному навчанні [12]. Наприклад, двонаправлена та багатовимірна довготривала короткочасна пам'ять (LSTM) Грейвса та інших [13]. У 2009 році переміг у трьох суміжних конкурсах з розпізнавання рукописного тексту, але не мав жодного попереднього знання трьох мов, що вивчаються [14].

Ciresan та його колеги створили перший розпізнавач шаблонів, щоб досягти конкурентоспроможності людини/надлюдини за такими критеріями, як розпізнавання дорожніх знаків (IJCNN 2012) [15].

Нейронні мережі навчаються, обробляючи приклади. Кожен приклад містить відомі «вхідні дані» та «результати», які утворюють між ними зважену за ймовірністю асоціацію. Ці асоціації зберігаються в структурі даних самої мережі. Навчання нейронної мережі з даного прикладу зазвичай здійснюється шляхом визначення різниці між обробленим виходом мережі (зазвичай передбачуваним) і цільовим результатом. Це помилка. Потім мережа коригує свої зважені асоціації відповідно до правил навчання і використовуючи це

значення помилки. Регулювання послідовності змусить нейронну мережу видавати результати, які все більше і більше схожі на цільові результати. Після внесення достатньої кількості цих налаштувань навчання може бути припинено за певними критеріями. Це називається контрольованим навчанням.

Такі системи «вчаться» виконувати завдання на розгляді прикладів, і зазвичай не потрібно писати певні правила для конкретних завдань. Наприклад, під час розпізнавання зображень вони можуть навчитися розпізнавати зображення, які містять котів, аналізуючи приклади зображень, позначених вручну як «кіт» або «без кішки», і використовуючи результати для ідентифікації котів на інших зображеннях. До цього вони нічого не знали про кішок, наприклад, у них є шерсть, хвости, вуса і котячі мордочки. Замість цього вони автоматично створюватимуть функції розпізнавання на основі оброблених прикладів [16].

Штучні нейронні мережі складаються з штучних нейронів, які концептуально походять від біологічних нейронів. Кожен штучний нейрон має вхід і виробляє вихід, який можна надіслати кільком іншим нейронам. Вхідними даними можуть бути значення ознак зовнішніх зразків даних, таких як зображення чи документи, або вихідні дані інших нейронів. Кінцевий вихід нейронної мережі Вихід нейрона виконує такі завдання, як розпізнавання об'єктів на зображенні [17].

Щоб знайти вихід нейрона, ми спочатку беремо зважену суму всіх вхідних даних, зважену на вагу з’єднання входу з нейроном. З цією метою ми додали термінологічне упередження. Цю зважену суму іноді називають активацією. Ця зважена сума потім передається через (зазвичай нелінійну) функцію активації для отримання вихідних даних. Вихідними даними є зовнішні дані, наприклад зображення та документи. Кінцевий результат виконує такі завдання, як розпізнавання об'єктів на зображеннях [18].

Модель нейронної мережі описує її архітектуру та конфігурацію, а також використовуваний алгоритм навчання.

Архітектура нейронної мережі визначає загальні принципи її побудови (площина шарування, повне з’єднання, слабке з’єднання, пряме поширення, циркуляція тощо).

Конфігурація визначає структуру мережі в межах заданої архітектури: кількість нейронів, кількість входів і виходів функції активації, що використовується мережею.

Розрізняють такі базові архітектури:

1. мережі прямого поширення - всі зв'язки направлені строго від вхідних нейронів до вихідних. До таких мереж відносяться, наприклад персептрон Розенблатта і багатошаровий персептрон;

2. рекурентні нейронні мережі - сигнал з вихідних нейронів або нейронів прихованого шару частково передається назад на нейрони вхідного шару мережі;

3. мережі радіально-базисних функцій - мережі, що містять єдиний прихований шар, нейрони якого використовують радіально-симетричну активаційну функцію, застосовуються для вирішення задач класифікації та прогнозування;

4. мережі Кохонена – клас мереж, що використовують навчання без учителя і призначених для вирішення завдань кластеризації. Вони містять всього два прошарки: вхідний (розподільний) і вихідний (кластеризуються);

5. карти Кохонена або карти ознак, що самоорганізуються - різновид мереж Кохонена, в яких число вихідних нейронів вибирається багато більше числа формованих кластерів. Використовуються для візуалізації результатів кластеризації багатовимірних даних;

6. повнозв'язні мережі - нейронні мережі, в яких кожен нейрон пов'язаний з усіма іншими нейронами. Такі мережі мають найвищу щільність зв'язків;

7. слабозв'язаних мережі - в них нейрони з'єднані тільки зі своїми найближчими сусідами;

8. нейронні мережі з плоскими шарами - в них нейрони утворюють каскади, звані шарами, при цьому нейрони кожного шару пов'язані з усіма нейронами наступного і попереднього шарів, а всередині шару зв'язків немає. Мережі з плоскими шарами можуть бути одношарові (містити один прихований шар) і багатошаровими (містити кілька прихованих шарів).

Кожна архітектура мережі призначення для вирішення певного класу задач аналізу даних (регресії, класифікації, кластеризації, прогнозування) і використовує спеціальні алгоритми навчання.

Поки що найпоширенішими видами стали такі типи нейронних мереж: - Згорткові нейронні мережі (CNN) імітують роботу зорової кори головного мозку і частково виконують функції абстрактного мислення. Вони чудово справляються із завданнями розпізнавання зображень, а їх обчислення легко розпаралелювати на графічному процесорі, що дозволяє використовувати елементи AI для створення відносно недорогих апаратних платформ.
CNN використовується в системах машинного зору для безпілотних транспортних засобів, комерційних дронів, роботів і охоронного відеоспостереження. Якщо увімкнути розблокування розпізнавання обличчя в налаштуваннях смартфона, ви також зможете користуватися CNN щодня.

-Рекурентна нейронна мережа (RNN) має короткочасну пам'ять, тому легко аналізувати послідовності будь-якої довжини. RNN розділяє потік даних на основні частини і оцінює взаємозв'язок між ними. Ці алгоритми в основному використовувалися для розпізнавання рукописного тексту та мови.Коли ви шукаєте мелодію на слух в Shazam, розмовляєте з Siri, Google Now або Алісою від «Яндекса», залишаєте замітки від руки для Cortana - на хмарних платформах беруться за справу рекурентні нейромережі.

- мережі з довготривалою і короткочасною пам'яттю (LSTM) стали подальшим розвитком RNN. Вони гарні для прогнозування змін будь-якої величини (наприклад, біржових курсів або купівельного попиту) шляхом екстраполяції. Також їх застосовують для глибокого аналізу природної мови. Наприклад, Google використовує LSTM в персональному помічника і системі

машинного перекладу Google Translate. Без LSTM якість перекладів так і залишалося б на рівні програм з дев'яностих, з якими часом було простіше перевести текст самому, ніж виправляти численні помилки.

– Controlled Recurrent Unit (GRU) - відносно нова модифікація RNN, з'явилася лише в 2014 році. Вони використовуються для синтезу емоційної та автентично звучаючої мови. Наприклад, у тестах сервісів Google Duplex і Microsoft XiaoIce люди не могли відрізнити розмовних роботів від співрозмовників у реальному часі. Варто зазначити, що Xiaoice дав Microsoft міцну позицію на азіатському ринку, який був обмежений мовними бар'єрами.

– Глибока нейронна мережа (DNN) – будь-яка мережа з більш ніж трьома шарами. Вони є основою глибоких механізмів машинного навчання і можуть виявляти приховані зв'язки між різними даними. Яскравим прикладом є використання IBM Watson для пошуку кореляції між розвитком захворювання та різними основними факторами у великій кількості наукових статей.

-Створення конкурентоспроможної мережі (GAN). Це комбінація нейронних мереж, одна з яких генерує варіанти, а інша відсіває їх (виступає в якості арбітра). Таке поєднання дозволяє реалізувати машинне навчання без учителя, що підвищує автономність/ Наприклад, PixelDTGAN генерує окремі зображення одягу, взуття та аксесуарів для каталогів онлайн-магазинів. В якості вхідних даних використовуються фотографії, на яких ці предмети гардероба демонструють фотомоделі. Поки зйомка для каталогів одягу вважається досить витратною частиною електронної комерції, але цілком можливо, що в найближчі роки нейромережі дозволять швидко проілюструвати каталог, навіть не залучаючи фотографа. Обробка фотографій теж забирає багато часу. Ви можете спробувати зробити це набагато швидше, додаючи і прибираючи графічні об'єкти за допомогою іншої нейромережі - IBM GANPaint. Подібна їй нейросеть DRAGAN вже застосовується для автоматичної відтворення персонажів аніме і мультфільмів. Вона дозволяє прискорити вихід нових серій і утримати аудиторію розважальних каналів, не перевантажуючи аніматорів колосальним об'ємом роботи. Так що там

мультфільми! GAN дозволяють анімувати тривимірну модель людини, переносячи на неї рух в реальному часі. Перші результати виглядають не дуже переконливо, однак особливість будь нейромережі - в тому, що вона покращується з кожною новою порцією даних.

1.4 Парадигми навчання нейронних мереж

Існує дві парадигми навчання нейронних мереж - з учителем і без вчителя. У першому випадку, на вхідний вектор є готова відповідь, у другому випадку нейронна мережа самонавчається. У кожного виду навчання є своя ніша завдань і за великим рахунком вони не перетинаються. На даний момент придумано і запатентовано велика кількість архітектур нейронних мереж і методів їх навчання. Але основними (вихідними) є - для навчання з учителем це «алгоритм зворотного поширення помилки», а для навчання без учителя це алгоритми Хебба і Кохонена. Ці парадигми сильно перетинаються з біологічної дійсністю, наприклад - дитина навчається з учителем або без?

Також, останнім часом, сформувалася нова, третя парадигма - навчання з підкріпленням.

1.4.1 Навчання з учителем

Навчання з учителем — це процес налаштування функції машинного навчання, яка відображає вхідні дані з вихідних даних на основі застосування пар вхідних і вихідних даних. Він виконує важливі функції первинного пожертвування та зберігає набір початкових вкладень. У навігаційній парі з візуальною стиковкою шкіри вона зберігається вхідними об'єктами (зазвичай векторами) і вхідними значеннями башани (також званими керуючими сигналами). Алгоритми, що використовуються для створення візуального аналізу, який можна використовувати для відображення вихідних даних і функцій нових програм. Найкраще рішення – дозволити алгоритму правильно призначати імена класів невидимим екземплярам. Цей процес заснований на

алгоритмі, який створює невидимі дані «розумним» способом (частково. Індуктивні очікування). Якість алгоритму розраховується за допомогою так званої статистики пропаганди помилування.

Паралельне завдання в психології людини і тварин часто називають поняттям навчання.

Щоб вирішити проблему навчання менеджменту, необхідно виконати наступні дії:

- Визначити тип тематичного дослідження. Перш ніж робити щось інше, користувач повинен вирішити, які дані використовувати як навчальний набір;
- Наприклад, у разі аналізу рукописного тексту це може бути один рукописний символ, ціле рукописне слово, ціле рукописне речення або це може бути цілий рукописний абзац;
- Зберіть тренувальний набір. Навчальний набір повинен представляти фактичне використання функції. Таким чином збирається набір вхідних об’єктів, а відповідні вихідні дані збираються від експертів або вимірювань.
- Визначте представлення вхідних ознак досліджуваної функції. Точність досліджуваної функції значною мірою залежить від представлення вхідного об'єкта. Зазвичай вхідний об’єкт перетворюється на вектор ознак, який містить багато ознак, що описують об’єкт. Кількість ознак не повинна бути занадто великою через прокляття розмірності; але має містити достатньо інформації для точного прогнозування результату;
- Визначте структуру вивченої функції та відповідний алгоритм навчання. Наприклад, інженер може вибрати використання машин опорних векторів або дерев рішень;
- Завершіть дизайн. Запустіть алгоритм навчання на зібраному навчальному наборі. Деякі алгоритми навчання під керуванням вимагають від користувача визначення певних параметрів керування. Ці параметри можна налаштувати шляхом оптимізації продуктивності підмножини (так званої

набором перевірки) навчального набору або за допомогою перехресної перевірки;

- Оцініть точність вивченої функції. Після налаштування параметрів і навчання продуктивність отриманої функції слід виміряти на тестовому наборі, який є окремо від навчального набору.

Доступний широкий спектр алгоритмів навчання з наглядом, кожен з яких має свої сильні та слабкі сторони. Не існує єдиного алгоритму навчання, який найкраще працює з усіма проблемами навчання під керівництвом.

Є чотири основні проблеми, які слід враховувати під час навчання з наглядом:

Компроміс зміщення-дисперсія.

Перша проблема — це компроміс між упередженням і дисперсією. Уявіть собі, що у нас є кілька різних, але однаково хороших наборів навчальних даних. Алгоритм навчання є упередженим для певного входу x, якщо під час навчання на кожному з цих наборів даних він систематично неправильний під час прогнозування правильного результату для x. Алгоритм навчання має високу дисперсію для певного входу x, якщо він передбачає різні вихідні значення під час навчання на різних навчальних наборах. Похибка передбачення вивченого класифікатора пов'язана із сумою зміщення та дисперсії алгоритму навчання. Як правило, існує компроміс між упередженням і дисперсією. Алгоритм навчання з низьким зміщенням повинен бути «гнучким», щоб він міг добре підходити до даних. Але якщо алгоритм навчання занадто гнучкий, він буде відповідати кожному набору навчальних даних по-різному, а отже, матиме високу дисперсію. Ключовим аспектом багатьох методів навчання з наглядом є те, що вони можуть регулювати цей компроміс між упередженням і дисперсією (автоматично або шляхом надання параметра зміщення/дисперсії, який може налаштувати користувач).

Складність функції та обсяг навчальних даних.

Друге питання стосується кількості навчальних даних, доступних для складності «реальної» функції (класифікатор або функція регресії). Якщо реальна функція проста, то «жорсткий» алгоритм навчання з високим зміщенням і низькою дисперсією можна вивчати з невеликої кількості даних. Але якщо реальна функція дуже складна (наприклад, тому що вона включає складні взаємодії між багатьма різними вхідними функціями і поводиться по-різному в різних частинах вхідного простору), то функція може навчатися лише з дуже великої кількості навчальних даних. І використовуйте «гнучкий» алгоритм навчання з низьким зміщенням і високою дисперсією. Існує чітка різниця між введенням і бажаним результатом.

Розмір вхідного простору.

Третє питання – розмір вхідного простору. Якщо розмірність вхідного вектора ознак дуже велика, Проблема навчання може бути важкою, навіть якщо реальна функція залежить лише від невеликої частини цих характеристик. Це пояснюється тим, що багато «додаткових» вимірів можуть заплутати алгоритм навчання та спричинити його високу дисперсію. Тому високі вхідні розміри зазвичай вимагають налаштувань класифікатора з низькою дисперсією та високим зміщенням. На практиці, якщо інженери можуть вручну видалити невідповідні функції з вхідних даних, це, ймовірно, підвищить точність функції навчання. Крім того, існує багато алгоритмів вибору функцій, які намагаються визначити релевантні функції та відкинути нерелевантні функції. Це приклад більш загальної стратегії зменшення розмірності, яка намагається відобразити вхідні дані в простір нижньої розмірності перед запуском алгоритму керованого навчання.

Шум у вихідному значенні.

Четверта проблема - це ступінь шуму в очікуваному вихідному значенні (контрольна цільова змінна). Якщо бажані вихідні значення часто є неправильними (через помилки людини або датчиків), то алгоритм навчання не повинен намагатися знайти функцію, яка точно відповідає навчальним прикладам. Занадто обережні спроби підігнати дані призводять до

переобладнання. Ви можете переобладнати, навіть якщо немає помилок вимірювання (стохастичний шум), якщо функція, яку ви намагаєтеся вивчити, занадто складна для вашої моделі навчання. У такій ситуації частина цільової функції, яку неможливо змоделювати, «псує» ваші навчальні дані – це явище отримало назву детермінованого шуму. Якщо присутній будь-який тип шуму, краще використовувати більш високі зміщення та меншу оцінку дисперсії.

