Кафедра \_\_\_\_\_\_\_\_\_ електронних обчислювальних машин\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

**до бакалаврської кваліфікаційної роботи на тему**

\_\_\_\_\_Програмна платформа створення штучних нейронних мереж\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_A software platform for creating artificial neural networks\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент групи \_\_\_\_\_\_\_\_\_КІ-406 Ярмола Ю. Ю.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр, прізвище та ініціали)

|  |  |  |  |
| --- | --- | --- | --- |
| **Керівник роботи** |  |  | ( Олексів М. В. ) |
| **Консультанти** |  |  | ( ) |
|  |  |  | ( ) |
|  |  |  | ( ) |
|  |  |  | ( ) |
|  |  |  | ( ) |
|  |  |  | ( ) |

Завідувач кафедри \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ІКТА\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра\_електронних обчислювальних машин\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Спеціальність\_123 «Комп’ютерна інженерія»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«ЗАТВЕРДЖУЮ»

Завідувач кафедри ЕОМ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 р.

**ЗАВДАННЯ**

**на кваліфікаційну роботу (проект) студента групи \_КІ-406\_\_ОР \_\_\_бакалавр**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ярмола Юрій Юрійович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*прізвище,* *ім’я,* *по батькові*)

1. Тема роботи (проекту)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Програмна платформа створення штучних нейронних мереж \_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_A software platform for creating artificial neural networks \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(у разі виконання комплексної роботи в дужках вказується “комплексна робота (проект)”)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

затверджена наказом по університету від «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 р. № \_\_\_\_\_\_\_\_\_\_\_\_

1. Термін подання студентом закінченої роботи (проекту) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Вихідні дані для роботи (проекту)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Перелік графічного матеріалу

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Перелік програмних продуктів, які належить використати в процесі розроблення роботи (проекту) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7. Консультування роботи (проекту), із зазначенням розділів роботи

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Розділ | Консультант | Завдання видав | | Завдання прийняв | |
| підпис | дата | підпис | дата |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

8. Дата, коли видано завдання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Керівник\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
|  |  |  | (*підпис*) |  |
|  | Завдання прийняв до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
|  |  |  | (*підпис*) |  |
|  | КАЛЕНДАРНИЙ ПЛАН | | |  |
|  |  |  |  |  |
| №  з/п | Назва етапів роботи (проекту) | | Термін виконання  етапів роботи (проекту) | Примітка |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |
|  |  | |  |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*підпис*)

Керівник\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*підпис*)

# Розширена анотація

# Зміст

[Розширена анотація 4](#_Toc199283553)

[Зміст 5](#_Toc199283554)

[Вступ 7](#_Toc199283555)

[Розділ 1. АНАЛІТИЧНИЙ ОГЛЯД ТА ОБГРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗВ’ЯЗАННЯ ЗАДАЧІ 11](#_Toc199283556)

[1.1 Аналіз сучасних підходів до створення штучних нейронних мереж 11](#_Toc199283557)

[1.2 Аналіз відомих інструментів та платформ для побудови, навчання та оцінки моделей 12](#_Toc199283558)

[1.3 Аналіз методів розв’язання задачі 16](#_Toc199283559)

[1.3.1 Методи формування даних для тренування та валідації 16](#_Toc199283560)

[1.3.2 Методи навчання штучних нейронних мереж 17](#_Toc199283561)

[1.3.3 Методи перевірки моделі 17](#_Toc199283562)

[1.4 Основні режими функціонування платформи 18](#_Toc199283563)

[Висновки до розділу 1 20](#_Toc199283564)

[Розділ 2. ВИБІР ТА ОБГРУНТУВАННЯ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ 21](#_Toc199283565)

[2.1. Аналіз задачі та розробка вимог до програмної системи 21](#_Toc199283566)

[2.1.1 Функціональні вимоги 21](#_Toc199283567)

[2.1.2 Нефункціональні вимоги 22](#_Toc199283568)

[2.1.3 Вимоги до апаратного забезпечення 23](#_Toc199283569)

[2.2. Розробка структурної схеми програмного рішення 25](#_Toc199283570)

[2.3. Вибір мови програмування та технологій 27](#_Toc199283571)

[2.4. Засоби розробки програмного забезпечення 28](#_Toc199283572)

[2.5. Вибрані бібліотеки та фреймворки 30](#_Toc199283573)

[Висновки до розділу 2 32](#_Toc199283574)

[Розділ 3. РЕАЛІЗАЦІЯ АЛГОРИТМУ РОЗВ’ЯЗАННЯ ЗАДАЧІ 34](#_Toc199283575)

[3.1 Розробка алгоритму розв’язання задачі 34](#_Toc199283576)

[3.1.1 Метод формування структурованих даних 34](#_Toc199283577)

[3.1.2 Попередня обробка та аугментація даних 35](#_Toc199283578)

[3.1.3 Опис архітектури штучної нейронної мережі 36](#_Toc199283579)

[3.2 Реалізація програмного забезпечення 38](#_Toc199283580)

[3.2.1 Розробка архітектури програмного забезпечення 38](#_Toc199283581)

[3.2.2 Реалізація модуля генерації датасету 41](#_Toc199283582)

[3.2.3 Реалізація модуля навчання моделі 44](#_Toc199283583)

[3.2.4 Реалізація модуля перевірки моделі 47](#_Toc199283584)

[3.3 Інтеграція та сценарії використання 49](#_Toc199283585)

[3.3.1 Інтеграція алгоритму в кінцевий продукт 49](#_Toc199283586)

[3.4 Валідація 49](#_Toc199283587)

[3.4.1 Аналіз методів тестування та оцінювання 50](#_Toc199283588)

[3.4.2 Тестування програмного забезпечення 52](#_Toc199283589)

[Висновки до розділу 3 54](#_Toc199283590)

[ЕКОНОМІЧНА ЧАСТИНА 55](#_Toc199283591)

[4.1. Розрахунок витрат на розробку програмного забезпечення 55](#_Toc199283592)

[4.2 Кошторис витрат на розробку програмного забезпечення 59](#_Toc199283593)

[4.3 Визначення експлуатаційних витрат 59](#_Toc199283594)

[4.4 Розрахунок витрат на підготовку даних та реалізацію проектного рішення на ПК 61](#_Toc199283595)

[4.5. Розрахунок ціни споживання проектного рішення 62](#_Toc199283596)

[4.6. Визначення показників економічної ефективності 63](#_Toc199283597)

[Висновки до економічної частини 64](#_Toc199283598)

[Висновки 66](#_Toc199283599)

[Список літератури 67](#_Toc199283600)

[Додатки 68](#_Toc199283601)

# Вступ

**Актуальність роботи**

На сьогоднішній день штучні нейронні мережі (ШНМ) активно застосовуються у багатьох галузях науки, техніки та промисловості. Їх використання дозволяє вирішувати задачі, які раніше вважалися надскладними, наприклад, розпізнавання образів, обробка природної мови, прогнозування поведінки систем та автоматизація рутинних процесів. Розвиток технологій машинного навчання створив передумови для інтеграції ШНМ у програмні продукти, спрямовані на підвищення ефективності аналізу даних, оптимізації процесів та прийняття рішень.

Однак, створення якісних нейронних мереж вимагає значного обсягу обчислювальних ресурсів, глибокого розуміння теоретичних основ та практичних інструментів. У цьому контексті виникає необхідність у розробці універсальних платформ, які забезпечували б автоматизацію основних етапів роботи з ШНМ — від створення датасетів до візуалізації результатів. Це дозволить значно зменшити час на підготовку даних, оптимізувати процеси навчання моделей і підвищити доступність технологій штучного інтелекту для широкого кола користувачів.