На практиці існує кілька підходів до зменшення шуму у вихідних значеннях, таких як рання зупинка, щоб запобігти переобладнанню, а також виявлення та видалення галасливих прикладів навчання перед навчанням алгоритму навчання під керівництвом. Існує кілька алгоритмів, які ідентифікують зашумлені навчальні приклади, і видалення передбачуваних прикладів зашумливого навчання перед навчанням зменшило помилку узагальнення зі статистичною значущістю.

Інші фактори, які слід враховувати при виборі та застосуванні алгоритму навчання, включають наступне:

- Неоднорідність даних. Якщо вектор ознак містить багато різних типів ознак (дискретні, дискретно впорядковані, обчислені, безперервні значення), деякі алгоритми простіше застосувати, ніж інші. Багато алгоритмів, у тому числі еталонні векторні машини, лінійна регресія, логістична регресія, нейронні мережі та методи найближчого сусіда, вимагають, щоб вхідні характеристики були числовими та розширюваними до подібного діапазону (наприклад, інтервал [-1,1]). Методи, які використовують функції відстані, такі як метод найближчого сусіда та опорні векторні машини з гаусовими ядрами, особливо чутливі до цього. Перевага дерев рішень полягає в тому, що вони можуть легко обробляти неоднорідні дані.

- Надлишковість даних. Якщо вхідний об’єкт містить зайву інформацію (наприклад, дуже корельовані ознаки), деякі алгоритми навчання (наприклад, лінійна регресія, логістична регресія та методи, засновані на відстані) не працюватимуть добре через чисельну нестабільність. Ці проблеми часто можна вирішити шляхом накладення певної форми регуляризації.

- Наявність взаємодій і нелінійності. Якщо кожна з функцій робить незалежний внесок у вихідні дані, то алгоритми, засновані на лінійних функціях (наприклад, лінійна регресія, логістична регресія, машини опорних векторів, наївні байєсівські машини) і функції відстані (наприклад, методи найближчого сусіда, машини опорних векторів). з ядрами Гаусса), як правило, добре працюють. Однак, якщо між функціями є складні взаємодії, то такі алгоритми, як дерева рішень і нейронні мережі, працюють краще, оскільки вони спеціально розроблені для виявлення цих взаємодій. Лінійні методи також можуть бути застосовані, але інженер повинен вручну вказати взаємодії при їх використанні.

Розглядаючи нову програму, інженер може порівняти кілька алгоритмів навчання та експериментально визначити, який з них найкраще працює для вирішення проблеми (див. перехресну перевірку). Налаштування продуктивності алгоритму навчання може зайняти дуже багато часу. Враховуючи фіксовані ресурси, часто краще витратити більше часу на збір додаткових навчальних даних і більш інформативних функцій, ніж витрачати додатковий час на настройку алгоритмів навчання.

Алгоритми:

Найбільш широко використовувані алгоритми навчання:

- Опорно-векторні машини;
- Лінійна регресія;
- Логістична регресія;
- Наївний Байєс;
- Лінійний дискримінантний аналіз;
- Дерева рішень;
- Алгоритм K-найближчого сусіда;
- Нейронні мережі (багатошаровий персептрон);
- Навчання подібності.

Підходи та алгоритми

- Аналітичне навчання;

- Штучна нейронна мережа;
- Зворотне поширення;
- Підвищення (мета-алгоритм);
- Байєсівська статистика;
- Міркування на основі випадків;
- Навчання дерева рішень;
- Індуктивне логічне програмування;
- Регресія процесу Гаусса;
- Генетичне програмування;
- Груповий метод обробки даних;
- Оцінки ядра;
- Навчальні автомати;
- Навчання систем класифікації;
- Мінімальна довжина повідомлення (дерева рішень, графіки рішень тощо);
- Мультилінійне підпросторове навчання;
- Наївний байєсівський класифікатор;
- Класифікатор максимальної ентропії;
- Умовне випадкове поле;
- Алгоритм найближчого сусіда;
- Ймовірно, приблизно правильне навчання (PAC);
- Правила Ripple down, методологія отримання знань;
- Алгоритми символічного машинного навчання;
- Підсимволічні алгоритми машинного навчання;
- Опорно-векторні машини;
- Машини мінімальної складності (MCM);
- Випадкові ліси;
- Ансамблі класифікаторів;
- Порядкова класифікація;

- Попередня обробка даних;
- Обробка незбалансованих наборів даних;
- Статистичне реляційне навчання;
- Proaftn, багатокритеріальний алгоритм класифікації.

1.4.2 Навчання без учителя

Неконтрольоване навчання або навчання без учителя — це тип машинного навчання, в якому алгоритм не має жодних попередньо призначених міток або оцінок для даних навчання. В результаті алгоритми навчання без нагляду повинні спочатку самостійно виявити будь-які закономірності, що виникають у цьому наборі навчальних даних. Поширені приклади включають кластеризацію, коли алгоритм автоматично групує свої навчальні приклади в категорії зі схожими функціями, і аналіз головних компонентів, де алгоритм знаходить способи стиснути навчальний набір, визначаючи, які функції є найбільш корисними для розрізнення різних навчальних прикладів, і відкидаючи решту. Це контрастує з навчанням під керівництвом, у якому навчальні дані містять заздалегідь призначені мітки категорій (часто людиною або з результатів ненавчального алгоритму класифікації) [19]. Інші проміжні рівні в спектрі нагляду включають навчання з підкріпленням, де для кожного прикладу навчання доступні лише числові бали замість детальних тегів, і навчання з напівнаглядом, де позначено лише частину навчальних даних.

Переваги навчання без нагляду включають мінімальне навантаження на підготовку та перевірку навчального набору, на відміну від методів навчання під керівництвом, де потрібна значна кількість експертної людської праці для призначення та перевірки початкових тегів, а також більша свобода виявлення та використання раніше невиявлених шаблонів. що, можливо, не помітили «експерти». Це часто відбувається за ціною неконтрольованих методів, які вимагають більшого обсягу навчальних даних і повільніше наближаються до прийнятної продуктивності, підвищених вимог до обчислень і зберігання під

час дослідницького процесу та потенційно більшої сприйнятливості до артефактів або аномалій у навчальних даних, які можуть бути очевидними. нерелевантні або визнані людиною помилковими, але їм надається надмірне значення алгоритмом навчання без нагляду.

Поширені сімейства алгоритмів, які використовуються в навчанні без нагляду, включають: (1) кластеризацію, (2) виявлення аномалій, (3) нейронні мережі (зауважте, що не всі нейронні мережі є безконтрольними; їх можна навчати керованими, неконтрольованими, напівкерованими або методи підкріплення) та (4) моделі прихованих змінних.

- Методи кластеризації включають ієрархічну кластеризацію, k-середніх, моделі сумішей, DBSCAN [20] та алгоритм OPTICS;
- Методи виявлення аномалій включають локальний фактор викидів і ізоляційний ліс;
- Підходи до вивчення моделей латентної змінної включають алгоритм очікування-максимізації, метод моментів і методи сліпого поділу сигналів (аналіз основних компонентів, аналіз незалежних компонентів, невід'ємна матрична факторизація, розкладання сингулярних значень);
- Методи нейронних мереж включають автокодери, мережі глибоких переконань, Hebbian навчання, генеративні змагальні мережі (GAN) і самоорганізуючі карти.

Метод моментів. Одним із статистичних підходів до навчання без нагляду є метод моментів. У методі моментів невідомі параметри, що цікавлять модель, пов'язані з моментами однієї або кількох випадкових величин. Ці моменти емпірично оцінюються з доступних вибірок даних і використовуються для розрахунку найбільш вірогідних розподілів значень для кожного параметра. Показано, що метод моментів є ефективним при вивченні параметрів моделей прихованих змінних, де крім спостережуваних змінних, наявних у навчальних і вхідних наборах даних, передбачається також існування ряду неспостережуваних прихованих змінних, які визначають категоризацію. кожного ж. Одним з практичних прикладів моделей

прихованих змінних у машинному навчанні є тематичне моделювання, яке є статистичною моделлю для передбачення слів (спостережуваних змінних) у документі на основі теми (латентної змінної) документа. Було показано, що метод моментів (техніка тензорної декомпозиції) послідовно відновлює параметри великого класу моделей прихованих змінних за певних припущень.

Алгоритм очікування-максимізації є ще одним практичним методом вивчення моделей прихованих змінних. Однак він може застрягти в локальному оптимумі, і він не гарантовано збіжиться до справжніх невідомих параметрів моделі. Навпаки, за допомогою методу моментів глобальна конвергенція гарантується за деяких умов.

Навчання.Під час фази навчання мережа без нагляду намагається імітувати дані, які їй надають, і використовує помилку в імітованих результатах, щоб виправити себе (тобто виправити свої ваги та упередження). Це нагадує мімікрію дітей, коли вони вивчають мову. Іноді помилка виражається як низька ймовірність того, що відбувається помилковий вихід, або вона може бути виражена як нестабільний високоенергетичний стан в мережі.

На відміну від основного використання зворотного поширення в контрольованому методі, неконтрольовані методи використовують різні алгоритми навчання, включаючи: правило навчання Хопфілда, правило навчання Больцмана, контрастна дивергенція, сон пробудження, варіаційний висновок, максимальне апостеріорне, вибірка Гіббса, зворотне поширення помилки реконструкції або зворотне поширення. перепараметрізації прихованого стану.

1.5 Розпізнавання зображень

Розпізнавання шаблонів – це автоматичне розпізнавання закономірностей у даних. Воно має застосування в статистичному аналізі даних, обробці сигналів, аналізі зображень, пошуку інформації,

біоінформатиці, стисканні даних, комп’ютерній графіці та машинному навчанні. Розпізнавання образів бере початок у статистиці та інженерії; деякі сучасні підходи до розпізнавання образів включають використання машинного навчання через збільшення доступності великих даних і нового великого обсягу обчислювальної потужності. Однак ці види діяльності можна розглядати як два аспекти однієї сфери застосування, і разом вони зазнали значного розвитку за останні кілька десятиліть.

Системи розпізнавання шаблонів у багатьох випадках навчаються на основі позначених «навчальних» даних, але коли мічені дані недоступні, можна використовувати інші алгоритми для виявлення раніше невідомих шаблонів. KDD та інтелектуальний аналіз даних зосереджені на неконтрольованих методах та більш міцному зв’язку з використанням у бізнесі. Розпізнавання образів більше зосереджується на сигналі, а також бере до уваги отримання та обробку сигналу. Він виник у інженерії, і цей термін популярний у контексті комп’ютерного зору: провідна конференція з комп’ютерного зору називається Conference on Computer Vision and Pattern Recognition.

У машинному навчанні розпізнавання образів — це присвоєння мітки заданому вхідному значенню. У статистиці дискримінантний аналіз був введений з цією ж метою в 1936 році. Прикладом розпізнавання шаблонів є класифікація, яка намагається призначити кожне вхідне значення одному з заданого набору класів (наприклад, визначити, чи є даний електронний лист "спамом"). або "не спам"). Однак розпізнавання образів є більш загальною проблемою, яка охоплює також інші типи виводу. Іншими прикладами є регресія, яка призначає вихідні дані з дійсним значенням кожному входу; позначення послідовності, яке призначає клас кожному члену послідовності значень (наприклад, тегування частини мови, яке призначає частину мови кожному слову у вхідному реченні); і синтаксичний розбір, який призначає дерево розбору вхідному реченню, описуючи синтаксичну структуру речення.

Алгоритми розпізнавання шаблонів, як правило, мають на меті надати розумну відповідь для всіх можливих вхідних даних і виконувати «найімовірніше» узгодження вхідних даних, беручи до уваги їх статистичні варіації. Це протилежно алгоритмам узгодження шаблонів, які шукають точні збіги у вхідних даних з уже існуючими шаблонами. Поширеним прикладом алгоритму відповідності шаблону є відповідність регулярних виразів, яка шукає шаблони заданого типу в текстових даних і включена в пошукові можливості багатьох текстових редакторів і текстових процесорів.

1.6 Висновок

У даному розділі розглянуто основні види аналізу зображень в соціальних мережах. Для кожного виду аналізу наведено перелік необхідних даних, короткий огляд основних методів, які використовуються для проведення аналізу. Також описані показники, які можна отримати після аналізу, їх інформативність та використання в подальшому аналізі або представлені результатів аналізу. Наведене перелік існуючих програмних засобів, описано їх переваги та недоліки, констатуємо, що на ринку представлені переважно системи, які вимагають великих ресурсів для навчання, і тому можуть використовуватися великими компаніями. Рішення, які можуть використовуватися невеликими компаніями, показують не дуже високу точність розпізнавання. Також виявлено залежність якості розпізнавання від освітлення, позиціонування тощо. Також в даному розділі наведено актуальність впровадження даної інформаційної системи, та доведено що розробко такої системи є актуальною задачею

2 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ФІЛЬТРАЦІЇ ЗОБРАЖЕНЬ

2.1 Вибір мови програмування

Python - інтерпретована мова програмування загального призначення високого рівня. Філософія дизайну Python підкреслює читабельність коду, використовуючи значні відступи. Його мовна структура та об’єктно-орієнтований підхід покликані допомогти програмістам писати чіткі логічні коди для малих та великих проектів.

Python динамічно збирає сміття. Він підтримує кілька парадигм програмування, включаючи структуроване, процедурне, об'єктно-орієнтоване та функціональне програмування.

Як спадкоємець мови програмування ABC, Гідо ван Россум почав займатись дослідженнями Python наприкінці 1980-х і вперше випустив цю мову під назвою Python 0.9.0 в 1991 році [21].

Python 2.0 був випущений 16 жовтня 2000 року з багатьма основними новими можливостями, включаючи збирач сміття для виявлення циклів та підтримку Unicode.

Python 3.0 був випущений 3 грудня 2008 року. Це серйозний перегляд мови, який не повністю сумісний з іншою стороною. Багато його основних функцій було перенесено до версій Python 2.6.x та 2.7.x. До розподілу Python 3 входить утиліта 2to3, яка автоматично (принаймні частково) перетворює код Python 2 у Python 3.

Термін дії Python 2.7 спочатку був встановлений у 2015 році, а потім перенесений на 2020 рік через побоювання, що велика кількість існуючого коду не може бути легко перенесена на Python 3. Більше виправлень безпеки та інших удосконалень не буде випущено. Наприкінці життєвого циклу Python 2 підтримує лише Python 3.6.x та новіші версії [22].

Оскільки всі версії Python мають проблеми з безпекою, швидкість Python 3.9.2 та 3.8.8 була прискорена, що призвело до можливого віддаленого виконання коду та отруєння веб-кешу.

Python - мова програмування з багатьма парадигмами. Він повністю підтримує об'єктно-орієнтоване програмування та структуроване програмування, а багато його функцій підтримують функціональне програмування та аспектно-орієнтоване програмування (включаючи дизайн метапрограми та мета-об'єкт (магічний метод)). Розширення підтримує багато інших парадигм, включаючи дизайн контрактів та логічне програмування.

Python використовує динамічне введення тексту та комбінацію підрахунку посилань та збирачів сміття.Сбирач сміття визначає цикл для управління пам'яттю. Він також має динамічну роздільну здатність імен (пізнє прив'язування), щоб асоціювати імена методів та змінних під час виконання програми.

Python призначений для надання певної підтримки функціонального програмування на основі Lisp. За допомогою функцій фільтрування, відображення та скорочення; перелічено розуміння генераторів, словників, наборів та виразів. Стандартна бібліотека має два модулі (itertools та functools) для реалізації функціональних інструментів, запозичених у Haskell та Standard ML.

Основні принципи мови викладені в документі Zen of Python (PEP 20), який включає такі сентенції:

- Красиво-краще, ніж потворно;
- Явне краще, ніж неявне;
- Просте - краще, ніж складне;
- Комплекс краще, ніж комплекс;
- Розрахунок читабельності.

Python не має всіх своїх функцій, вбудованих у своє ядро, але розроблений таким чином, щоб бути дуже розширюваним (з модулями). Ця компактна модульність робить його особливо популярним як спосіб додавання

програмованих інтерфейсів до існуючих програм. Бачення Ван Россума щодо малих основних мов, великих стандартних бібліотек та легко розширюваних перекладачів випливає з його невдоволення ABC, який підтримує протилежний підхід. Python прагне спростити та зменшити плутанину граматики та граматики, дозволяючи розробникам обирати методи кодування. На відміну від девізу Perl "Існує кілька способів зробити це", Python висвітлює філософію дизайну "Для цього повинен бути один - бажано, лише один очевидний спосіб" [23]. Алекс Мартеллі, співробітник Фонду програмного забезпечення Python і автор книги про Python, писав: "Опис чогось як" розумного "не вважається компліментом до культури Python".

Розробники Python прагнуть уникнути передчасної оптимізації та відмовляються виправляти некритичні частини посилальної реалізації CPython. Ці виправлення відбуваються за рахунок ясності, тим самим трохи покращуючи швидкість. Коли швидкість важлива, програмісти Python можуть переміщати критично важливі для часу функції до розширень, написаних мовами, такими як C, або використовувати компілятор PyPy, що встигає за часом. Також доступний Cython, який перетворює сценарії Python на C і здійснює виклики API рівня C безпосередньо до інтерпретатора Python.

Важливою метою розробників Python є збереження задоволення від використання. Це відображається в назві мови (данина поваги британській комедійній компанії Monty Python), а іноді і в цікавих підручниках та довідкових методах, таких як стандартний foo та bars.

Користувачів та шанувальників Python, особливо тих, хто вважається обізнаним або досвідченим, часто називають Pythonistas. Python має бути простою для читання мовою. Його формат візуально не бентежить, він часто використовує англійські ключові слова, тоді як інші мови використовують розділові знаки. На відміну від багатьох інших мов, він не використовує фігурні дужки для розділення блоків і допускає крапку з комою після операторів, але крапка з комою використовується рідко. Він має менше граматичних винятків та особливих випадків, ніж C або Pascal.

Python використовує пробіли замість фігурних дужок або ключових слів для розділення блоків. Збільшення відступу відбувається після певних операторів; зменшення відступу означає кінець поточного блоку. Тому візуальна структура програми точно відображає семантичну структуру програми. Цю особливість іноді називають “зовнішнім” правилом, і деякі інші мови також мають це правило, але в більшості мов відступ не має семантичного значення. Рекомендований розмір відступу - чотири пробіли.

Оператори Python включають (серед іншого):

- Використовуйте знак рівності = оператор присвоєння;
- Оператор if, який використовується разом з else та elif (скорочення від else-if) для умовного виконання блоку коду;
- Оператор for, який фіксує кожен елемент у локальній змінній для використання вкладеним блоком для перегляду об'єкта, що ітерацію;
- Поки оператор, доки умова є істинним, виконується блок коду;
- Оператор try, який дозволяє захоплювати та обробляти витяги, що містяться у вкладених блоках коду (крім речень); він також гарантує, що незалежно від того, як блок виходить, код очищення в блоці завжди буде виконаний в кінцевому підсумку;
- Оператор підвищення застосовується для отримання зазначеного винятку або повторного підняття спійманого винятку;
- Оператори класів, які виконують блоки коду та приєднують свій локальний простір імен до класу для використання в об'єктно-орієнтованому програмуванні;
- Оператор Def, який визначає функцію або метод;
- Оператор with з Python 2.5, випущений у вересні 2006 р. [24], який охоплює блок коду в диспетчері контексту (наприклад, отримання блокування перед запуском кодового блоку, а потім звільнення блокування або відкриття файлу. Потім закрийте його), дозволяють поведінку подібних ресурсів ініціалізувати (RAII) та замінюють загальне використання try / final;
- Оператор Break, вихід із циклу;

- Оператор continue пропускає цю ітерацію та переходить до наступного пункту;
- Оператор del видалить змінну, що означає, що посилання на ім'я значення буде видалено, а спроби використання цієї змінної призведуть до помилки. Ви можете перепризначити видалені змінні;
- Виконувати функції NOP через операторів. Синтаксично це необхідно для створення порожнього блоку коду;
- Заява про затвердження, що використовується для перевірки умов, що застосовуються під час налагодження;
- Оператор Yield, який повертає значення з функції генератора. У Python 2.5 yield також є оператором. Ця форма використовується для реалізації спільного плану;
- Оператор return використовується для повернення значень із функцій;
- Оператор import використовується для імпорту модулів, функції чи змінні яких можна використовувати в поточній програмі. Існує три способи використання імпорту: імпорт <ім'я модуля> [як <псевдонім>] або <ім'я модуля> імпорт * або <ім'я модуля> для імпорту <визначення 1> [як <псевдонім 1>], <визначення 2 > [Як <псевдонім 2>].

Роль оператора присвоєння (=) полягає у прив'язуванні імені як посилання на окремий динамічно розподілений об'єкт. Потім ви можете будь-коли відхилити змінну до будь-якого об’єкта. У Python ім'я змінної є спільним власником посилання, і з ним не пов’язаний фіксований тип даних. Однак у цей час змінна посилається на об’єкт із типом. Це називається динамічним набором тексту і на відміну від статично набраних мов програмування, де кожна змінна може містити лише певний тип значення.

Python не підтримує оптимізацію хвостових викликів або першокласні розширення, і за словами Гвідо ван Россума, це ніколи не буде. Однак краща підтримка функцій, подібних до корутину, передбачена в 2.5, розширюючи тим самим генератор Python. Ледачий ітератор має максимум 2,5 генератора. Інформація передається в одному напрямку від генератора. За допомогою

Python 2.5 ви можете передавати інформацію назад до функції генератора, тоді як за допомогою Python 3.3 ви можете передавати інформацію через кілька рівнів стека.

Деякі вирази Python подібні до таких, що зустрічаються в таких мовах, як C та Java, а інші - ні:

- Додавання, віднімання та множення однакові, але поведінка ділення інша. У Python є два типи розділів. Вони є базовим діленням (або цілочисельним поділом) // та плаваючою комою / діленням. Python також використовує оператори ** для просування;
- У Python 3.5 було введено нове твердження @infix. Він використовується бібліотеками, такими як NumPy, для множення матриць;
- У Python 3.8 введено синтаксис: =, який називається "моржовим оператором". Він присвоює значення змінним як частину більшого виразу [91];
- У Python == порівнює з Java за значенням, Java порівнює числові значення за значенням, а об'єкти порівнює за посиланням. (Порівняння значень об'єктів у Java може бути здійснено за допомогою методу equals ().) Оператор is може бути використаний для порівняння ідентифікаторів об'єктів (порівняння посилань). У Python можна порівняти порівняння, наприклад <= b <= c;
- Python використовує слово і, або не для своїх булевих операторів, а не для символів &&, ||! Використовується в Java та C.Python має тип виразу, що називається розумінням списку, а також більш загальний вираз, який називається генераторським виразом;
- Анонімні функції реалізовані з використанням лямбда-виразів, однак вони обмежені тим фактом, що тіло може бути лише виразом;
- Якщо с інакше у, умовний вираз у Python буде записаний як х (відмінний від порядку операнда оператора с? Х: у), що є однаковим у багатьох інших мовах;

- Python розрізняє списки та кортежі. Список записаний як [1, 2, 3], він був змінений і не може використовуватися як ключ словника (ключ словника повинен бути постійним у Python). Кортеж записується як (1, 2, 3) і є незмінним, тому, поки всі елементи кортежу є незмінними, він може використовуватися як ключ словника. Оператор + може використовуватися для об'єднання двох кортежів, що безпосередньо не змінює їх вміст, але створює новий кортеж, що містить елементи двох передбачених кортежів. Отже, враховуючи, що змінна t спочатку дорівнює (1,2,3), коли виконується t = t + (4, 5), спочатку обчисліть t + (4, 5), щоб отримати (1,2, 3) . 3, 4, 5), а потім призначити його назад до t, тим самим ефективно "змінюючи вміст t", одночасно відповідаючи інваріантній природі об'єкта кортежу. У чіткому контексті кортежі не потребують дужок;

- Python має функцію декомпресії послідовності, де кілька виразів (кожен вираз обчислює весь вміст, який може бути призначений) (змінні, атрибути запису тощо) мають однакову форму з текстом, що становить кортеж, і, як правило, розміщуються зліва Одна сторона знака рівності у твердженні про присвоєння. Оператор очікує ітерабельний об’єкт праворуч від знака рівності. При сортуванні об’єкт видасть таку ж кількість значень, що й даний вираз, і відфільтрує їх, присвоюючи кожне створене значення лівій стороні відповідного виразу;

- Python має оператор% "рядковий формат". Це схоже на рядок printf у C, наприклад, "спам =% s яйце =% d"% ("бла", 2) оцінюється як "спам = бла-яйця = 2". У Python 3 та 2.6+ метод str () формату () доповнив це, наприклад "спам = {0} яйце = {1}". Формат ("бла", 2). Python 3.6 додав "f-рядки": blah = "blah"; яйця = 2; f'spam = {blah} яйця = {яйця};

- Рядки в Python можна поєднувати шляхом "додавання" (те саме твердження, що і додавання цілих чи значень з плаваючою комою). Наприклад, "спам" + "яйце" повертає "спамера". Навіть якщо ваш рядок містить цифри, вони все одно будуть додані як рядки замість цілих чисел. Наприклад, "2" + "2" повертає "22";

- Python має різні типи рядкових літералів:

– Рядки, укладені в одинарні або подвійні лапки. На відміну від Unix Shell, мови Perl та Perl впливають на Perl, а одинарні та подвійні лапки працюють однаково. Обидва типи рядків використовують зворотні скісні риски (\) як вихідні символи. Інтерполяція рядків стала "відформатованим рядковим літералом" у Python 3.6.Рядки з подвійними лапками, які починаються та закінчуються серією з трьох одинарних або подвійних лапок. Вони можуть охоплювати кілька рядків і функціонувати, як тут документи в оболонках, Perl і Ruby;

– Початковий рядок, представлений префіксом рядкового літералу перед r. Ескейп-послідовності не інтерпретуються; тому необроблені рядки корисні в місцях, де поширені зворотні скісні риски літер, наприклад, регулярні вирази та шляхи у стилі Windows. Порівняйте "@ -citation" у C #;

- Python має індекси масивів та вирази масивів у списках, позначених як [ключ], [старт: зупинка] або [старт: зупинка: крок]. Індекс базується на нулі та є від'ємним відносно кінця. Нарізання переміщує елементи від початкового індексу до (але не враховуючи) індексу зупинки. Третій варіант зрізу називається кроком або висотою, що дозволяє пропускати та обертати елементи. Індекс зрізу можна опустити, наприклад [:] повертає копію всього списку. Кожен елемент зрізу є неглибокою копією.

Python чітко розмежовує вирази та висловлювання, що відрізняється від таких мов, як Common Lisp, Scheme або Ruby. Це призводить до дублювання певних функцій.

Python зазвичай використовується в проектах зі штучного інтелекту та машинного навчання з такими бібліотеками, як TensorFlow, Keras, Pytorch та Scikit-learn. Як мова сценаріїв з модульною архітектурою, простим синтаксисом та розширеними засобами обробки текстів, Python також використовується для обробки природних мов, він успішно впроваджений у багато програмних продуктів на мовах сценаріїв, включаючи програмне забезпечення з кінцевими елементами, таке як Abaqus, 3D-параметричні

моделятори, такі як FreeCAD, 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, Nuke visual ефекти Composer, програми для двовимірних зображень (такі як GIMP, Inkscape, Scribus та Paint Shop Pro) та програми для приміток (наприклад, автор та група). Налагоджувач GNU використовує Python як хороший принтер для відображення складних структур, таких як контейнери C ++.

Есрі виступає за те, що Python є найкращим вибором для сценаріїв в ArcGIS. Він також використовувався в декількох відеоіграх і був першою з трьох доступних мов програмування, що використовуються Google App Engine, а інші дві використовуються як Java та Go.

Багато операційних систем використовують Python як стандартний компонент. Він поставляється з більшістю дистрибутивів Linux, AmigaOS 4 (з використанням Python 2.7), FreeBSD (як пакет), NetBSD, OpenBSD (як пакет) та macOS, і може використовуватися з командного рядка (терміналу). Багато дистрибутивів Linux використовують інсталятори, написані на Python: Ubuntu використовує інсталятор Ubiquity, тоді як Red Hat Linux та Fedora використовують інсталятор Anaconda. Gentoo Linux використовує Python у своїй системі управління пакетами Portage.

Python широко використовується для захисту інформації, включаючи розробку експлойт-програм.

Більшість програм Sugar Laptop XO, які зараз розробляються Sugar Labs, написані на Python. Проект одноплатної комп’ютерної програми Raspberry Pi прийняв Python як основну мову програмування користувача.

LibreOffice включає Python і має намір замінити Java на Python. Його постачальник скриптів Python є основною функцією у версії 4.0 від 7 лютого 2013 року.

Виходячи з обсягу та потужних особливостей мови Python, описаних вище, можна зробити висновок, що вона універсальна. Ось чому для виконання цієї роботи було обрано саме цю мову програмування.

2.2 Вибір додаткового інструментарію

Scikit-image (раніше scikits.image) – це бібліотека зображень з відкритим кодом для мови програмування Python. Він включає алгоритми сегментації, геометричного перетворення, маніпулювання кольоровим простором, аналізу, фільтрації, морфології, виявлення особливостей тощо. Він призначений для взаємодії з числовою та науковою бібліотеками Python NumPy та SciPy.

Проект scikit-image розпочався на scikits.image, а його автором був Стефан ван дер Вальт. Його назва походить від ідеї, що це "SciKit" (набір інструментів SciPy), розроблений та розповсюджений окремо стороннім розширенням SciPy. Оригінальна база коду пізніше була широко переписана іншими розробниками. У листопаді 2012 року в різних наукових роботах наукові знання та наукові знання були описані як «ретельно модифіковані та популярні». Scikit-image також дуже активний у Google Summer of Code. [25]

scikit-image в основному написаний на Python, а деякі основні алгоритми написані на Cython для підвищення продуктивності.

Keras — це бібліотека програмного забезпечення з відкритим кодом, яка надає інтерфейс Python для штучних нейронних мереж. Keras діє як інтерфейс для бібліотеки TensorFlow.

До версії 2.3 Keras підтримував кілька серверних програм, зокрема TensorFlow, Microsoft Cognitive Toolkit, Theano та PlaidML [26]. Починаючи з версії 2.4, підтримується лише TensorFlow. Розроблений для швидкого експериментування з глибокими нейронними мережами, він зосереджений на зручності, модульності та розширюваності. Її було розроблено в рамках дослідницької роботи проекту ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) [27], а її основним автором і супроводжувачем є Франсуа Шолле, інженер Google. Шолле також є автором моделі глибокої нейронної мережі Xception [28].

Особливості. Keras містить численні реалізації часто використовуваних будівельних блоків нейронних мереж, таких як шари, цілі, функції активації,

оптимізатори та безліч інструментів, які полегшують роботу з зображеннями та текстовими даними, щоб спростити кодування, необхідне для написання глибокого коду нейронної мережі. Код розміщено на GitHub, а форуми підтримки спільноти містять сторінку проблем GitHub і канал Slack.

Окрім стандартних нейронних мереж, Keras підтримує згорточні та рекурентні нейронні мережі. Він підтримує інші загальні рівні корисності, такі як відключення, нормалізація пакетів і об’єднання [29].

Keras дозволяє користувачам створювати глибокі моделі на смартфонах (iOS і Android), в Інтернеті або на віртуальній машині Java [30]. Це також дозволяє використовувати розподілене навчання моделей глибокого навчання на кластерах графічних процесорів (GPU) і тензорних процесорів (TPU) [31].