Дослідження актуальне, оскільки забезпечує систематизацію підходів до розробки платформ, що працюють із ШНМ, та надає нові інструменти для ефективного впровадження штучного інтелекту в різні галузі.

**Мета і завдання дослідження**

Метою роботи є розробка платформи для автоматизації процесів створення, навчання та використання штучних нейронних мереж.

Для досягнення мети поставлено такі завдання:

1. Провести аналіз сучасних підходів та інструментів для роботи з ШНМ, визначити їх переваги та недоліки.
2. Розробити алгоритми автоматизації ключових етапів роботи з ШНМ, таких як підготовка даних, навчання моделей, перевірка точності та прогнозування.
3. Обґрунтувати вибір інструментів та технологій для реалізації платформи.
4. Розробити та впровадити платформу, яка дозволить ефективно використовувати можливості штучних нейронних мереж.
5. Провести тестування створеної платформи та оцінити її ефективність.

**Об’єкт та предмет дослідження**

Об’єктом дослідження є процеси автоматизації роботи з ШНМ. Предметом дослідження є методи та інструменти для створення універсальної платформи, яка дозволяє автоматизувати ключові етапи розробки та використання нейронних мереж.

**Методологія дослідження**

Для досягнення поставленої мети використано такі методи дослідження:

1. Аналітичний метод для вивчення наукових публікацій, технічної документації та сучасних інструментів у галузі машинного навчання.
2. Емпіричний метод для розробки алгоритмів, проведення експериментів та тестування моделі.
3. Метод моделювання для створення і навчання ШНМ із використанням спеціалізованих інструментів.
4. Метод системного аналізу для оцінки ефективності створеної платформи.
5. Методи візуалізації для аналізу результатів роботи моделі та підготовки звітів.

**Практичне значення**

Практична цінність роботи полягає у розробці універсальної платформи, яка дозволяє значно спростити процеси створення та використання ШНМ. Реалізована система може використовуватись для аналізу даних, побудови прогнозів, автоматизації рутинних процесів та вирішення прикладних задач у галузях, де потрібна висока точність та швидкість обробки інформації.

Платформа забезпечує:

* інтерактивний інтерфейс для створення та оновлення датасетів;
* автоматизацію процесу навчання моделей;
* оцінку точності моделей на тестових наборах даних;
* прогнозування з використанням навченої моделі;
* зручну візуалізацію результатів для прийняття рішень.

Розроблена платформа може бути інтегрована у різні галузі, такі як медицина, фінанси, освіта, промисловість тощо.

**Структура дипломної роботи**

Дипломна робота складається з вступу, п’яти розділів, висновків, списку використаних джерел та додатків.

* У першому розділі проведено аналітичний огляд літератури та обґрунтовано актуальність дослідження.
* У другому розділі описано вибір інструментів та методів розробки платформи.
* У третьому розділі детально описано розробку, тестування та оцінку ефективності платформи.
* У четвертому розділі розглянуто економічну ефективність впровадження платформи.

Робота містить аналітичний огляд, технічні рішення, результати тестування та рекомендації для подальшого використання розробленої платформи.

# Розділ 1. АНАЛІТИЧНИЙ ОГЛЯД ТА ОБГРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗВ’ЯЗАННЯ ЗАДАЧІ

За останні кілька років у світі різко зріс інтерес до штучних нейронних мереж (ШНМ). Це можна пояснити виходом зрозумілим для більшості людей нейронної мережі у вигляді чату, що відповідає на будь-які запитання, генерує та розпізнає зображення, виконує як прості так і складні задачі. Цей інтерес також спровокував розвиток і інших спеціалізованих нейронних мереж, які здатні виконувати добре одну вузьку задачу, наприклад генерація або розпізнавання зображень. Проте у всіх цих ШНМ у основі роботи лежить принцип імітації нейрону нервової клітини людського мозку, який має різні реалізації, продуктивність та інші параметри які впливають на навчання та роботу готової мережі.

## 1.1 Аналіз сучасних підходів до створення штучних нейронних мереж

Основні принципи створення штучних нейронних мереж базуються на моделюванні подібних на біологічні нейрони мереж, що дають змогу обробляти великі обсяги даних та знаходити складні залежності у них. Штучні нейронні мережі мають різні архітектури для моделювання нейронів, ось найпопулярніші з них:

* Багатошарові перцептрони

Основний блок для багатьох інших нейронних мереж. Використовується для бінарної класифікації.

* Згорткові нейронні мережі

Ефективні для обробки зображень і використовують згорткові шари для виявлення патернів у зображеннях.

* Рекурентні нейронні мережі

Призначені для обробки послідовних даних, таких як текст або часові ряди.

* Мережі асоціативної пам’яті

Використовуються для розв'язання завдань асоціативної пам'яті і оптимізації.

* Трансформери

Використовуються для обробки послідовних даних та роботи з прикладами з різних контекстів.

Методи навчання включають підконтрольне навчання для прогнозів, безконтрольне для кластеризації та навчання з підкріпленням для прийняття рішень. Для оптимізації використовуються алгоритми, такі як градієнтний спуск та його модифікації (Adam, RMSprop), що дозволяють зменшити функцію втрат і покращити точність моделі.

Популярні платформи для реалізації ШНМ, як TensorFlow, PyTorch та Keras, забезпечують інструменти для швидкого створення, навчання і тестування моделей. Водночас вони мають обмеження, зокрема потребу у великих обчислювальних ресурсах і залежність від якості даних.

## 1.2 Аналіз відомих інструментів та платформ для побудови, навчання та оцінки моделей

Розробка, навчання та оцінка моделей є складною задачею, яка вимагає спеціалізованих інструментів чи платформ, які забезпечують високу продуктивність роботи з даними, підготовка чи створення даних та гнучкість у розробці чи дослідженні ШНМ. Найпопулярніші платформи розраховані зазвичай на програмістів, що мають досвід у розробці ШНМ, хоча є і платформи, які розраховані на дослідження чи базові налаштування через зручний GUI. Вибір відповідного інструменту залежить від типу задачі, обсягу даних та вимог до обчислювальних ресурсів.

* TensorFlow

Відкрита програмна бібліотека для машинного навчання цілій низці задач, розроблена компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявляння та розшифровування образів та кореляцій, аналогічно до навчання й розуміння, які застосовують люди.

* PyTorch

Відкрита бібліотека машинного навчання на основі бібліотеки Torch, що застосовується для задач комп'ютерного зору та обробки природної мови.

* Keras

Відкрита нейромережна бібліотека, написана мовою Python. Вона здатна працювати поверх TensorFlow, Microsoft Cognitive Toolkit, R, Theano та PlaidML. Спроектовану для уможливлення швидких експериментів з мережами глибокого навчання, її зосереджено на тому, щоби вона була зручною в користуванні, модульною та розширюваною.

* Transformers

Бібліотека попередньо навчених моделей обробки природної мови, комп'ютерного зору, аудіо та мультимодальних моделей для логічного висновку та навчання.

* Roboflow

Комерційна платформа для розпізнавання об’єктів на відео та експериментів з навчанням моделей комп’ютерного зору. Особливо платформа корисна для роботи з власними наборами зображень

Кожна розроблена платформа має свої сильні та слабкі сторони, які потрібно враховувати при проектуванні платформи розробки ШНМ. Нижче наведені недоліки уже реалізованих рішень.