TensorFlow — це безкоштовна бібліотека програмного забезпечення з відкритим кодом для машинного навчання та штучного інтелекту. Він може використовуватися в ряді завдань, але особливу увагу приділяє навчанню та висновку глибоких нейронних мереж.[32]

TensorFlow розроблено командою Google Brain для внутрішнього використання Google у дослідженнях і виробництві. Початкова версія була випущена під ліцензією Apache 2.0 у 2015 році [33]. Google випустив оновлену версію TensorFlow під назвою TensorFlow 2.0 у вересні 2019 року.[60]

TensorFlow можна використовувати в багатьох мовах програмування, включаючи Python, JavaScript, C++ і Java [61]. Ця гнучкість дає змогу використовувати різноманітні програми в різних секторах.

Автодиференціація. Автодиференціація — це процес автоматичного обчислення вектора градієнта моделі щодо кожного її параметра. За допомогою цієї функції TensorFlow може автоматично обчислювати градієнти для параметрів у моделі, що корисно для таких алгоритмів, як зворотне поширення, яким потрібні градієнти для оптимізації продуктивності [34] Для цього фреймворк повинен відстежувати порядок операцій, які виконуються над вхідними тензорами в моделі, а потім обчислювати градієнти щодо відповідних параметрів [35].

Завзяте виконання. TensorFlow включає режим «нетерплячого виконання», який означає, що операції оцінюються негайно, а не додаються до обчислювального графіка, який виконується пізніше [36]. Код, який виконується з нетерпінням, можна поетапно досліджувати за допомогою налагоджувача, оскільки дані доповнюються в кожному рядку коду, а не пізніше в обчислювальному графіку. Цю парадигму виконання вважають легшою для налагодження через її покрокову прозорість.

Поширювати. TensorFlow надає API для розподілу обчислень між декількома пристроями з різними стратегіями розподілу. Ці розподілені обчислення часто можуть пришвидшити виконання навчання та оцінювання моделей TensorFlow і є звичайною практикою в області ШІ.

Втрати. Для навчання та оцінки моделей TensorFlow надає набір функцій втрат (також відомих як функції вартості) [37]. Деякі популярні приклади включають середню квадратичну помилку (MSE) і двійкову крос-ентропію (BCE). Ці функції втрат обчислюють «помилку» або «різницю» між виходом моделі та очікуваним виходом (ширше, різницею між двома тензорами). Для різних наборів даних і моделей різні втрати використовуються для визначення пріоритетів певних аспектів продуктивності.

Метрики. Щоб оцінити продуктивність моделей машинного навчання, TensorFlow надає API-доступ до загальновживаних показників. Приклади включають різні показники точності (бінарні, категоричні, розріджені категоричні) разом з іншими показниками, такими як точність, відкликання та перехрестя через об’єднання (IoU) [38].

TensorFlow.nn — це модуль для виконання примітивних операцій нейронної мережі над моделями. Деякі з цих операцій включають варіації згорток (1/2/3D, Atrous, по глибині), функції активації (Softmax, RELU, GELU, Sigmoid тощо) та їх варіації, а також інші операції Tensor (max-pooling, bias-add тощо) [39].

Оптимізатори. TensorFlow пропонує набір оптимізаторів для навчання нейронних мереж, включаючи ADAM, ADAGRAD і Stochastic Gradient

Descent (SGD) [40]. Під час навчання моделі різні оптимізатори пропонують різні режими налаштування параметрів, що часто впливає на конвергенцію та продуктивність моделі.

2.3 Висновок

В даному розділі наведено обгрунтування вибору мови реалізації програмного засобу. В результаті чого було обрано мову програмування Python. Також наведено характеристику мови та додаткових іструментів для розробки інформаційно системи фільтрації зображень. Було описано перелік бібліотек необіхдних для розробки та описано їх можливості. аналізу інструментальних засобів розробки системи. Окрім цього було проведено аналітичний огляд бібліотеки Tensorflow, яка буде використовуватися при розробці.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ФІЛЬТРАЦІЇ ЗОБРАЖЕНЬ

3.1 Проектування внутрішньої будови системи

У програмній інженерії діаграма класів уніфікованої мови моделювання (UML) — це статична структурна діаграма, яка описує структуру системи та показує системні класи, їх властивості, операції (або методи) і зв’язки між об’єктами.

Діаграми класів є основними будівельними блоками об'єктно-орієнтованого моделювання. Вони використовуються для загального концептуального моделювання структури програми та детального моделювання для перетворення моделей у програмний код. Діаграми класів також можна використовувати для моделювання даних. Класи на діаграмі класів представляють основні елементи, взаємодію та класи програмування в програмі.

На схемі ці класи представлені вікнами, що містять три відділення:

– у верхньому відділенні міститься назва класу. Надруковано жирним шрифтом і відцентровано, а перша літера написана великими літерами;

– середній відсік містить атрибути класу. Вони вирівняні за лівим краєм, а перша буква мала;

– у нижньому відділенні містяться операції, які може виконувати клас. Вони також вирівняні за лівим краєм, а перша буква – мала.

При проектуванні системи багато класів ідентифікуються і групуються в схему класів, що допомагає визначити статичні відносини між ними. При детальному моделюванні класи концептуального проектування часто поділяють на підкласи.

Залежність — це семантичний зв’язок між залежними елементами моделі та незалежними елементами. Він існує між двома елементами, якщо зміна, визначена одним елементом (сервером або цільовим), може спричинити зміну іншого елемента (клієнта чи джерела). Це об’єднання одностороннє. Залежності показані пунктирними лініями з порожнистими стрілками, що вказують від клієнтів до постачальників.

Для подальшого опису поведінки системи ці діаграми класів можуть бути доповнені діаграмами станів або автоматами станів UML.

Асоціація являє собою серію посилань. Бінарні асоціації (з двома кінцями) зазвичай представляють у вигляді рядків. Асоціація може пов'язувати будь-яку кількість класів. Асоціація з трьома ланками називається потрійною асоціацією. Асоціації можна назвати, а кінці асоціації можна прикрасити іменами ролей, індикаторами власності, множинністю, видимістю та іншими атрибутами.

Є чотири різні типи асоціацій: двонаправлені, односпрямовані, агрегатні (включаючи складені агрегати) і рефлексивні. Найбільш поширеними є двонаправлені та односпрямовані асоціації.

Наприклад, категорія польоту пов’язана з категорією літака з двостороннім рухом. Асоціація являє собою статичні відносини, спільні між об'єктами двох класів.

Агрегації є різновидом відношення «has»; агрегації є більш конкретними, ніж асоціації. Це асоціація, яка представляє частину мети або частину відносин. Як показано, у професора «має» клас викладати. Як тип асоціації, агрегати можуть бути названі й мати те саме прикраси, що й асоціації. Однак агрегат не може включати більше двох класів; це має бути бінарна асоціація. Крім того, у процесі впровадження існує невелика різниця між агрегацією та асоціацією, і графік може повністю опустити зв’язок агрегації. [41]

Агрегація може відбуватися, коли клас є колекцією або контейнером інших класів, але міститься клас не має сильної залежності від часу життя контейнера. Коли контейнер знищено, вміст контейнера все ще існує.

В UML він представлений графічно у вигляді порожнистого ромба на класі хоста, з'єднаного з класом хоста лінією. Агрегат є семантично розширеним об’єктом і вважається одиницею в багатьох операціях, навіть якщо він фізично складається з кількох менших об’єктів.

Приклад: бібліотека та студенти. Тут студенти можуть існувати без бібліотеки, а зв’язок між студентами та бібліотекою є сукупністю.

Це показує, що один із двох споріднених класів (підкласів) вважається особливою формою іншого (суперкласу), а суперклас є узагальненням підкласу. На практиці це означає, що будь-який екземпляр підтипу також є екземпляром суперкласу. Типове дерево узагальнень цієї форми можна знайти в біологічних класифікаціях: людина — підклас людиноподібних мавп, людиноподібні мавпи — підклас ссавців тощо. Найкраще цей зв’язок можна зрозуміти за допомогою фрази «А є Б» (люди — ссавці, ссавці — тварини).

Графічне представлення узагальнення UML — це форма порожнистого трикутника в кінці суперкласу рядка (або дерева рядків), який з’єднує його з одним або кількома підтипами.

Відносини узагальнення також відомі як відносини успадкування або відносини «є».

Суперклас (базовий клас) у відносинах узагальнення також називають «батьківським класом», суперкласом, базовим класом або базовим класом.

Підтип у відносинах спеціалізації також називається "дочірнім" підкласом, похідним класом, похідним типом, успадкованим класом або успадкованим типом.

Зауважте, що ці відносини зовсім не схожі на біологічні відносини між батьком і дитиною: використання цих термінів дуже поширене, але воно може ввести в оману.

А є типом В. Наприклад, «дуб — це дерево», «автомобіль — транспортний засіб»

Узагальнення можуть бути показані лише в діаграмах класів і діаграмах використання.

У моделюванні UML відносини реалізації — це відносини між двома елементами моделі, де один елемент моделі (клієнт) реалізує (реалізує або виконує) поведінку, задану іншим елементом моделі (постачальником).

Графічне представлення реалізації UML — це порожнистий трикутник на кінці штрихового інтерфейсу (або дерева ліній), який з'єднує його з одним або кількома реалізаторами. Проста стрілка використовується в кінці інтерфейсу з пунктирами, щоб підключити його до користувача. У діаграмах компонентів використовується графічна умова «м'яч і розетка» (реалізатор розміщує кульку або льодяник, а користувач відображає розетку).

Реалізації можна показати лише на діаграмах класів або компонентів. Реалізації – це зв'язки між класами, інтерфейсами, компонентами та пакетами, які з'єднують елементи замовника з елементами постачальника. Відношення реалізації між класом/компонентом та інтерфейсом вказує на те, що клас/компонент реалізує операції, запропоновані інтерфейсом.

Залежність — це слабкіша форма спілкування, яка вказує на те, що один клас залежить від іншого класу, оскільки він використовує його в певний момент часу. Клас залежить від іншого класу, якщо незалежний клас є змінною параметра або локальною змінною методу залежного класу. Це відрізняється від асоціації властивостей залежних класів, які є екземплярами незалежних класів. Іноді відносини між двома класами слабкі. Вони взагалі не реалізуються зі змінними-членами. Натомість вони можуть бути реалізовані як аргументи для функцій-членів.

Ці відносини часто описують як «А має Б» (мама з кошенятами, кошенята з мамою).

Представлення асоціації UML - це лінія, що з'єднує два пов'язані класи. Кожен кінець рядка має інші символи. Наприклад, ми можемо

використовувати загострені стрілки для вказівки. Ми можемо представити право власності, помістивши кулю, роль, яку відіграє елемент на цьому кінці, вказавши ім’я ролі та кілька екземплярів цієї сутності (з іншого боку, пов’язуючи область задіяних об’єктів).

Класи сутностей моделюють довгострокову інформацію, яку обробляє система, а іноді і поведінку, пов’язану з цією інформацією. Їх не слід ідентифікувати як таблиці бази даних або інші сховища даних.

Вони намальовані у вигляді кіл з короткою лінією, прикріпленою до нижньої частини кола. Крім того, їх можна намалювати як звичайні класи з «основним» стереотипом у назві класу.

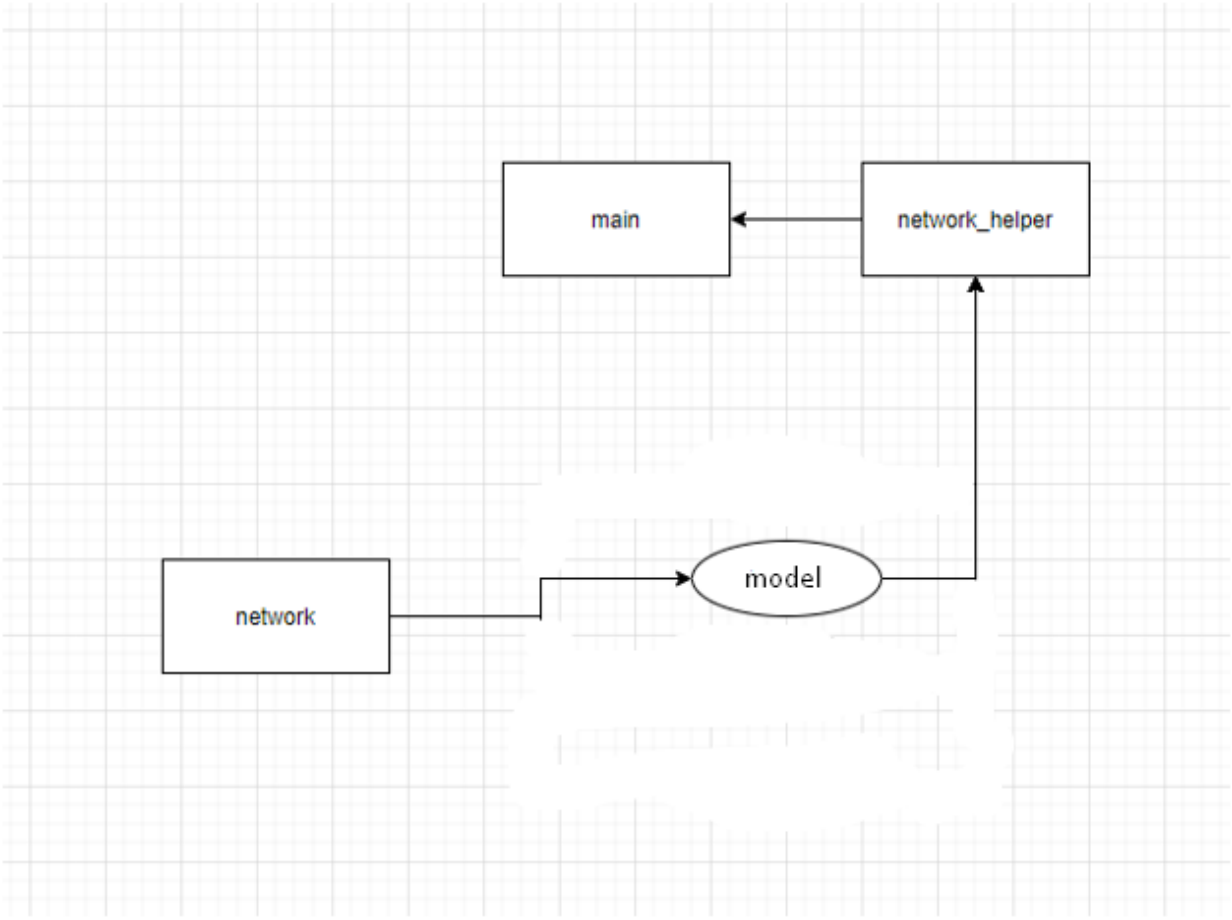


Рисунок 3.1 — Діаграма класів

3.2 Аналіз варіантів діяльності

Найпростіша діаграма використання — це уявлення про взаємодію користувача з системою, яка показує взаємозв’язок між користувачем і різними видами використання, в яких бере участь користувач. Діаграми варіантів використання визначають різні типи користувачів системи та різні варіанти використання, і часто супроводжуються діаграмами інших типів. Мета зображується колом або овалом.

У той час як самі варіанти використання можуть детально досліджувати кожну можливість, діаграми варіантів використання можуть допомогти надати огляд системи більш високого рівня.

Завдяки своїй спрощеній природі сценарії використання можуть бути хорошим інструментом комунікації для зацікавлених сторін. Ці малюнки намагаються імітувати реальний світ, даючи зацікавленим сторонам уявлення про те, як буде розвиватися система. Було проведено дослідження, щоб визначити, чи є реальний випадок використання програми. Було виявлено, що діаграми варіантів використання більш спрощено повідомляють про наміри системи зацікавленим сторонам, і вони були «пояснені більш повно, ніж діаграми класів».

Метою використання діаграм є показати динамічні аспекти системи. Для повного функціонального та технічного представлення системи доступні додаткові схеми та документація. Вони забезпечують спрощене та графічне представлення того, що насправді має робити система.

Опис проект за допомогою діаграми використання:

– рамки системи (англ. system border) - прямокутник із назвою у верхніх частинах та еліпсами (прецедентами) всередині;

– актор (англ. актор) стилізований людський персонаж, описує набір ролей користувача (розуміється в широкому змісті: людина, зовнішня сутність, клас, інша система), взаємодіючого з деякою сутністю (системною,

підсистемою, класом). Актори не можуть бути пов'язані між собою з іншим (за вимкнення відносин щодо обробки / дослідження);

– прецедент – еліпс із надписом, що означає виконувану систематичну дію (може включати можливі варіанти), що призводить до спостережуваних акторами результатів. Надпис може бути ім'ям або описом (з точки зору актора) того, "що" робить система (а не "як"). Ім'я прецедента зв’язано з неперервним (атомарним) сценарієм - конкретною послідовністю дій. Під час сценарію актори обмінюються із систематичними повідомленнями. Сценарій може бути приведений на діаграмі прецедентів у відео UML-коментарі. З одним прецедентом може бути пов'язано кілька різних сценаріїв.

На рисунку 3.2 зображено діаграму варіантів використання, яка описує можливі дії користувача в системі.

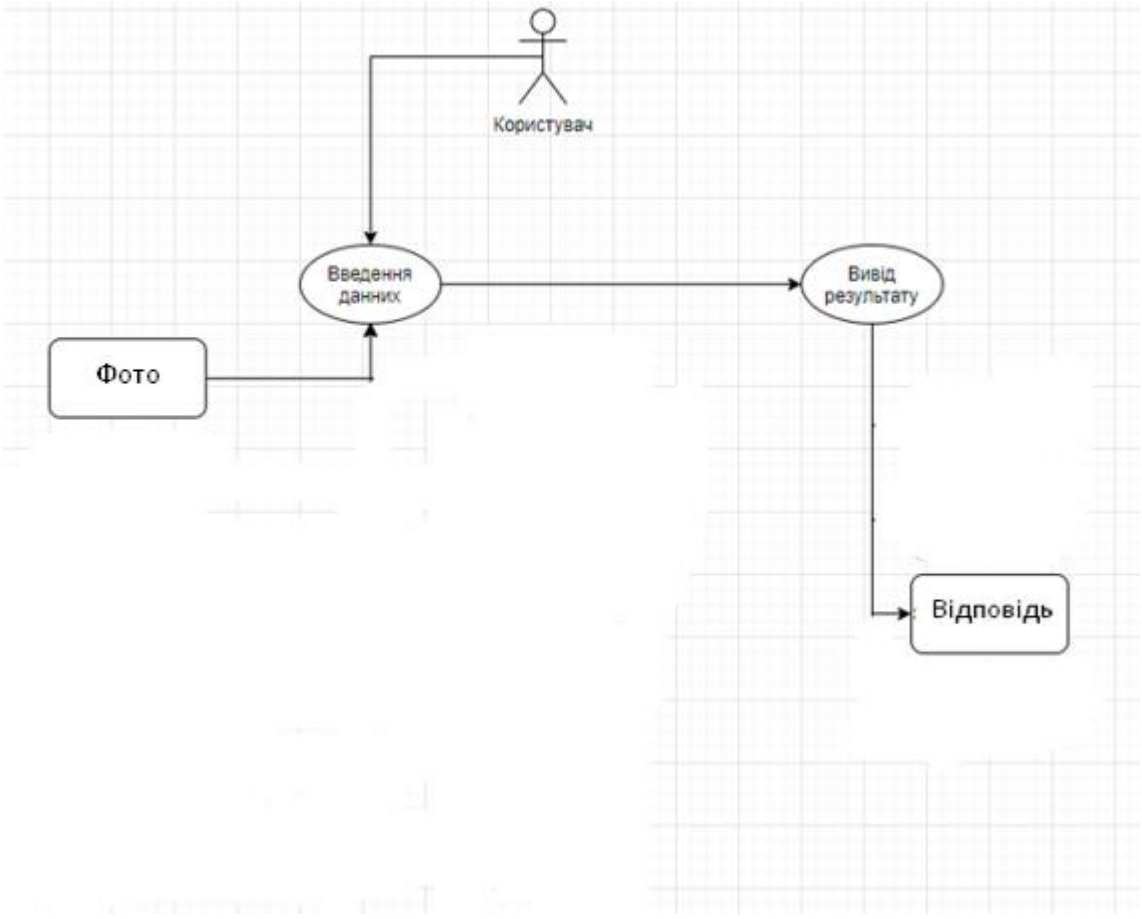


Рисунок 3.2 — Діаграма варіантів використання

3.3 Тестування системи

Для тестування розробленої системи було обрано метод чорного ящика. Тестування чорного ящика — це метод тестування програмного забезпечення, який перевіряє функціональність програми, не заглядаючи в її внутрішні структури чи роботи. Цей метод тестування можна застосувати практично на кожному рівні тестування програмного забезпечення: одиничному, інтеграційному, системному та приймальному. Його іноді називають тестуванням на основі специфікації.

Спеціальні знання коду програми, внутрішньої структури та знання програмування загалом не потрібні. Тестер знає, що програмне забезпечення має робити, але не знає, як воно це робить. Наприклад, тестувальник знає, що конкретні вхідні дані повертають певний незмінний вихід, але не знає, як програмне забезпечення виробляє вихідні дані в першу чергу.

Існує багато типів тестування чорної скриньки, але нижче наведено найвідоміші:

- Функціональне тестування – цей тип тестування чорної скриньки пов’язаний із функціональними вимогами системи; це роблять тестувальники програмного забезпечення;
- Нефункціональне тестування – цей тип тестування чорної скриньки пов’язаний не з тестуванням конкретної функціональності, а з нефункціональними вимогами, такими як продуктивність, масштабованість, зручність використання;
- Регресійне тестування – регресійне тестування виконується після виправлень коду, оновлень або будь-якого іншого обслуговування системи, щоб перевірити, чи новий код не вплинув на існуючий.

Тестові випадки будуються на основі специфікацій та вимог, тобто того, що програма повинна робити. Тестові випадки, як правило, походять із зовнішніх описів програмного забезпечення, включаючи специфікації, вимоги та параметри конструкції. Хоча використовувані тести мають переважно

функціональний характер, можуть використовуватися й нефункціональні тести. Конструктор тестів вибирає як дійсні, так і недійсні вхідні дані та визначає правильний вихід, часто за допомогою тестового оракула або попереднього результату, який, як відомо, є хорошим, без будь-якого знання внутрішньої структури тестового об’єкта.

Тестування починається з моделі прогнозування на рік, для даної моделі тест-кейси представлені в таблиці 3.1.

Таблиця 3.1 — Тест кейси для моделі прогнозування на рік

№ п.п.	Опис	Очікуваний результат	Отриманий результат
1	Введення даних, що вибиваються з ряду	Виведення результату, який не відповідає дійсності, проте є точним з огляду на введені данні	Виведення результату, який не відповідає дійсності, проте є точним з огляду на введені данні
2	Введення текстових даних замість зображень	Відповідне повідомлення про помилку в роботі програми	Відповідне повідомлення про помилку в роботі програми
3	Введення занадто великого зображення	Виведення результату, який не відповідає дійсності, проте є точним з огляду на введені данні	Виведення результату, який не відповідає дійсності, проте є точним з огляду на введені данні
4	Введення порожніх полів	Відповідне повідомлення про помилку в роботі програми	Відповідне повідомлення про помилку в роботі програми
55	Введення адекватних вхідних даних	Виведення результату	Виведення результату
6	Введення усіх полів порожніми	Відповідне повідомлення про помилку в роботі програми	Відповідне повідомлення про помилку в роботі програми

Тестування показує повну функціональну вірність, відсутність неточностей і багів в написаній системі.

3.4 Висновок

У даному розділі було розроблено загальну структурну схему функціонування додатку, представлено UML діаграму класів, та алгоритм функціонування. Розроблено дизайн інтерфейсу, та пояснено архітектурні особливості системи і їх вибір, на основі цього було реалізовано інформаційну систему фільтрації зображень.

Тестування даної системи було проведено за методом чорного, в результаті чого суттєвих дефектів не виявлено, та можна зробити висновок, що всі функцію працюють відповідно до їх функціоналу.

4 ЕКОНОМІЧНА ЧАСТИНА

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

Магістерська кваліфікаційна робота за темою: «Інформаційна технологія фільтрації зображень» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки.

Даний напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

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

2) розраховано витрати на здійснення науково-технічної розробки;

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

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія фільтрації зображень» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [76]

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в
Ринкові переваги (недоліки)					
	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на
	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за
	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно
	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в
Ринкові перспективи					

	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною
	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкуренті в немає

Продовження таблиці 4.1

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

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

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	5	5
2. Ринкові переваги (наявність аналогів)	1	1	1
3. Ринкові переваги (ціна продукту)	2	2	2
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	3	2	3
7. Ринкові перспективи (конкуренція)	3	2	3
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	4	4	3
10. Практична здійсненність (необхідність нових матеріалів)	5	5	5
11. Практична здійсненність (термін реалізації)	5	4	4
12. Практична здійсненність (розробка документів)	5	5	5
Сума балів	43	40	41
Середньоарифметична сума балів $СБ_c$	41,3		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [77].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

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

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія фільтрації зображень» становить 41,3 бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий). Розроблюване програмне забезпечення має такі переваги: 1) фільтрує зображення перед збереженням на сервер, що дозволає економити місце і зменшити оплату за використання серверу; 2) займає менше місця на жорсткому диску; 3) фільтрує на наявність реклами або інформації про наркотики, зброю, алкоголь та 18+ контенту в певному додатку, на відміну від конкурентів які надають ці всі фільтри окремими додатками чи API; 4) простий та зрозумілий інтерфейс роботи на відміну від конкурентів (google face recognition, тощо); 5) розміщується безпосередньо на серверах компанії, що дає більший контроль та забезпечує конфіденційність даних.

4.2 Оцінювання рівня новизни розробки

Виводячи на ринок новинку виробник вважає, що тієї новизни, якою наділена нова розробка є достатньо для того, щоб вона була сприйнята споживачем як нова. Але це не завжди так, в силу того, що споживач і виробник неоднозначно визначать її рівень новизни. Тому доцільним є визначення рівня новизни розробки отриманої в результаті досліджень за темою «Інформаційна технологія фільтрації зображень».

Саме визначення рівня і ступеня інтегральної новизни є найбільш актуальним, оскільки її рівень визначає ступінь однакового позитивного сприйняття новизни розробки як виробником, так і споживачем, а отже і ринком в цілому, а це, у свою чергу, є гарантією того, що новинка знайде своє місце на ринку, користуватиметься попитом у споживачів і забезпечить відшкодування витрат, зазнаних товаровиробником під час розроблення та виробництва ехнічної розробки [78].

Рівень новизни нової продукції розраховуємо експертним методом шляхом протиставлення нової продукції та її аналогів, що існують в даний час на ринку, за чинниками що визначають її значення, в системі «краще-гірше». Рівень новизни встановлюємо відносно рівня аналога (або продукту, що досить близький до аналога).

Для визначення *і*-го виду новизни, застосуємо чинники, які впливають на її рівень. Кожен чинник *і*-го виду новизни розраховуємо в балах. Більша кількість набраних балів свідчить про більший рівень новизни. Для оцінювання рівня новизни використаємо думки експертів, які встановлюють визначені бали відповідним чинникам. Бал відповідності проставляється в діапазоні від (-5 –значно гірше аналога до +5 – значно краще аналога). Результати попереднього оцінювання зведемо до відповідного листа оцінювання (таблиця 4.4).

Таблиця 4.4 – Лист оцінювання рівня новизни експертами

Види та чинники		Бали та експерти		
		Експерт 1	Експерт 2	Експерт 3
1		2	3	4
Споживча новизна	Питома вага 0,24	Максимальний бал $Б_{i\ MAX}$		25

Продовження таблиці 4.4

1. Зміна поведінкових звичок споживача		4	4	4
2. Ступінь задоволення потреб і запитів		5	5	5
3. Спосіб задоволення потреби		4	4	4
4. Формування нової потреби		4	4	4
5. Формування нового споживача		1	1	1
Середній бал експертів $Б_{i\ отр}$		18		
Товарна новизна	Питома вага 0,202	Максимальний бал $Б_{i\ MAX}$		30
1. Параметричні зміни показників продукції				
1.1. Якісні		4	4	4
1.2. Технічні		3	4	3
1.3. Економічні		3	3	3
1.4. Сервісні		4	4	4
2. Якість продукції по відношенню до конкурентів		3	3	3
3. Функціональні зміни		4	4	4
Середній бал експертів $Б_{i\ отр}$		21		
Виробнича новизна	Питома вага 0,042	Максимальний бал $Б_{i\ MAX}$		25
1. Рівень унікальності товару для підприємства		5	5	5
2. Рівень унікальності для галузі		3	3	3
3. Рівень унікальності товару для країни		1	1	1
4. Зміна виробничої системи		4	4	4
5. Відносно існуючого асортименту		2	2	2
Середній бал експертів $Б_{i\ отр}$		15		
Прогресивна новизна	Питома вага 0,2	Максимальний бал $Б_{i\ MAX}$		25

Продовження таблиці 4.4

1. Зміна технології виготовлення		4	4	4
2. Рівень застосування нових компонентів і матеріалів		2	2	2
3. Зміна технологічного принципу дії виробу		1	2	1
4. Зміна конструктивного виконання		3	3	3
5. Рівень застосування інновацій		2	2	2
Середній бал експертів $Б_{i\ отр}$		12		
Ринкова новизна	Питома вага 0,1	Максимальний бал $Б_{i\ MAX}$		20
1. Новий виріб на новому ринку		0	0	0
2. Новий виріб на відомому ринку		4	4	4
3. Модернізований виріб		2	2	2
4. Нова модель		3	3	3
Середній бал експертів $Б_{i\ отр}$		9		
Екологічна новизна	Питома вага 0,035	Максимальний бал $Б_{i\ MAX}$		20
1. Рівень екологічної чистоти технології виробництва		5	5	5
2. Рівень впровадження мало- та безвідходних технологій		5	5	5
3. Рівень екологічно небезпечних режимів експлуатації продукції		5	5	5
4. Рівень забруднення навколишнього середовища		5	5	5
Середній бал експертів $Б_{i\ отр}$		20		
Соціальна новизна	Питома вага 0,036	Максимальний бал $Б_{i\ MAX}$		20

Продовження таблиці 4.4

1. Використання нового товару приводить до покращення стану здоров'я нації	0	0	0
2. Використання нового товару приводить до зростання доходів населення	0	0	0
3. Виробництво нового товару приводить до збільшення (зменшення) кількості робочих місць на підприємстві	4	5	4
4. Виробництво нового товару приводить до підвищення кваліфікації персоналу	3	3	3
Середній бал експертів $Б_{i\ отр}$	7		
Маркетингова новизна / Питома вага 0,145	Максимальний бал $Б_{i\ MAX}$		20
1. Нові методи маркетингових досліджень	0	0	0
2. Вживання нових стратегій сегментації ринку	3	3	3
3. Вибір нової маркетингової стратегії обхвату і розвитку цільового сегмента	1	1	2
4. Побудова нових каналів збуту	2	1	1
Середній бал експертів $Б_{i\ отр}$	6		

Значення *i*-го виду новизни розрахуємо за формулою [42]:

$$I_i = \frac{Б_{i\,отр}}{Б_{i\,MAX}}, \tag{4.1}$$

де $Б_{i\ отр}$ – отримана кількість балів за шкалою оцінок чинників, що визначають *i*-й вид новизни;

$Б_{i\ MAX}$ – максимальна кількість балів, що може бути отримана за *i*-м видом новизни.