**TensorFlow**

TensorFlow є однією з найпотужніших відкритих бібліотек для реалізації нейронних мереж, розробленою компанією Google. Проте, незважаючи на високу гнучкість та широкі можливості, бібліотека характеризується наступними недоліками:

* Висока складність засвоєння, що ускладнює використання її користувачами без відповідної підготовки;
* Відсутність вбудованих інструментів для анотації зображень та управління наборами даних;
* Необхідність інтеграції з додатковими рішеннями для забезпечення повного циклу розробки моделей комп’ютерного зору.
* Не можливість використовувати повні можливості через CUDA на операційній системі Windows.

**PyTorch**

PyTorch — ще одна популярна бібліотека для розробки моделей машинного навчання, яка вирізняється підтримкою динамічного графа обчислень та активно використовується у наукових дослідженнях. Однак, її використання супроводжується такими обмеженнями:

* Необхідність глибоких знань програмування для реалізації більшості завдань;
* Відсутність інструментів для створення, анотації та попередньої обробки наборів зображень у вбудованому вигляді;
* Значні трудові витрати на початкових етапах проекту.

**Keras**

Keras є високорівневою нейронною бібліотекою, що працює поверх інших фреймворків, таких як TensorFlow чи Theano. Основною перевагою Keras є простота у використанні. Водночас, до недоліків можна віднести:

* Орієнтацію виключно на етап побудови та навчання моделей, без підтримки інструментів для керування наборами зображень;
* Обмежену функціональність у контексті розв’язання прикладних задач комп’ютерного зору без сторонніх засобів.

**Transformers**

Бібліотека Transformers, створена компанією Hugging Face, містить велику кількість попередньо навчених моделей для задач обробки природної мови, комп’ютерного зору, аудіо тощо. Незважаючи на переваги, бібліотека має наступні недоліки:

* Основний фокус зосереджено на обробці текстової інформації, тоді як можливості для комп’ютерного зору є обмеженими;
* Відсутність засобів для повноцінної роботи з власними зображеннями без попередньої підготовки даних.

**Roboflow**

Roboflow є сучасною хмарною платформою для реалізації проектів у галузі комп’ютерного зору. Її головна перевага — це зручність роботи з наборами зображень, включно з їх анотацією, попередньою обробкою та генерацією нових даних. Однак, навіть при значних перевагах, платформа має низку обмежень:

* Обмеження функціональності у безкоштовній версії, що ускладнює масштабування проекту;
* Тренування моделей відбувається на сторонніх серверах, що може створити загрози безпеці при роботі з конфіденційною інформацією;
* Обмежена гнучкість налаштування моделей порівняно з відкритими бібліотеками низького рівня.

## 1.3 Аналіз методів розв’язання задачі

У цьому підрозділі розглядаються основні підходи та методи, що застосовуються у задачах створення, навчання та оцінювання штучних нейронних мереж (ШНМ). Розуміння принципів їх роботи є необхідною основою для подальшої розробки та реалізації відповідної програмної платформи.

### 1.3.1 Методи формування даних для тренування та валідації

Якість навчання штучної нейронної мережі значною мірою залежить від якості вхідних даних. У науковій та практичній літературі сформовано усталений підхід до створення навчальних вибірок, який включає наступні ключові етапи:

1. Збір даних. Джерелами інформації можуть бути відкриті набори даних, користувацькі записи або власноруч зібрані дані. Типи даних варіюються залежно від задачі: зображення, текст, аудіо тощо.
2. Анотація. Передбачає маркування зібраної інформації. Для зображень — це визначення координат об’єктів або присвоєння класів.
3. Попередня обробка. Застосовується для нормалізації та очищення даних: зміна розмірів, форматів, фільтрація шумів, видалення зайвого.
4. Аугментація. Метод штучного збільшення кількості прикладів за допомогою перетворень: обертання, масштабування, дзеркалювання тощо.
5. Розподіл на вибірки. Дані поділяються на навчальну, валідаційну та тестову вибірки, що забезпечує можливість точного навчання та оцінки.

### 1.3.2 Методи навчання штучних нейронних мереж

Процес навчання ШНМ є базовим етапом у побудові моделей штучного інтелекту. У сучасних дослідженнях найбільш поширеним є підхід із використанням зворотного поширення помилки та градієнтного спуску. Основні кроки:

1. Ініціалізація. Визначення архітектури мережі та початкових значень вагів.
2. Функція втрат і оптимізатор. Вибір метрики, яка відображає помилку, та способу її мінімізації (наприклад, Adam, SGD).
3. Пряме проходження (forward pass). Обчислення вихідного сигналу на основі вхідних даних.
4. Обчислення помилки. Визначення відхилення результату від очікуваного.
5. Зворотне поширення помилки (backpropagation). Розрахунок похідних функції втрат для оновлення вагів.
6. Оновлення вагів. Застосування методів оптимізації.
7. Багаторазове повторення. Процес виконується багаторазово (епохи) для покращення результатів.
8. Валідація. Оцінка проміжного результату на валідаційній вибірці для виявлення перенавчання.
9. Фінальна оцінка. Після завершення навчання модель перевіряється на тестовій вибірці.

### 1.3.3 Методи перевірки моделі

Оцінка ефективності моделі — завершальний етап у загальній схемі роботи з нейронною мережею. Основні етапи перевірки включають:

1. Формування тестової вибірки. Створюється до початку навчання для забезпечення об'єктивності.
2. Завантаження навченої моделі. Модель, збережена після навчання, використовується без повторної оптимізації.
3. Генерація передбачень. На основі вхідних даних модель генерує результати.
4. Порівняння з еталонними значеннями. Дає змогу оцінити точність.
5. Обчислення метрик. Наприклад, точність, повнота, F1-міра — залежно від типу задачі.
6. Аналіз результатів. У разі незадовільної точності можливий перегляд підходу до навчання або структури мережі.

## 1.4 Основні режими функціонування платформи

Відомі рішення для розв’язання задачі створення, навчання та використання моделей штучного інтелекту передбачають поділ роботи платформи на кілька основних функціональних режимів. Ці режими спрямовані на забезпечення зручності використання, оптимізації процесів навчання, тестування та застосування моделей. У рамках аналізу літератури було виявлено, що подібні платформи зазвичай функціонують у таких основних режимах:

1. **Створення та оновлення датасету**   
   Більшість платформ передбачає можливість збору, обробки та структурування даних. Відомі підходи включають автоматичне очищення та анотацію, що дозволяє зменшити трудомісткість цього процесу. Однак у деяких випадках користувачі також мають змогу вручну коригувати анотації для забезпечення більшої точності.
2. **Навчання моделі**   
   Цей режим зазвичай включає механізми автоматизованого налаштування гіперпараметрів та оптимізації ваг нейронної мережі. У науковій літературі широко використовуються фреймворки, такі як TensorFlow та PyTorch, які надають інструменти для гнучкого налаштування та навчання моделей.
3. **Перевірка точності моделі**   
   Оцінка якості побудованих моделей здійснюється за допомогою метрик, таких як точність, повнота, F1-міра тощо. Деякі платформи також надають інструменти для виявлення помилкових передбачень, що сприяє подальшому вдосконаленню моделей.
4. **Прогнозування**   
   Інференс є ключовою функцією платформ, яка дозволяє застосовувати навчені моделі для обробки нових даних. Сучасні рішення часто оптимізують цей процес для прискорення роботи, зокрема шляхом використання апаратних прискорювачів (наприклад, GPU або TPU).
5. **Візуалізація результатів**   
   Для підвищення зручності користувачів платформи зазвичай інтегрують засоби візуалізації, такі як графіки, таблиці або інтерактивні інтерфейси для перегляду результатів роботи моделі. Деякі платформи надають можливість створення звітів у різних форматах, що є важливим для аналізу ефективності.