Загальний рівень інтегральної новизни розраховуємо шляхом перемноження отриманого значення *i*-го виду новизни на її вагомість, причому

вагомість *i*-го виду новизни визначаємо експертним методом, за формулою [43]:

$$N_{інт} = \sum_{i}^{n} W_i \cdot I_i \,, \tag{4.2}$$

де $N_{інт}$ – рівень інтегральної (сукупної) новизни;

W_i – вагомість (питома вага) *i*-го виду новизни;

n – загальна кількість видів новизни.

$$N_{інт} = (0{,}24 \cdot 18/25)+(0{,}202 \cdot 21/30)+(0{,}042 \cdot 15/25)+(0{,}2 \cdot 12/25)+(0{,}1 \cdot 9/20)+ (0{,}035 \cdot 20/20)+(0{,}036 \cdot 7/20)+(0{,}145 \cdot 6/20) = 0{,}575.$$

Отримане значення інтегрального рівня новизни зіставляємо зі шкалою, що наведена в табл. 4.5 [41].

Таблиця 4.5 – Рівні новизни нового товару та їхня характеристика

Рівні новизни товару	Значення інтегральної новизни	Характеристика товару	Вид нового товару
Найвища	1,00	Абсолютно новий товар	Новий товар, що наділений ознаками інноваційності (інноваційний товар)
Висока	0,8…0,99	Товар, який не має аналогів	
Значуща	0,6…0,79	Принципова зміна споживчих властивостей товару	
Достатня	0,4…0,59	Принципова технологічна модифікація товару	
Незначна	0,2…0,39	Кардинальна зміна параметрів	Новий товар
Помилкова	0,00…0,19	Малоістотна модифікація	

Згідно таблиці 4.5 розробка відповідає рівню при значенні інтегральної новизни 0,575 - достатня новизна; за характеристикою: принципова

технологічна модифікація товару; вид розробки - новий товар, що наділений ознаками інноваційності (інноваційний товар).

4.3 Розрахунок витрат на проведення науково-дослідної роботи

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

4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [41]

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (4.3)$$

де k – кількість посад дослідників залучених до процесу досліджень;

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

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=20 дні.

$$З_о = 15100{,}00 \cdot 35 / 20 = 26425{,}00 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.6 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник дослідження	15100,00	755,00	35	26425,00
Інженер-програміст 1-ї категорії	14650,00	732,50	20	14650,00
Інженер-дослідник	14600,00	730,00	15	10950,00
Всього				52025,00

Основна заробітна плата робітників

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

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \qquad (4.4)$$

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

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

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

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \qquad (4.5)$$

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

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [41];

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

T_p – середнє число робочих днів в місяці, приблизно T_p = 20 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (20 \cdot 8) = 76{,}00$ *грн.*

$З_{р1} = 76{,}00 \cdot 12{,}00 = 912{,}04$ *грн.*

Таблиця 4.7 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Підготовка робочого місця розробника програмного забезпечення	12,00	2	1,10	76,00	912,04
Підготовка серверного обладнання	16,00	3	1,35	93,28	1492,43
Інсталяція програмного забезпечення для розробки програмного забезпечення	5,50	4	1,50	103,64	570,02
Компіляція програмних блоків	4,35	5	1,70	117,46	510,95
Налагодження програмних блоків	6,45	5	1,70	117,46	757,61
Тестування системи	10,00	3	1,35	93,28	932,77
Всього					5175,81

Додаткова заробітна плата дослідників та робітників

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

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.6)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод}$ = (52025,00 + 5175,81) · 10 / 100% = 5720,08 грн.

4.3.2 Відрахування на соціальні заходи

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

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.7)$$

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

Зн = (52025,00 + 5175,81 + 5720,08) · 22 / 100% = 13842,60 грн.

4.3.3 Сировина та матеріали

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

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

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.8)$$

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

n – кількість видів матеріалів;

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

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

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

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

$M_1 = 3,0 \cdot 273,00 \cdot 1,1 - 0 \cdot 0 = 900,90$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.8 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір 500 80 г\м	273,00	3,0	0	0	900,90
Папір для записів 100 70 г\м	155,00	5,0	0	0	852,50
Органайзер офісний	174,00	2,0	0	0	382,80
Набір офісний (канцелярське приладдя)	222,00	3,0	0	0	732,60
Картридж для принтера	1465,00	1,0	0	0	1611,50
Диск оптичний CD-R	23,25	3,0	0	0	76,73
Flesh-пам'ять 64 GB	625,00	1,0	0	0	687,50
Тека для паперів	97,00	4,0	0	0	426,80
Всього					5671,33

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія фільтрації зображень», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.9)$$

де $Н_j$ – кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*).

$К_в = 1 \cdot 4640{,}00 \cdot 1{,}12 = 5196{,}80$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.9 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Router TP-Link230	1	4640,00	5196,80
База даних (тестувальна) DataBase 64-A10	1	2850,00	3192,00
Всього			8388,80

4.3.5 Спецустаткування для наукових (експериментальних) робіт

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

Балансову вартість спецустаткування розраховуємо за формулою:

$$В_{спец} = \sum_{i=1}^{k} Ц_i \cdot С_{пр.i} \cdot К_i \ , \qquad (4.10)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.і}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

$K_і$ – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_і$ = *1,10…1,12);*

k – кількість найменувань устаткування.

$B_{спец}$ = *40325,00 · 1 · 1,1 = 44357,50 грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.10 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Серверне обладнання DEXX FIJ1313C-UA	1	40325,00	44357,50
Всього			44357,50

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

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

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.і} \cdot K_i \ , \qquad (4.11)$$

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

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

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

k – кількість найменувань програмних засобів.

$B_{прг} = 10105{,}00 \cdot 1 \cdot 1{,}1 = 11115{,}50$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.11 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Середовище розробки програмного забезпечення PyCharm	1	10105,00	11115,50
Середовище мови програмування Python	1	4200,00	4620,00
Бібліотека машинного навчання Tensor Flow	1	9380,00	10318,00
Всього			26053,50

4.3.7 Амортизація обладнання, програмних засобів та приміщень

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

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.12)$$

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

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

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

$A_{обл} = (27820{,}00 \cdot 2) / (2 \cdot 12) = 2318{,}33$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.12 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер розробника ПЗ	27820,00	2	2	2318,33
Персональний комп'ютер інженера-дослідника	23560,00	2	2	1963,33
Робоче місце інженера-програміста	9230,00	5	2	307,67
Робоче місце інженера-дослідника	9500,00	5	2	316,67
Пристрої передачі даних	6600,00	5	2	220,00
Оргтехніка	8360,00	5	2	278,67
Приміщення лабораторії розробки ПЗ	400000,00	25	2	2666,67
ОС Windows	8580,00	2	2	715,00
Прикладний пакет Microsoft Office	7860,00	2	2	655,00
Всього				9441,33

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.13)$$

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

t_i – тривалість роботи обладнання на етапі дослідження, год;

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

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

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

$B_e = 0,42 \cdot 240,0 \cdot 6,20 \cdot 0,95 / 0,97 = 624,96$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.13 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер розробника ПЗ	0,42	240,0	624,96
Персональний комп'ютер інженера-дослідника	0,05	240,0	74,40
Робоче місце інженера-програміста	0,12	240,0	178,56
Робоче місце інженера дослідника	0,10	240,0	148,80
Пристрої передачі даних	0,05	45,0	13,95
Оргтехніка	0,45	8,5	23,72
Серверне обладнання DEXX FIJ1313C-UA	0,32	45,0	89,28
Router TP-Link230	0,01	45,0	2,79
Всього			1156,46

4.3.9 Службові відрядження

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

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

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (4.14)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 23%.

$B_{св}$ = (52025,00 + 5175,81) · 23 / 100% = 13156,19 грн.

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

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

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (4.15)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 42%.

$B_{сп}$ = (52025,00 + 5175,81) · 42 / 100% = 24024,34 грн.

4.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.16)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 100%.

$I_в$ = (52025,00 + 5175,81) · 100 / 100% = 57200,81 грн.

4.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.17)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (52025,00 + 5175,81) · 100 / 100% = 57200,81 грн.

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

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

(4.18)

$B_{заг}$= 52025,00 +5175,81 +5720,08 +13842,5967 +5671,33 +8388,80 +44357,50 + 26053,50 + 9441,33 +1156,46 +13156,19 +24024,34 +57200,81 +57200,81 = 323414,56 грн.

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

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.19)$$

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

ЗВ = 323414,56 / 0,9 = 359349,51 грн.

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Інформаційна технологія фільтрації зображень» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

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

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

Таблиця 4.14 – Майбутній економічний ефект

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	500	800	1000	800

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 2000 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 9750,00 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 576,63 грн.

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

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.20)$$

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

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 45\%$;

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

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (576,63 \cdot 2000,00 + 10326,63 \cdot 500) \cdot 0,83 \cdot 0,45 \cdot (1 - 0,18/100\%) = 1934573,60$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(576,63·2000,00+10326,63·1300)·0,83·0,45·(1-0,18/100%)=4464761,95 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(576,63·2000,00+10326,63·2300)·0,83·0,45·(1-0,18/100%)=7627497,39 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4$ =(576,63·2000,00+10326,63·3100)·0,83·0,45·(1-0,18/100%)=10157685,74 грн.

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

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.21)$$

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

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

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,19;

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

ПП =1934573,60/(1+0,19)^1+4464761,95/(1+0,19)^2+7627497,39/(1+0,19)^3+ +10157685,74/(1+0,19)^4=1625692,10+3152857,81+4526277,57+5065320,46=14370147,95 грн.

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

$$PV = k_{інв} \cdot ЗВ, \tag{4.22}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2,5;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 359349,51 грн.

$PV = k_{інв} \cdot ЗВ$ = 2,5 · 359349,51 = 898373,77 грн.

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

$$E_{абс} = ПП - PV \tag{4.23}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 14370147,95 грн;

PV – теперішня вартість початкових інвестицій, 898373,77 грн.

$E_{абс} = ПП - PV$ = 14370147,95 - 898373,77 = 13471774,17 грн.

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

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.24}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 13471774,17 грн;

PV – теперішня вартість початкових інвестицій, 898373,77 грн;

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

$$E_{в} = \sqrt[Тж]{1+\frac{E_{абс}}{PV}} - 1 = (1+13471774,17/898373,77)^{1/4} = 1,00.$$

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

$$\tau_{мін} = d + f, \qquad (4.25)$$

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

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,55.

$\tau_{мін}$ = 0,12+0,55 = 0,67 < 1,00 свідчить про те, що внутрішня економічна дохідність інвестицій $E_{в}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія фільтрації зображень» доцільно.

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

$$T_{ок} = \frac{1}{E_{в}}, \qquad (4.26)$$

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

$T_{ок}$ = 1 / 1,00 = 1,00 р.

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

4.5 Висновок

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія фільтрації зображень» становить 41,3 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

Також термін окупності становить 1,00 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія фільтрації зображень».

ВИСНОВКИ

В результаті виконання магістерської кваліфікаційної роботи була розроблена інформаційна система фільтрації зображень, яка аналізує зображення на вміст шкідливого контенту за допомогою бібліотеки TensorFlow та Keras безпосередньо перед збереженням до сховища.

Розроблена система може використовуватися при вирішенні різних задач фільтрації, і, в першу чергу, має безпосереднє застосування в системах контролю і ідентифікації контенту.

В ході написання першого розділу роботи було виявлено значення основних понять, таких як фільтрація контенту та нейронні мережі. Було проаналізовано методи фільтрації контенту, основні поняття нейронних мереж. Також виявлено фактори які впливають на якість розпізнавання, до таких відносяться: освітлення, позиціонування тощо.

Другий розділ роботи присвячено огляду інструментальних засобів розробки системи, таких як мова програмування і фреймворки, що дозволяють працювати з нейронними мережами. В результаті чого було обрано мову програмування Python та бібліотеку Tensorflow в якості основних засобів для розробки, в якості середовища для розробки було обрано PyCharm.

Третій розділ роботи містить в собі інформацію стосовно розробки системи, таку як: аналіз варіантів використання системи, на основі якого побудовано системи; графічне представлення внутрішньої будови системи, яка дозволяє зробити висновок про схему системи; тестування системи, націлене на виявлення неточностей в роботі і перевірку роботоздатності в цілому.

Аналіз результатів тестування показує, що достовірність розпізнавання не бажаного контенту становить 93%, у той час коли достовірність розпізнавання програм – аналогів складає 75% – 85% при навчанні на однаковій навчальній множині у 300 зображень. Тобто мета магістерської

кваліфікаційної роботи досягнута – ефективність ідентифікації даних типу «шкідливий контент» за допомогою інформаційної системи фільтрації зобрежнь підвищена на 8%.

Тестування системи проводилося за методом чорної скриньки, в результаті чого дефектів не виявлено, з цього слідує висновок що весь функціонал працює правильно.

Виходячи з проведених економічних досліджень, тема «Інформаційні технології для фільтрації зображень» має рівень потенціалу розвитку бізнесу 41,3 бали, що свідчить про комерційну важливість проведення даних досліджень (високий рівень потенціалу розвитку бізнесу). Період окупності становить 1 рік, що вказує на те, що технологічна розробка є комерційно привабливою та може спонукати потенційних інвесторів фінансувати впровадження розробки та виводити її на ринок

В результаті проведеної роботи отримано повнофункціональну, навчену модель нейронної мережі, призначену для фільтрації забороненого контенту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сулима Я. О., Барабан С.В. «Інформаційна технологія фільтрації зображень» [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/16870.

2. McCulloch, Warren; Walter Pitts (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". Bulletin of Mathematical Biophysics. 5 (4): 115–133. doi:10.1007/BF02478259.

3. Kleene, S.C. (1956). "Representation of Events in Nerve Nets and Finite Automata". Annals of Mathematics Studies (34). Princeton University Press. pp. 3–41. Retrieved 17 June 2017.

4. Hebb, Donald (1949). The Organization of Behavior. New York: Wiley. ISBN 978-1-135-63190-1.

5. Farley, B.G.; W.A. Clark (1954). "Simulation of Self-Organizing Systems by Digital Computer". IRE Transactions on Information Theory. 4 (4): 76–84. doi:10.1109/TIT.1954.1057468.

6. Haykin (2008) Neural Networks and Learning Machines, 3rd edition

7. Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model For Information Storage And Organization in the Brain". Psychological Review. 65 (6): 386–408. CiteSeerX 10.1.1.588.3775. doi:10.1037/h0042519. PMID 13602029.

8. Werbos, P.J. (1975). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.

9. Rosenblatt, Frank (1957). "The Perceptron—a perceiving and recognizing automaton". Report 85-460-1. Cornell Aeronautical Laboratory.

10. Olazaran, Mikel (1996). "A Sociological Study of the Official History of the Perceptrons Controversy". Social Studies of Science. 26 (3): 611–659. doi:10.1177/030631296026003005. JSTOR 285702. S2CID 16786738.

11. Schmidhuber, J. (2015). "Deep Learning in Neural Networks: An Overview". Neural Networks. 61: 85–117. arXiv:1404.7828. doi:10.1016/j.neunet.2014.09.003. PMID 25462637. S2CID 11715509.

12. Ivakhnenko, A. G. (1973). Cybernetic Predicting Devices. CCM Information Corporation.

13. Ivakhnenko, A. G.; Grigor′evich Lapa, Valentin (1967). Cybernetics and forecasting techniques. American Elsevier Pub. Co.

14. Schmidhuber, Jürgen (2015). "Deep Learning". Scholarpedia. 10 (11): 85–117. Bibcode:2015SchpJ..1032832S. doi:10.4249/scholarpedia.32832.

15. Dreyfus, Stuart E. (1 September 1990). "Artificial neural networks, back propagation, and the Kelley-Bryson gradient procedure". Journal of Guidance, Control, and Dynamics. 13 (5): 926–928. Bibcode:1990JGCD...13..926D. doi:10.2514/3.25422. ISSN 0731-5090.