Подальші кроки у розв’язанні задачі розробки платформи передбачають аналіз існуючих підходів до кожного з цих режимів, вибір найбільш ефективних методів і їх адаптацію до специфічних вимог завдання. Усі ці компоненти будуть інтегровані в архітектуру системи, щоб забезпечити її ефективність і функціональність.

## Висновки до розділу 1

У першому розділі було здійснено аналітичний огляд літератури та існуючих підходів до розв’язання задачі створення, навчання та використання моделей штучного інтелекту. Розглянуто типові функціональні режими роботи платформ, які дозволяють реалізувати повний цикл обробки даних і роботи з моделями: створення та оновлення датасету, навчання моделі, перевірка точності, прогнозування та візуалізація результатів.

Аналіз показав, що основними напрямами розвитку таких платформ є автоматизація процесів, підвищення точності моделей та забезпечення зручності для користувачів. Разом з тим, кожен із розглянутих підходів має свої переваги та недоліки, які необхідно враховувати при розробці власного рішення.

Подальші кроки дослідження передбачають визначення конкретних вимог до функціональності платформи, вибір інструментів і технологій для її реалізації, а також розробку алгоритмів і архітектури системи, що враховуватиме особливості вирішуваного завдання.

# Розділ 2. ВИБІР ТА ОБГРУНТУВАННЯ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ

У даному розділі здійснюється всебічний аналіз та обґрунтований вибір інструментів, технологій і середовищ, що використовуються для реалізації програмної платформи. Розглядаються можливі альтернативи, порівнюються їхні функціональні можливості, переваги та недоліки з урахуванням вимог до системи. На основі проведеного аналізу формується аргументований вибір оптимальних засобів, які забезпечать ефективну, масштабовану та зручну у підтримці реалізацію програмного забезпечення.

## 2.1. Аналіз задачі та розробка вимог до програмної системи

Основною задачею у даній роботі є розробка програмної платформи, здатної автоматизовано формувати анотований датасет та навчати ним різні ШНМ з мінімальним втручанням у код. Розроблена система має забезпечувати повний цикл — від збору та анотації зображень до їх завантаження у модель, а також виконання класифікації на основі отриманих результатів. Такий підхід дозволяє значно спростити підготовку даних та забезпечити ефективне використання ШНМ для вирішення задач комп’ютерного зору.

### 2.1.1 Функціональні вимоги

До основних функціональних можливостей системи належать:

* **Автоматизоване створення датасету** — система повинна забезпечити формування вибірки зображень із відповідною структурою (у форматі папок з анотаціями), що підтримується більшістю фреймворків глибокого навчання.
* **Підтримка популярних типів відео** – реалізувати підтримку популярних типів відео таких як mp4, avi, mov, WebM.
* **Анотування зображень** — реалізувати привласнення кожному зображенню певної категорії для класифікації.
* **Механізм навчання ШНМ** — використання згорткової нейронної мережі, зокрема архітектури на базі torch.nn.Sequential, яка включає згорткові шари (Conv2d), пулінг (MaxPool2d) та повнозв'язні шари (Linear), є необхідною функціональністю для навчання моделі на підготовленому датасеті.
* **Класифікація нових зображень** — система повинна забезпечити можливість класифікації зображень, що не входили до навчальної вибірки, з використанням збереженої моделі.

### 2.1.2 Нефункціональні вимоги

**Обсяг і тип оброблюваних даних**

Система має оперувати з зображеннями, представленими у форматах PNG, JPEG тощо, з можливістю масштабування до вказаного розміру (наприклад, 32x32 пікселі). Очікується обробка наборів даних, що складаються з десятків тисяч зображень, з чіткою структурою за класами, що дає змогу ефективно застосовувати механізми автоматичної обробки.

**Вимоги до продуктивності**

* Навчання ШНМ має відбуватися із максимальною ефективністю: використання графічного процесора (GPU) за допомогою CUDA.
* Процес класифікації одного зображення повинен займати не більше 3 секунд.
* Необхідно забезпечити можливість збереження ваг моделі та їх подальшого завантаження без повторного навчання.

**Вимоги до масштабованості**

* Архітектура системи має бути придатною до горизонтального масштабування. Наприклад, можлива інтеграція з хмарними сервісами (Google Colab, AWS, Azure) для розширення обчислювальних можливостей.
* Враховуючи можливість генерації довільної кількості зображень, важливо передбачити механізм автоматичного розподілу навантаження під час генерації або навчання.

### 2.1.3 Вимоги до апаратного забезпечення

Для забезпечення коректної, ефективної та стабільної роботи програмного забезпечення, яке реалізує автоматичне формування анотованих датасетів та класифікацію зображень за допомогою методів машинного навчання, висуваються певні вимоги до апаратного забезпечення. Ці вимоги залежать від обсягів оброблюваних даних, складності моделей, а також частоти запуску операцій тренування та тестування.

**Мінімальні вимоги:**

Мінімальні характеристики дозволяють запускати систему в обмеженому режимі — з невеликим датасетом, спрощеною архітектурою моделі або використанням попередньо навченої нейромережі.

* Процесор (CPU): 4-ядерний (Intel Core i5 або AMD Ryzen 5)
* Оперативна пам’ять (RAM): 8 ГБ
* Накопичувач (HDD/SSD): 50 ГБ вільного простору на диску (рекомендується SSD)
* Графічна підсистема (GPU): Необов’язково, але рекомендовано для пришвидшення обробки зображень

У разі відсутності графічного процесора, моделі на базі PyTorch можуть бути запущені на CPU, однак час навчання значно зростає, що робить систему менш продуктивною при роботі з великими обсягами даних.

**Рекомендовані вимоги:**

Для повноцінного функціонування системи — з можливістю обробки великих наборів зображень, тренування глибоких нейронних мереж та візуалізації процесів — рекомендуються такі характеристики:

* Процесор (CPU): 6–8 ядер (Intel Core i7, AMD Ryzen 7 або еквівалент)
* Оперативна пам’ять (RAM): 16–32 ГБ
* Накопичувач (SSD): не менше 100 ГБ вільного простору
* Графічна підсистема (GPU): NVIDIA з підтримкою CUDA (наприклад, GeForce RTX 3060 або вище) — для ефективного тренування моделей у PyTorch
* Операційна система: Windows 10/11, Ubuntu 20.04+ або інша сучасна ОС, що підтримує бібліотеки машинного навчання

**Специфічні вимоги до середовища розробки:**

* IDE: JetBrains PyCharm Professional або Community Edition — для зручної роботи з Python-кодом, налагодженням та інтеграцією з віртуальними середовищами
* Підтримка Python 3.9+, сумісність з бібліотеками OpenCV, PyTorch, Pillow тощо

Таким чином, відповідність апаратного забезпечення рекомендованим характеристикам дозволяє досягти високої продуктивності, скоротити час тренування моделей та забезпечити зручну взаємодію користувача з інтерфейсом системи. У майбутньому, за умови розширення функціональності та масштабування обробки даних, ці вимоги можуть бути переглянуті у бік збільшення.

## 2.2. Розробка структурної схеми програмного рішення

Для забезпечення ефективної реалізації програмної системи було проведено аналіз кількох варіантів архітектурного проектування. Серед основних кандидатів розглядалися монолітна, мікросервісна та модульна архітектура. З урахуванням специфіки завдань, масштабів проекту та вимог до продуктивності — було прийнято рішення на користь модульної архітектури.

* Монолітна архітектура, попри простоту реалізації, характеризується тісним зчепленням компонентів, що ускладнює підтримку коду, модифікацію окремих функціональних блоків і тестування. У разі розширення системи або оновлення певного функціоналу можуть виникати труднощі, пов’язані з необхідністю зміни всього застосунку.
* Мікросервісна архітектура передбачає розбиття програми на окремі сервіси, кожен з яких працює незалежно. Цей підхід є потужним і масштабованим, проте вимагає складної інфраструктури — зокрема, налаштування взаємодії між сервісами, організації мережевих запитів, а також систем моніторингу та логування. З огляду на обмежений обсяг задач та відсутність потреби у високій розподіленості системи, впровадження мікросервісної архітектури не є доцільним для даного проекту.
* Модульна архітектура, яка була обрана для реалізації, полягає у логічному поділі програмного забезпечення на незалежні функціональні блоки (модулі), які взаємодіють між собою через чітко визначені інтерфейси. У межах цієї архітектури кожен модуль відповідає за окремий етап обробки даних або виконання функціоналу. Такий підхід забезпечує високу гнучкість, спрощує тестування, дозволяє ізолювати помилки та реалізовувати зміни без впливу на інші компоненти системи.

У даному програмному рішенні виділено кілька основних модулів:

* Модуль генерації даних — відповідає за автоматичне створення зображень із заданими параметрами та їх анотування (наприклад, розміщення геометричних фігур, накладення міток тощо).
* Модуль формування датасету — агрегує згенеровані зображення та створює структуру, придатну для подальшого навчання моделі (наприклад, директорії з класами).
* Модуль навчання ШНМ — здійснює побудову та тренування штучної нейронної мережі на сформованому датасеті, з використанням бібліотеки PyTorch, збереження ваг моделі.
* Модуль класифікації — використовує навчену модель для передбачення класу нових вхідних зображень.
* Допоміжні утиліти (utils) — включають функції візуалізації, завантаження зображень, обробки результатів тощо.



Рис. 2.1 – Структурна схема архітектури програмного рішення

Такий підхід дозволяє розробнику швидко адаптувати систему до нових вимог. Наприклад, у разі зміни архітектури моделі нейронної мережі або бажання використовувати інші дані — достатньо змінити лише відповідний модуль, не зачіпаючи решту системи.

У підсумку, модульна архітектура є найзбалансованішим рішенням для проекту: вона забезпечує достатню продуктивність, спрощує масштабування системи в майбутньому та значно полегшує підтримку й розвиток коду.

## 2.3. Вибір мови програмування та технологій

Вибір мови програмування є критично важливим етапом розробки платформи, оскільки він визначає гнучкість, продуктивність та зручність підтримки системи. У цьому проєкті основною мовою програмування обрано Python. Основними критеріями вибору були: простота та читабельність коду, наявність потужної екосистеми для роботи з даними та навчання моделей штучного інтелекту, кросплатформеність, активна підтримка спільноти та доступність навчальних матеріалів.

Python був обраний завдяки своїй високій читабельності, широкій підтримці бібліотек для аналізу даних (NumPy, Pandas), глибокого навчання (TensorFlow, PyTorch) та візуалізації результатів (Matplotlib, Seaborn). Мова забезпечує зручність інтеграції з іншими технологіями, регулярне оновлення інструментів та масштабованість. Незважаючи на нижчу продуктивність порівняно з компільованими мовами (наприклад, C++ чи Java), це компенсується використанням оптимізованих бібліотек, що дозволяє ефективно виконувати задачі.

Python активно використовується в наукових та дослідницьких колах, що обумовлено наявністю перевірених бібліотек, таких як:

* NumPy — для виконання чисельних обчислень та роботи з багатовимірними масивами;
* Matplotlib та Pillow — для візуалізації, генерації зображень та обробки графічних даних;
* PyTorch — як основний фреймворк для побудови та тренування штучної нейронної мережі, що надає гнучкий інтерфейс, динамічне обчислення графу та підтримку GPU-прискорення.

Крім того, Python забезпечує просту інтеграцію між модулями, можливість швидкого прототипування, активну спільноту розробників, що пришвидшує вирішення потенційних проблем, а також хорошу підтримку сучасних інструментів для аналізу та обробки зображень.

Для організації проектної структури та розділення відповідальностей використовувалися засоби модульного програмування. В результаті код був поділений на логічні компоненти, кожен з яких відповідає за конкретну функціональність: генерація датасету, анотація, навчання моделі та класифікація зображень.

З точки зору сумісності з іншими платформами та розширюваності, Python дозволяє легко інтегруватися з іншими сервісами, підтримує міжплатформенну розробку та забезпечує можливість подальшого перенесення моделі на сервер або хмарну платформу (наприклад, для інференсу в режимі реального часу).

Таким чином, обрана мова програмування та набір технологій повністю відповідають вимогам проекту — як з точки зору реалізації, так і подальшої підтримки, розширення та масштабування.

## 2.4. Засоби розробки програмного забезпечення

У процесі реалізації програмного забезпечення важливо забезпечити комфортні умови для розробки, тестування, налагодження та супроводу системи. Для цього використовуються спеціалізовані інструменти, що сприяють підвищенню ефективності праці, структурованості коду та якості кінцевого продукту. У даному проекті обрано низку засобів, які відповідають цим критеріям.

* **Середовище розробки**

Основним інструментом для написання та налагодження коду є інтегроване середовище розробки PyCharm від компанії JetBrains. Воно забезпечує зручне автодоповнення коду, інтеграцію з системами контролю версій, підтримку віртуальних середовищ, інструменти для тестування та відлагодження, а також має широкі можливості для роботи з Python-бібліотеками, що використовуються у проекті. Крім того, PyCharm надає інструменти для аналізу якості коду та рефакторингу, що позитивно впливає на підтримку коду в довгостроковій перспективі.

* **Система контролю версій**

Для керування версіями програмного коду застосовувалася система Git. Усі зміни відслідковуються та документуються за допомогою репозиторію, розміщеного на платформі GitHub, що дозволяє зберігати історію змін, працювати з гілками та спрощує співпрацю в команді. Git також забезпечує захист від втрати даних та можливість відкату до попередніх стабільних версій коду.

* **Віртуальне середовище**

Для ізоляції залежностей було створено віртуальне середовище за допомогою venv, яке дозволяє уникнути конфліктів між бібліотеками, що використовуються в проекті. Всі зовнішні пакети, необхідні для виконання функціоналу, встановлюються локально в межах цього середовища. Це підвищує стабільність роботи системи та забезпечує однакові умови запуску на різних пристроях.

* **Бібліотеки для візуалізації та обробки зображень**

Під час реалізації було використано бібліотеки Matplotlib, Pillow (PIL) та інші засоби для обробки та візуалізації зображень. Вони надали широкі можливості для перегляду, анотування, попередньої обробки даних та виводу результатів роботи моделі.

* **Інструменти для тестування та налагодження**

PyCharm має вбудовану підтримку засобів для покрокового виконання коду (debugging), а також дозволяє швидко запускати окремі фрагменти коду для перевірки логіки. Це дозволяє ефективно відслідковувати помилки, працювати з точками зупину та змінними під час виконання, що значно пришвидшує цикл тестування.