16. Mizutani, E.; Dreyfus, S.E.; Nishio, K. (2000). "On derivation of MLP backpropagation from the Kelley-Bryson optimal-control gradient formula and its application". Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium. IEEE: 167–172 vol.2. doi:10.1109/ijcnn.2000.857892. ISBN 0-7695-0619-4. S2CID 351146.

17. Kelley, Henry J. (1960). "Gradient theory of optimal flight paths". ARS Journal. 30 (10): 947–954. doi:10.2514/8.5282.

18. "A gradient method for optimizing multi-stage allocation processes". Proceedings of the Harvard Univ. Symposium on digital computers and their applications. April 1961.

19. Minsky, Marvin; Papert, Seymour (1969). Perceptrons: An Introduction to Computational Geometry. MIT Press. ISBN 978-0-262-63022-1.

20. Linnainmaa, Seppo (1970). The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors (Masters) (in Finnish). University of Helsinki. pp. 6–7.

21. Linnainmaa, Seppo (1976). "Taylor expansion of the accumulated rounding error". BIT Numerical Mathematics. 16 (2): 146–160. doi:10.1007/bf01931367. S2CID 122357351.

22. Dreyfus, Stuart (1973). "The computational solution of optimal control problems with time lag". IEEE Transactions on Automatic Control. 18 (4): 383–385. doi:10.1109/tac.1973.1100330.

23. Werbos, Paul (1982). "Applications of advances in nonlinear sensitivity analysis" (PDF). System modeling and optimization. Springer. pp. 762–770.

24. Mead, Carver A.; Ismail, Mohammed (8 May 1989). Analog VLSI Implementation of Neural Systems (PDF). The Kluwer International Series in Engineering and Computer Science. 80. Norwell, MA: Kluwer Academic Publishers. doi:10.1007/978-1-4613-1639-8. ISBN 978-1-4613-1639-8.

25. David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams , "Learning representations by back-propagating errors ," Nature', 323, pages 533–536 1986.

26. J. Weng, N. Ahuja and T. S. Huang, "Cresceptron: a self-organizing neural network which grows adaptively," Proc. International Joint Conference on Neural Networks, Baltimore, Maryland, vol I, pp. 576–581, June 1992.

27. J. Weng, N. Ahuja and T. S. Huang, "Learning recognition and segmentation of 3-D objects from 2-D images," Proc. 4th International Conf. Computer Vision, Berlin, Germany, pp. 121–128, May 1993.

28. J. Weng, N. Ahuja and T. S. Huang, "Learning recognition and segmentation using the Cresceptron," International Journal of Computer Vision, vol. 25, no. 2, pp. 105–139, Nov. 1997.

29. J. Schmidhuber., "Learning complex, extended sequences using the principle of history compression," Neural Computation, 4, pp. 234–242, 1992.

30. Domingos, Pedro (22 September 2015). The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World. chapter 4: Basic Books. ISBN 978-0465065707.

31. Smolensky, P. (1986). "Information processing in dynamical systems: Foundations of harmony theory.". In D. E. Rumelhart; J. L. McClelland; PDP Research Group (eds.). Parallel Distributed Processing: Explorations in the Microstructure of Cognition. 1. pp. 194–281. ISBN 978-0-262-68053-0.

32. Ng, Andrew; Dean, Jeff (2012). "Building High-level Features Using Large Scale Unsupervised Learning". arXiv:1112.6209 [cs.LG].

33. Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016). Deep Learning. MIT Press.

34. Cireşan, Dan Claudiu; Meier, Ueli; Gambardella, Luca Maria; Schmidhuber, Jürgen (21 September 2010). "Deep, Big, Simple Neural Nets for Handwritten Digit Recognition". Neural Computation. 22 (12): 3207–3220. arXiv:1003.0358. doi:10.1162/neco_a_00052. ISSN 0899-7667. PMID 20858131. S2CID 1918673.

35. Dominik Scherer, Andreas C. Müller, and Sven Behnke: "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition," In 20th International Conference Artificial Neural Networks (ICANN), pp. 92–101, 2010. doi:10.1007/978-3-642-15825-4_10.

36. 2012 Kurzweil AI Interview Archived 31 August 2018 at the Wayback Machine with Jürgen Schmidhuber on the eight competitions won by his Deep Learning team 2009–2012

37. "How bio-inspired deep learning keeps winning competitions | KurzweilAI". www.kurzweilai.net. Archived from the original on 31 August 2018. Retrieved 16 June 2017.

38. Graves, Alex; and Schmidhuber, Jürgen; Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks, in Bengio, Yoshua; Schuurmans, Dale; Lafferty, John; Williams, Chris K. I.; and Culotta, Aron (eds.), Advances in Neural Information Processing Systems 22 (NIPS'22), 7–10 December 2009, Vancouver, BC, Neural Information Processing Systems (NIPS) Foundation, 2009, pp. 545–552.

39. Graves, A.; Liwicki, M.; Fernandez, S.; Bertolami, R.; Bunke, H.; Schmidhuber, J. (2009). "A Novel Connectionist System for Improved Unconstrained Handwriting Recognition" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 31 (5): 855–868. CiteSeerX 10.1.1.139.4502. doi:10.1109/tpami.2008.137. PMID 19299860. S2CID 14635907.

40. Graves, Alex; Schmidhuber, Jürgen (2009). Bengio, Yoshua; Schuurmans, Dale; Lafferty, John; Williams, Chris editor-K. I.; Culotta, Aron (eds.). "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks". Neural Information Processing Systems (NIPS) Foundation. Curran Associates, Inc: 545–552.

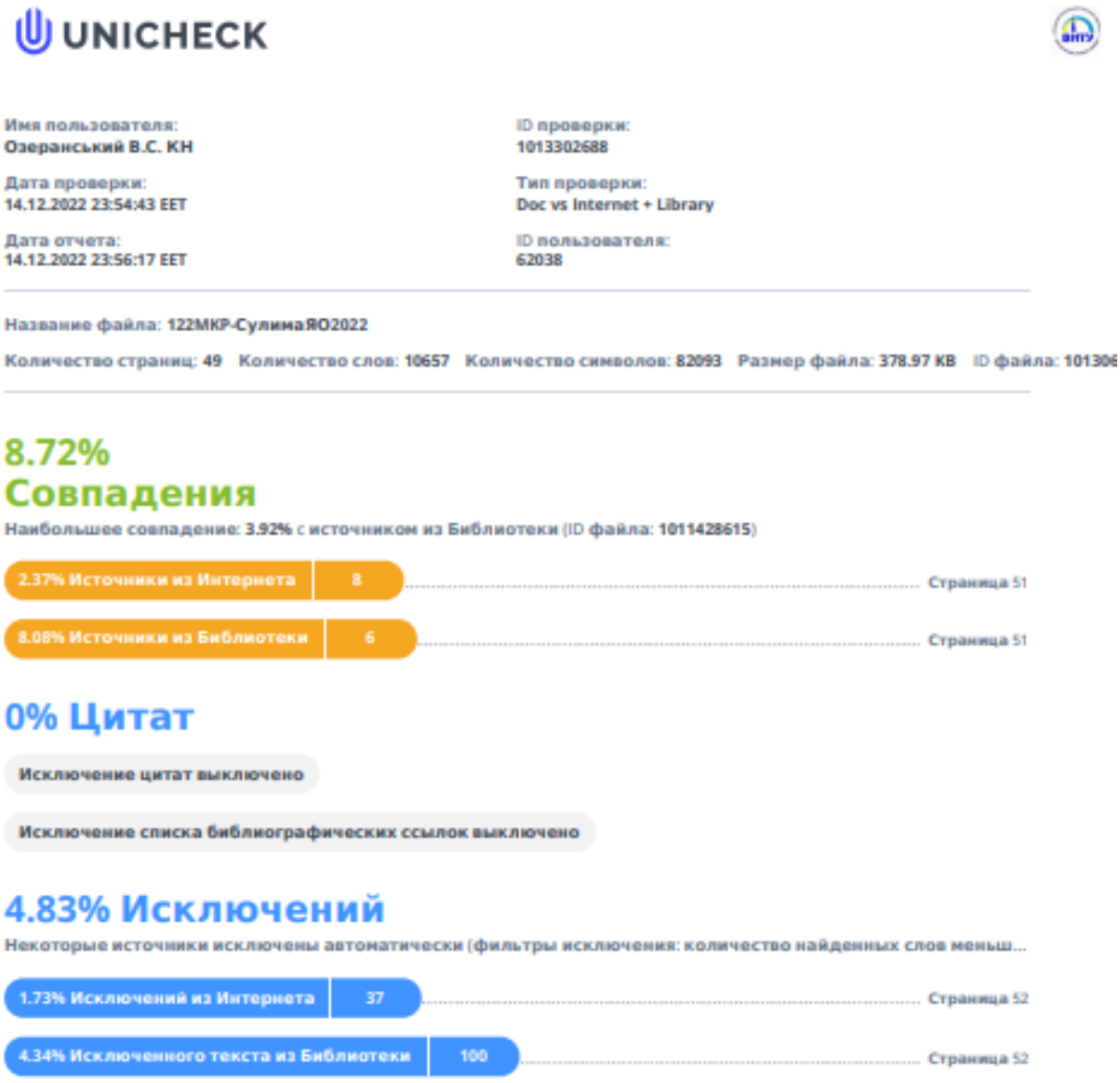
41. Graves, A.; Liwicki, M.; Fernández, S.; Bertolami, R.; Bunke, H.; Schmidhuber, J. (May 2009). "A Novel Connectionist System for Unconstrained Handwriting Recognition". IEEE Transactions on Pattern Analysis and Machine Intelligence. 31 (5): 855–868. CiteSeerX 10.1.1.139.4502. doi:10.1109/tpami.2008.137. ISSN 0162-8828. PMID 19299860. S2CID 14635907.

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

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

Додаток А (обов'язковий)

Результат перевірки на плагіат в онлайн-системі UNICHECK



UNICHECK

Имя пользователя:
Озеранський В.С. КН

ID проверки:
1013302688

Дата проверки:
14.12.2022 23:54:43 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
14.12.2022 23:56:17 EET

ID пользователя:
62038

Название файла: 122МКР-СулимаЯО2022

Количество страниц: 49 Количество слов: 10657 Количество символов: 82093 Размер файла: 378.97 KB ID файла: 101306

8.72%
Совпадения

Наибольшее совпадение: 3.92% с источником из Библиотеки (ID файла: 1011428615)

2.37% Источники из Интернета 8 Страница 51

8.08% Источники из Библиотеки 6 Страница 51

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

4.83% Исключений

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

1.73% Исключений из Интернета 37 Страница 52

4.34% Исключенного текста из Библиотеки 100 Страница 52

Додаток Б (обов'язковий)

Лістинг програми

```
import glob
import os

import matplotlib.pyplot as plot
import keras
from keras.models import Model
from keras.preprocessing import image
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D, Activation, AveragePooling2D,
BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import load_img
from tensorflow.keras.optimizers import Adam

trainDirectory = "./data/train_set/"
testDirectory = "./data/test_set/"
class_indices = {
    'Violence': 0,
    'Drugs': 1,
    'Guns': 2,
    'Alcohol': 3
}


def GetImages(dir):
    if not os.path.exists(dir):
        return 0
    numbers = 0
    for currentDirectory, directories, files in os.walk(dir):
        for directory in directories:
            numbers += len(glob.glob(os.path.join(currentDirectory, directory
+ "/*")))
    return numbers


trainExamples = GetImages(trainDirectory)
countClasses = len(glob.glob(trainDirectory + "/*"))
testExamples = GetImages(testDirectory)
print("Кількість класів:", countClasses)
print("Кількість зразків для навчання:", trainExamples)
print("Кількість зразків для тестування:", testExamples)
trainImageDataGenerator = ImageDataGenerator(rescale=1. / 255,
                                             shear_range=0.2,
                                             zoom_range=0.2,
                                             validation_split=0.2,  # Частка
перевірки 20%
                                             horizontal_flip=True)
testImageDataGenerator = ImageDataGenerator(rescale=1. / 255)
imageWidth, imageHeight = 256, 256
inputShape = (imageWidth, imageHeight, 3)
sizeBatch = 32
trainGenerator = trainImageDataGenerator.flow_from_directory(trainDirectory,

target_size=(imageWidth, imageHeight),
```

```
batch_size=sizeBatch)
testGenerator = testImageDataGenerator.flow_from_directory(testDirectory,
                                                           shuffle=True,

target_size=(imageWidth, imageHeight),

batch_size=sizeBatch)
m = Sequential()
m.add(Conv2D(32, (5, 5), input_shape=inputShape,
             activation='relu'))  # 32 Об'єкти повинні бути вилучені з
вхідного зображення / Relu - лінійна активація для активації нейронів
m.add(MaxPooling2D(pool_size=(3, 3)))
m.add(Conv2D(32, (3, 3), activation='relu'))
m.add(MaxPooling2D(pool_size=(2, 2)))
m.add(Conv2D(64, (3, 3), activation='relu'))
m.add(MaxPooling2D(pool_size=(2, 2)))
m.add(Flatten())  # Flatten перетворює шар 4D-масиву до 1D-масиву
m.add(Dense(512, activation='relu'))
m.add(Dropout(0.25))  # Dropout змушує деякі нейрони ослабнути
m.add(Dense(128, activation='relu'))

m.add(Dense(countClasses, activation='softmax'))  # activation softmax для
виходів ймовірності
m.summary()

layers = [layer.name for layer in m.layers]
print('Назви шарів: ', layers)
image1 = image.load_img('data/test_set/Alcohol/images.jpg', target_size=(256,
256))
image = image.img_to_array(image1)
image = image / 255
image = np.expand_dims(image, axis=0)

conv2d = Model(inputs=m.input, outputs=m.get_layer('conv2d').output)
max_pooling2d = Model(inputs=m.input,
outputs=m.get_layer('max_pooling2d').output)
conv2d_1 = Model(inputs=m.input, outputs=m.get_layer('conv2d_1').output)
max_pooling2d_1 = Model(inputs=m.input,
outputs=m.get_layer('max_pooling2d_1').output)
conv2d_2 = Model(inputs=m.input, outputs=m.get_layer('conv2d_2').output)
max_pooling2d_2 = Model(inputs=m.input,
outputs=m.get_layer('max_pooling2d_2').output)
flatten = Model(inputs=m.input, outputs=m.get_layer('flatten').output)
conv2dFeatures = conv2d.predict(image)
max_pooling2dFeatures = max_pooling2d.predict(image)
conv2d_1Features = conv2d_1.predict(image)
max_pooling2d_1Features = max_pooling2d_1.predict(image)
conv2d_2Features = conv2d_2.predict(image)
max_pooling2d_2Features = max_pooling2d_2.predict(image)
flattenFeatures = flatten.predict(image)
figure = plot.figure(figsize=(14, 7))

validationGenerator =
trainImageDataGenerator.flow_from_directory(trainDirectory,
                                                                    # Той самий
каталог, що й навчальні дані

target_size=(imageHeight, imageWidth),

batch_size=sizeBatch)
optimizer = Adam(lr=0.001)
m.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['accuracy'])
train = m.fit_generator(trainGenerator,
```

```
                        epochs=15,
                        steps_per_epoch=trainGenerator.samples // sizeBatch,
                        validation_data=validationGenerator,
                        validation_steps=validationGenerator.samples //
sizeBatch,
                        verbose=1)

accuracy = train.history['accuracy']
valAccuracy = train.history['val_accuracy']
loss = train.history['loss']
valLoss = train.history['val_loss']
epochs = range(1, len(accuracy) + 1)
plot.plot(epochs, accuracy, 'b', label='Точність тренування')
plot.plot(epochs, valAccuracy, 'r', label='Точність перевірки')
plot.title('Точність навчання та перевірки')
plot.legend()
plot.figure()
plot.plot(epochs, loss, 'b', label='Втрати тренування')
plot.plot(epochs, valLoss, 'r', label='Втрати перевірки')
plot.title('Втрати навчання та переірки')
plot.legend()
plot.show()

score, accuracy = m.evaluate(testGenerator, verbose=1)
print("Оцінка тесту становить {}".format(score))
print("Точність тесту становить {}".format(accuracy))
```

Додаток В (обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ФІЛЬТРАЦІЇ ЗОБРАЖЕНЬ

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

Керівник: к.т.н., доцент каф. КН
Барабан С.В.
(прізвище та ініціали)
«_____»_____2022 р.