У сукупності обрані інструменти дозволили забезпечити структурований, надійний та масштабований підхід до реалізації програмної системи, що відповідає сучасним вимогам до якості розробки інтелектуального програмного забезпечення.

## 2.5. Вибрані бібліотеки та фреймворки

У процесі розробки системи були обрані різноманітні бібліотеки та фреймворки для забезпечення високої ефективності, зручності у використанні та адаптивності. Вибір цих інструментів був обумовлений особливостями завдання, зокрема необхідністю обробки зображень, побудови та тренування моделей для класифікації, а також забезпеченням гнучкості у навчанні.

* **PyTorch –** Бібліотека для реалізації машинного навчання, яка є однією з найбільш популярних бібліотек для побудови та тренування моделей глибокого навчання. PyTorch має високу гнучкість, підтримує динамічні обчислювальні графи, що особливо зручно для експериментів, і надає потужні засоби для роботи з нейронними мережами. Цей фреймворк дозволяє значно спростити процес тренування та тестування моделей, а також інтеграцію з різними типами даних.
* **OpenCV -** Для обробки зображень на етапі підготовки даних. Вона є стандартним інструментом для комп'ютерного зору та має безліч алгоритмів для обробки та аналізу зображень, що є необхідним при роботі з візуальними даними. OpenCV забезпечує такі можливості, як зміна розміру зображень, застосування фільтрів, виявлення контурів та інші важливі функції для попередньої обробки.
* **Pillow (PIL) -** Бібліотека, яка є форком Python Imaging Library (PIL), використовувалась для базових операцій з обробки зображень, таких як зчитування, збереження, обрізка та інші операції. Ця бібліотека дозволяє ефективно працювати з різними форматами зображень та виконувати операції, необхідні для створення анотованих датасетів.
* **NumPy -** Для роботи з великими масивами даних, виконання математичних операцій і векторизації алгоритмів. Вона є основним інструментом для обчислень в Python, особливо при обробці числових даних, і забезпечує високу швидкість обчислень завдяки вбудованим функціям для роботи з масивами.
* **Matplotlib -** Для візуалізації результатів класифікації, процесу навчання моделі, графічного представлення метрик точності та втрат. Вона дозволяє будувати графіки, гістограми, діаграми і наочно демонструвати результати тренування моделей, що полегшує аналіз та вдосконалення роботи моделі.
* **Tkinter -** Для створення інтерфейсу користувача (GUI). Він забезпечує простоту розробки віконних додатків, які дозволяють зручно взаємодіяти з користувачем. За допомогою Tkinter був реалізований інтерфейс для завантаження зображень, запуску процесу навчання моделі, а також для перегляду результатів класифікації.

Таким чином, вибір цих бібліотек та фреймворків був обумовлений їх здатністю підтримувати всі етапи обробки даних — від підготовки та анотування зображень до тренування моделей машинного навчання та виведення результатів. Обрані інструменти забезпечують високу продуктивність, гнучкість та можливість масштабування, що є важливим для подальшого вдосконалення системи.

## Висновки до розділу 2

У другому розділі було здійснено детальний аналіз технічних вимог та обґрунтування вибору архітектурних і технологічних рішень, необхідних для розробки програмної платформи. На основі аналізу функціональних і нефункціональних вимог визначено вимоги до апаратних ресурсів.

Було обґрунтовано вибір модульної архітектури, яка завдяки своїй гнучкості та масштабованості дозволяє ефективно інтегрувати нові компоненти, забезпечує зрозумілу логіку взаємодії між модулями, а також сприяє зручності супроводу та тестування.

Розробка платформи проводиться мовою Python. Це обумовлено її широким застосуванням у галузі машинного навчання, наявністю численних бібліотек (зокрема, PyTorch, OpenCV, Pillow) і сумісністю з сучасними інструментами розробки, такими як PyCharm. Вибір бібліотек здійснюється з урахуванням їхньої функціональності, продуктивності та підтримки сучасних підходів до обробки зображень і побудови моделей ШНМ.

Також сформульовано апаратні вимоги до системи, що враховують можливість запуску на мінімальній конфігурації, а також орієнтовані на ефективну роботу із залученням GPU для прискорення процесів навчання моделей.

Загалом, прийняті рішення створюють міцну технічну основу для подальшої реалізації функціональності системи, її розвитку та адаптації до змін у вимогах користувача або середовища застосування.

# Розділ 3. РЕАЛІЗАЦІЯ АЛГОРИТМУ РОЗВ’ЯЗАННЯ ЗАДАЧІ

## 3.1 Розробка алгоритму розв’язання задачі

Для розробки ШНМ найважливішим елементом є набір даних для навчання та тестування нейронної мережі, адже від якості та кількості цих даних буде залежати ефективність роботи створеної моделі. Саме тому варто виділити цей етап як один із головних при розробці ШНМ, що буде висвітлено у цьому розділі.

### 3.1.1 Метод формування структурованих даних

На етапі формування даних передбачено створення базового набору, який використовуватиметься для навчання, валідації та тестування штучної нейронної мережі. Створення алгоритму, який дозволятиме отримувати власну вибірку передбачає повний контроль над результатом та актуальністю отриманої моделі для конкретних потреб клієнта.

Процес формування даних включає кілька важливих підетапів:

1. Збір даних - планується створення навчального набору даних шляхом аналізу відеофайлів, наданих користувачем. Для цього передбачається використання алгоритмів трекінгу, які дозволять автоматично відстежувати заданий об'єкт на відео. Такий підхід забезпечить формування якісного і контрольованого набору даних, що відповідатиме конкретним вимогам проекту.
2. Структуризація - дані планується упорядкувати у форматі, зручному для обробки штучною нейронною мережею. Зокрема, кожне зображення буде доповнене міткою, яка визначає його класову належність. Усі дані будуть структуровані та розподілені за категоріями для забезпечення їх ефективного використання на етапах навчання та тестування моделі.
3. Розподіл наборів - для забезпечення коректності навчання та оцінки моделі дані були розділені на три підмножини:

* Навчальний набір - використовується для безпосереднього навчання моделі.
* Валідаційний набір - забезпечує перевірку якості моделі під час навчання, допомагаючи уникнути перенавчання.
* Тестовий набір - призначений для остаточної оцінки точності та узагальнювальної здатності моделі.

Таке структурування даних дозволяє забезпечити об'єктивність та ефективність навчального процесу.

### 3.1.2 Попередня обробка та аугментація даних

Попередня обробка та аугментація є ключовими етапами у підготовці якісного навчального набору даних для ШНМ. Вони забезпечують підготовку даних у форматі, що максимально відповідає потребам моделі, та підвищують стійкість мережі до зовнішніх змін у вхідних даних.

Попередня обробка даних – це процес перетворення необроблених даних у структурований та зрозумілий для програмного забезпечення вид за допомогою різних методів. У системі створення датасету для навчання ШНМ, використовуються такі методи попередньої обробки даних як нормалізація розміру зображень, обробка кількості та глибини кольорових каналів. Завдяки цій структуризації, дані легко використовувати у подальшій обробці, а саме аугментації.   
 Аугментація, або ж нарощування даних – це синтетичне збільшення даних шляхом часткової зміни уже відомих даних. Цей метод дозволяє збільшити кількість навчальних даних для ШНМ у кілька разів, щоб розпізнати той чи інший клас, навіть якщо вхідні дані були пошкоджені чи змінені. Як приклад можна надати – зміна кута нахилу об’єкту, інший колір фону, розмите зображення чи шум на фото.

### 3.1.3 Опис архітектури штучної нейронної мережі

У цьому підрозділі представлено детальний опис архітектури штучної нейронної мережі (ШНМ), яка використовується для розв’язання поставленої задачі класифікації на основі власноруч зібраних даних. Архітектура побудована з урахуванням специфіки завдання, типу вхідних даних та доступних моделей. Особливістю підходу є використання сучасних архітектур, таких як ResNet50 або MobileNetV3, що дозволяє досягти балансу між точністю та ефективністю обчислень.

Концепція архітектури

Обрана модель виконує задачі класифікації з використанням згорткових нейронних мереж (CNN). Такі моделі довели свою ефективність у задачах, пов’язаних із аналізом зображень. Для досягнення високої точності було вирішено застосувати підготовлені попередньо або пусті – залежно від налаштувань гіперпараметрів, моделі з бази даних PyTorch, адаптуючи їх під специфіку задачі шляхом тонкого налаштування (fine-tuning).

Основні компоненти архітектури

1. Вхідний шар   
   Вхідний шар моделі приймає на вхід зображення фіксованого розміру, яке нормалізується на основі обчислених середнього значення та стандартного відхилення даних. Це забезпечує стабільність і прискорює процес навчання. Розмір зображення (128×128 пікселів) обрано для забезпечення балансу між швидкістю обчислень і якістю результату.
2. Приховані шари   
   Модель складається з кількох ключових блоків:

* Згорткові шари - витягують локальні ознаки (текстури, контури) із зображення.
* Шари нормалізації - використовуються для стандартизації ознак, що допомагає уникнути перенасичення градієнтів.
* Шари підвибірки - зменшують розмірність ознак, що сприяє зменшенню обчислювальних витрат і покращенню узагальнення.
* Повнозв’язні шари - виконують узагальнення витягнутих ознак, формуючи вихідний прогноз.

1. Вихідний шар   
   Вихідний шар моделі реалізований як повнозв’язний шар із функцією активації Softmax, який повертає ймовірності приналежності зображення до одного з класів.

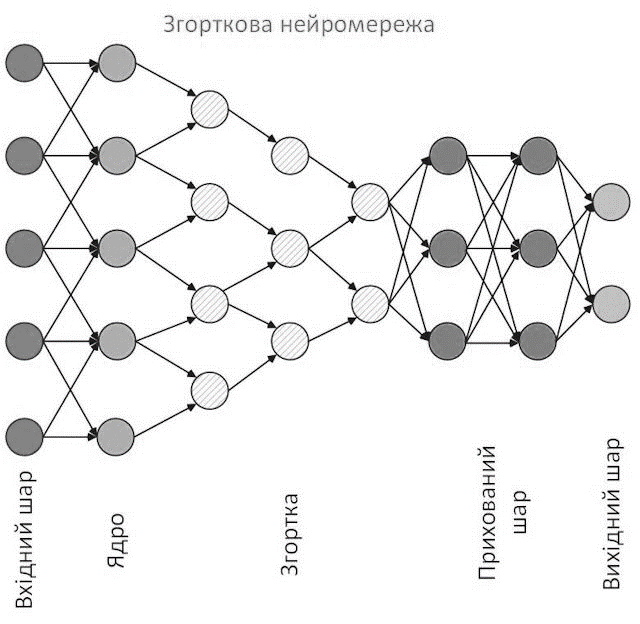


Рис.3.1 – Схема згорткової нейромережі

Запропонована архітектура базується на сучасних підходах до розробки ШНМ, що дозволяє досягти високої точності класифікації на основі власноруч зібраних даних. Вона забезпечує адаптивність до змін у даних, ефективність обчислень і прозорість процесу навчання завдяки ретельному аналізу кожного етапу роботи.

## 3.2 Реалізація програмного забезпечення

У цьому підрозділі описується структура програмного забезпечення, яке реалізує процес створення, навчання та оцінювання моделей штучних нейронних мереж для класифікації зображень. Основна увага приділяється модульності та ефективності програмної архітектури, що забезпечує зручність використання, масштабованість і можливість розширення.

### 3.2.1 Розробка архітектури програмного забезпечення

Програмна реалізація складається з трьох основних компонентів, які відповідають ключовим етапам роботи з моделями штучних нейронних мереж:

1. **Модуль створення датасету**:
   * Цей модуль відповідає за генерацію зображень із заданими параметрами для формування навчальних даних. Генерація виконується з урахуванням різних трансформацій, таких як повороти, масштабування, дзеркальне відображення, регулювання яскравості та додавання шумів.
   * Передбачено інтерактивний інтерфейс, який дозволяє користувачу задавати бажані параметри. Це підвищує зручність налаштування та забезпечує гнучкість створення специфічних наборів даних залежно від потреб користувача.
2. **Модуль навчання моделі**:
   * Цей компонент реалізує процес навчання нейронної мережі на основі сформованого датасету. Програмне забезпечення підтримує використання сучасних архітектур, таких як ResNet50 або MobileNetV3, які можуть бути обрані користувачем залежно від завдання.
   * Модуль забезпечує автоматичний моніторинг процесу навчання. У ньому реалізовано обчислення метрик навчання та валідації, що дозволяє аналізувати якість моделі та вносити корективи за потреби.
   * Результати навчання, такі як графіки втрат і точності, можна візуалізувати, що є важливим інструментом для аналізу ефективності навчання.
3. **Модуль оцінювання моделі**:
   * Після завершення навчання передбачено можливість тестування моделі на нових даних. Модуль оцінювання дозволяє перевіряти точність передбачення та отримувати детальну інформацію про роботу моделі на невідомих даних.
   * Результати оцінювання подаються у зручному для аналізу форматі, що полегшує прийняття рішень щодо подальшого використання моделі.

**Процес та взаємодія між компонентами**

Для забезпечення зручності використання програмного забезпечення всі три компоненти об'єднані в одну логічну систему. Модуль керування надає користувачу можливість обирати між створенням датасету, навчанням або оцінюванням моделі за допомогою простого інтерфейсу. Користувачеві не потрібно заглиблюватися у внутрішню структуру програми – достатньо лише вибрати необхідну дію, і система автоматично виконає всі необхідні операції.

**Особливості реалізації**

* **гнучкість параметризації** - для налаштування програми передбачено використання параметрів, які легко змінюються через конфігураційні файли або командний рядок. Наприклад, можна задати розмір пакета даних, кількість епох, швидкість навчання тощо;
* **збереження результатів -** після навчання модель та метадані (наприклад, статистичні параметри даних і список класів) зберігаються у спеціальному форматі, що дозволяє повторно використовувати модель без необхідності повторного навчання;
* **інтерактивність** - модуль створення датасету має графічний інтерфейс, що значно спрощує роботу для користувачів, які не мають технічної підготовки. Інтерфейс дозволяє швидко задавати параметри генерації зображень і отримувати результат у реальному часі;



Рис. 3.1 – Діаграма послідовностей взаємодії модулів системи між собою

Реалізована структура програмного забезпечення має низку переваг:

* чітке розділення функціональності між модулями забезпечує простоту обслуговування та масштабованість;
* інтерактивний інтерфейс полегшує використання системи навіть для користувачів без спеціальних технічних знань;
* використання сучасних архітектур та можливість налаштування параметрів робить програму універсальним інструментом для вирішення завдань класифікації зображень;

Таким чином, організація програмного забезпечення забезпечує високу продуктивність, зручність використання та адаптивність до потреб різних завдань.