Вінниця ВНТУ – 2022 рік

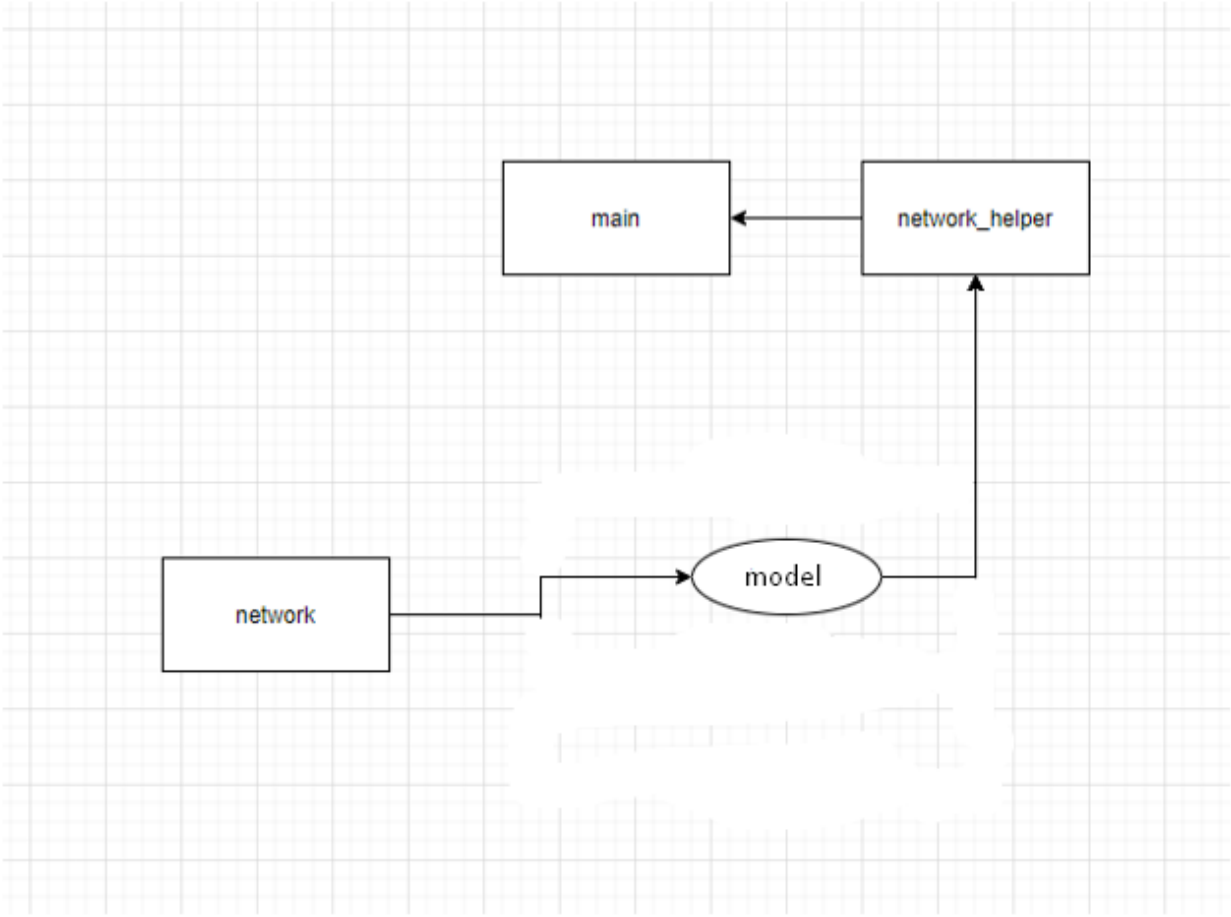


Рисунок В.1 Діаграма класів

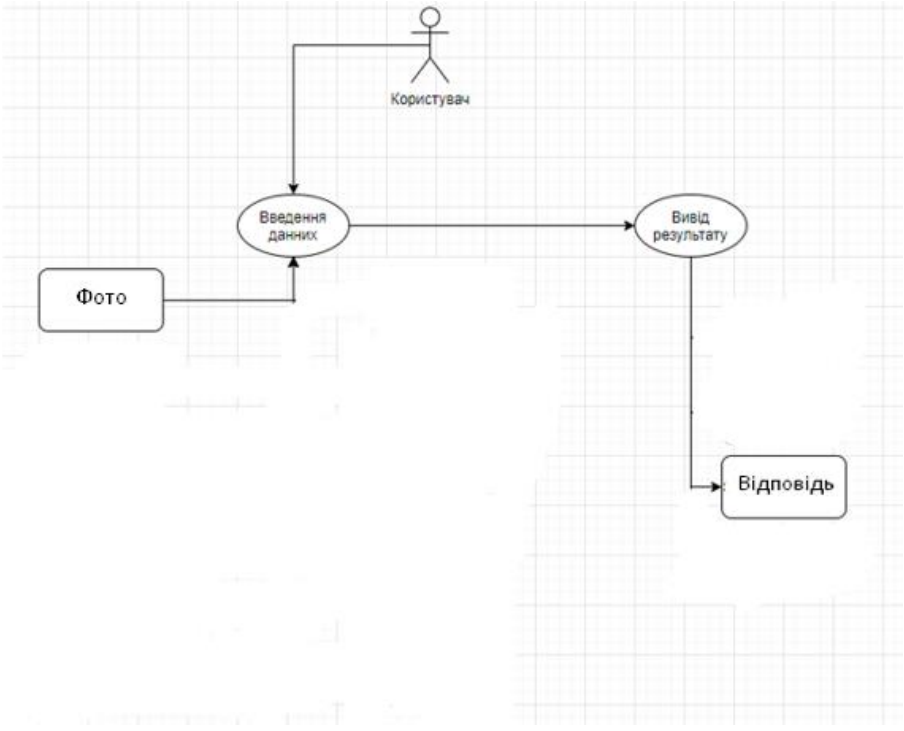


Рисунок В.2 Діаграма використання

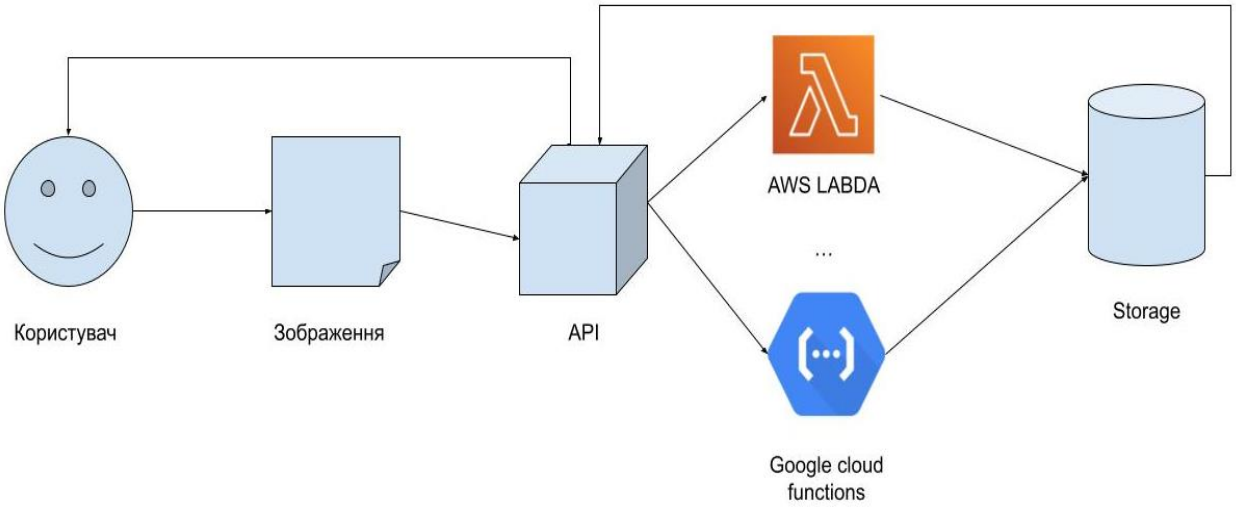


Рисунок В.3 – Структурна схема варіантів використання системи

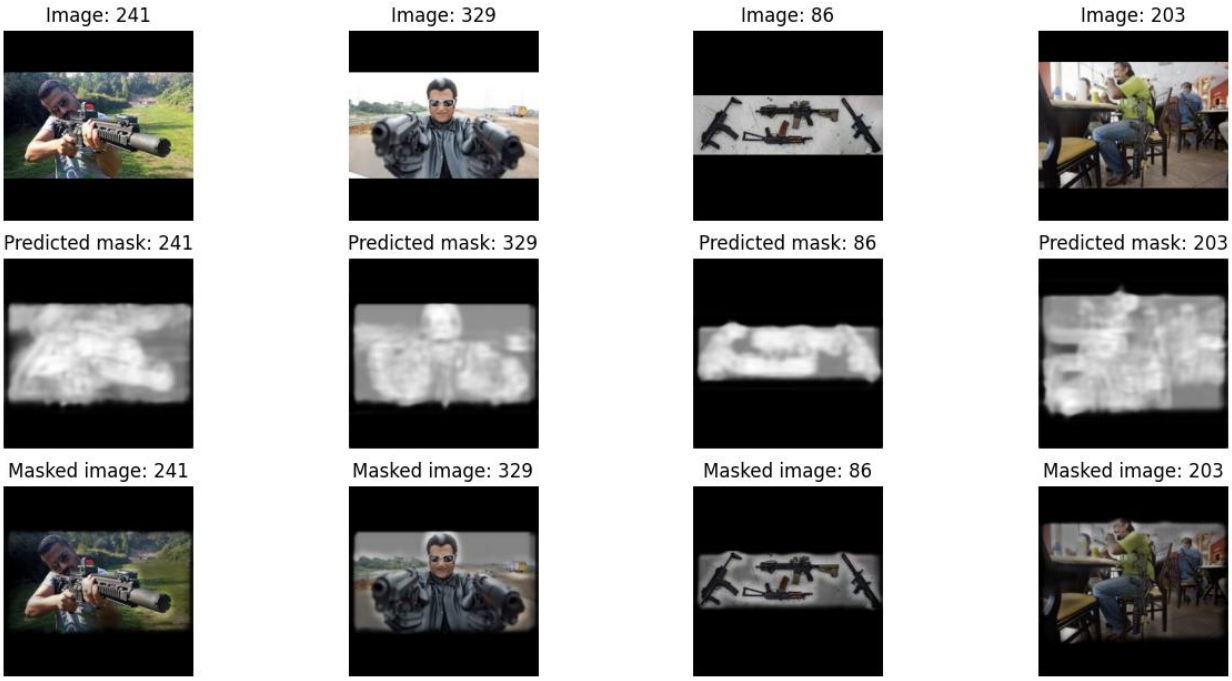


Рисунок В.4 Результати роботи програми

№ п.п.	Опис	Очікуваний результат	Отриманий результат
1	Введення даних, що вибиваються з ряду	Виведення результату, який не відповідає дійсності, проте є точним з огляду на введені данні	Виведення результату, який не відповідає дійсності, проте є точним з огляду на введені данні
2	Введення текстових даних замість зображень	Відповідне повідомлення про помилку в роботі програми	Відповідне повідомлення про помилку в роботі програми
3	Введення занадто великого зображення	Виведення результату, який не відповідає дійсності, проте є точним з огляду на введені данні	Виведення результату, який не відповідає дійсності, проте є точним з огляду на введені данні
4	Введення порожніх полів	Відповідне повідомлення про помилку в роботі програми	Відповідне повідомлення про помилку в роботі програми
55	Введення адекватних вхідних даних	Виведення результату	Виведення результату
6	Введення усіх полів порожніми	Відповідне повідомлення про помилку в роботі програми	Відповідне повідомлення про помилку в роботі програми

Рисунок В.5 результати тестування розробленої інформаційної системи фільтрації зображень

Додаток Г (довідниковий)

Інструкція користувача

Для запуску розробленої інформаційної системи фільтрації зображень слід відкрити виконавчий файл content_filtering.py, після чого встановити необхідні бібліотеки за допомогою команди «pip install». Після запуску програми слід обрати одне або декілька зображень для розпізнавання, підтримуються наступні формати: JPEG, PNG, BMP, JPG. Головне вікно програми зображено на рисунку Г.1.

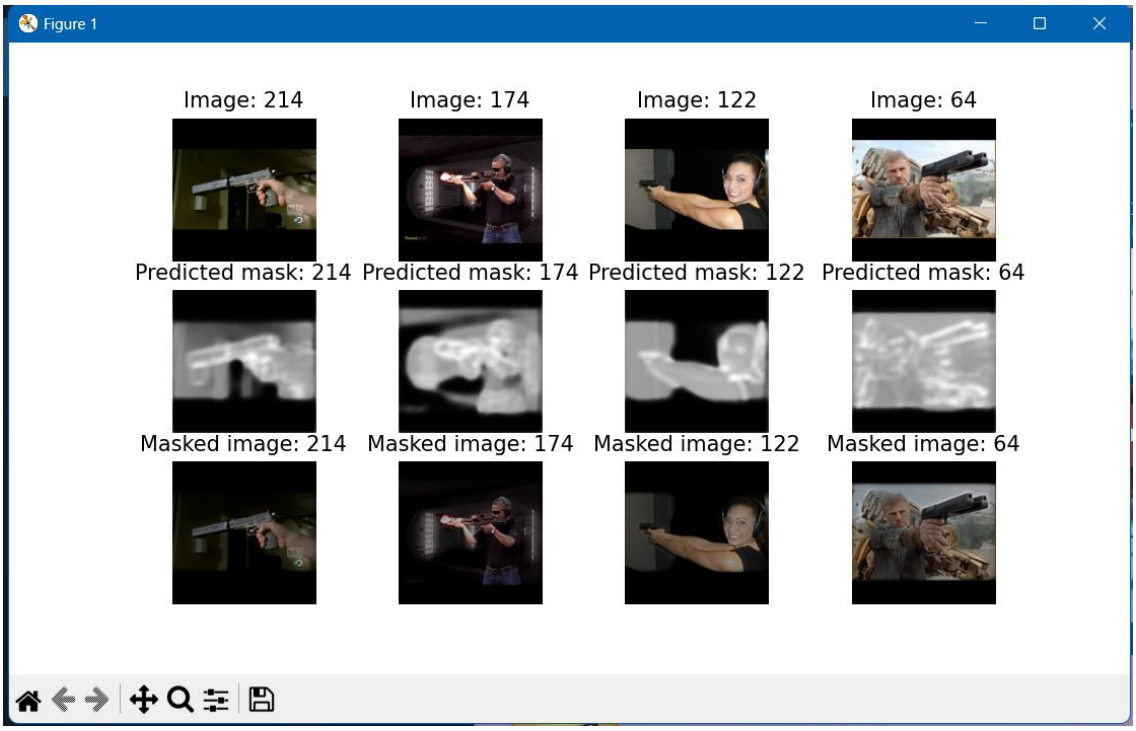


Рисунок Г.1 процес навчання нейронної мережі на новій навчальній вибірці

В результаті своєї роботи інформаційна система фільтрації зображень покаже: у 1 рядку вхідні зображення; у 2 рядку вхідні зображення з маскою; у 3 рядку вхідне зображення на якому контуром виділено заборонений вміст який розпізнала мережа.