### 3.2.2 Реалізація модуля генерації датасету

Генератор даних для навчання ШНМ, представлений у цьому модулі, є багатофункціональним інструментом для створення зображень із різними аугментаціями, а також для формування відповідних анотацій. Цей інструмент створено з урахуванням вимог до машинного навчання, зокрема для задач комп’ютерного зору, таких як обробка відео чи розпізнавання об’єктів. У цій роботі наведено детальний опис структури, функціоналу, алгоритмів роботи та використаних технологій.

Модуль генератора побудовано з використанням функціоального підходу, що дозволяє легко змінювати та масштабувати функціонал. Основними компонентами є:

1. інтерфейс користувача - для інтерактивного налаштування параметрів використовується бібліотека tkinter. Вікна для вибору відео, параметрів аугментації та класу об’єкта забезпечують зручність роботи користувача.
2. алгоритм відслідковування об’єктів - для визначення області інтересу (ROI – Region of Interest) і подальшого трекінгу об’єктів використовується трекер TrackerCSRT, який належить до сімейства трекерів OpenCV. Його перевагою є точність та стабільність при обробці складних кадрів.
3. аугментація зображень - реалізовано модуль для обробки зображень, який підтримує такі види аугментації, як:

* Обертання;
* Дзеркальне відображення;
* Додавання шуму;
* Розмиття;
* Зміна яскравості та контрастності.

4. збереження даних - оброблені зображення та відповідні анотації зберігаються у форматі JSON із нормалізованими координатами.



Рис. 3.2 – Блок схема алгоритму роботи модуля генерації датасету

Алгоритм роботи модуля генерації датасету побудований таким чином, щоб забезпечити автоматизацію процесу виділення об’єктів з відео, їхнього трекінгу та подальшої генерації аугментованих зображень із відповідними анотаціями. Робота починається з ініціалізації програми, під час якої користувачеві пропонується вибрати відеофайл для обробки. Цей файл є джерелом зображень, які згодом будуть аналізуватися та оброблятися.

Після завантаження відеофайлу відкривається графічний інтерфейс, де користувач може задати параметри генерації датасету. До цих параметрів входять налаштування аугментацій, такі як повороти зображень, їхнє дзеркальне відображення, зміна яскравості, контрасту, розмиття та додавання шуму. Окрім цього, користувач задає розміри зображень, які будуть використовуватися у вихідному датасеті, максимальну кількість кадрів для обробки, а також режим перезапису наявних файлів, якщо така ситуація виникає.

Далі програма зчитує відео та пропонує користувачеві вибрати область інтересу (ROI) на першому кадрі. Вибір здійснюється за допомогою графічного інтерфейсу, де користувач може виділити область прямокутником. Область, обрана користувачем, слугуватиме базою для трекінгу в наступних кадрах. Після підтвердження вибору ініціалізується алгоритм трекінгу, який використовує методику CSRT для точного відстеження об’єкта.

На кожному кадрі програма зчитує поточну область інтересу, коригуючи її, якщо вона виходить за межі зображення. Потім ця область вирізається з кадру та масштабується до заданих параметрів. Отримане зображення зберігається в папці, що була створена для поточного запуску програми. Крім цього, до кожного зображення застосовуються задані користувачем аугментації. Це дозволяє створювати різноманітні варіації вихідного зображення, що підвищує якість та надійність отриманого датасету.

Після збереження кожного кадру та відповідних аугментованих варіацій програма формує JSON-файл із анотаціями. У цьому файлі вказуються координати виділеної області в нормалізованому вигляді (центр та розміри у відносних значеннях до ширини та висоти зображення). Усі анотації збираються в один загальний файл, який містить метаінформацію про кожен кадр та його модифікації.

Робота програми завершується, коли оброблено задану кількість кадрів або коли користувач вручну завершує процес. Програма також передбачає обробку можливих помилок, таких як некоректний вибір області інтересу чи помилки зчитування кадрів із відео. Для підвищення ефективності обробки використовується багатопоточність, що дозволяє одночасно виконувати кілька завдань, таких як збереження кадрів і виконання аугментацій. У підсумку користувач отримує повноцінний датасет із зображеннями, які готові до використання для тренування нейронних мереж чи інших задач комп’ютерного зору.

### 3.2.3 Реалізація модуля навчання моделі

Модуль навчання моделі розроблено з метою автоматизації процесу тренування нейронної мережі для класифікації зображень. Основна увага приділяється обробці вхідних даних, створенню ефективної архітектури моделі, а також налаштуванню гіперпараметрів для досягнення високої точності.



Рис.3.4 – Блок схема алгоритму навчання ШНМ

Робота модуля починається з ініціалізації основних параметрів, таких як розмір батчу - набір даних, які передаються моделі для обробки одночасно під час однієї ітерації навчання, кількість епох, швидкість навчання, розмір зображень та вибір архітектури моделі. Використання GPU автоматично визначається, якщо воно доступне, що значно прискорює навчання.

Далі, на першому етапі, завантажується датасет, де перевіряється наявність та коректність даних. Якщо датасет порожній, програма видає відповідне повідомлення про помилку. Обчислюються середнє значення (mean) та стандартне відхилення (std) для нормалізації вхідних даних.

Після цього виконується поділ вибірки на тренувальну та валідаційну. Використовується стратифікація за класами, що гарантує рівномірний розподіл даних між підвибірками. Для поділу використовується функція train\_test\_split, яка формує індекси для підвибірок.

На наступному етапі створюються завантажувачі даних (DataLoader), які відповідають за подачу зображень до моделі. Батчі формуються з урахуванням обраних індексів для тренувальної та валідаційної вибірки.

Далі ініціалізується архітектура моделі на основі параметра MODEL\_TYPE. Обраний тип моделі, наприклад ResNet50 або MobileNetV3, завантажується з відповідних бібліотек та адаптується до кількості класів у датасеті.

Паралельно налаштовуються функція втрат (крос-ентропія) та оптимізатор (Adam), що використовуються для навчання. Модель переводиться в режим тренування.

В процесі навчання, яке триває визначену кількість епох, виконується:

1. Прямий прохід даних через модель (forward pass), обчислення втрат та зворотній прохід (backpropagation) з оновленням вагів.
2. Обчислення точності для тренувальної та валідаційної вибірки після кожної епохи.
3. Візуалізація результатів навчання шляхом побудови графіків втрат та точності.
4. Дострокова зупинка навчання при досягненні заданого критерію точності.

Після завершення тренування модель разом із метаданими зберігається у файл. Також зберігаються результати навчання у JSON-форматі, включаючи час навчання, втрати, точність та іншу інформацію, яку можна використати для порівняння

### 3.2.4 Реалізація модуля перевірки моделі

Модуль перевірки моделі розроблено для оцінювання продуктивності нейронної мережі на нових, раніше невідомих даних. Основна мета – підтвердити здатність моделі коректно класифікувати зображення із тестового набору. Робота модуля охоплює завантаження навченої моделі, підготовку даних і проведення передбачень із подальшим аналізом результатів.



Рис. 3.5 – Блок схема алгоритму роботи модуля перевірки моделі

Реалізація модуля перевірки моделі дозволяє оцінити її реальну продуктивність у класифікації зображень. Завдяки цьому модулю користувачі можуть швидко перевірити, наскільки добре модель справляється зі своїм завданням, і за потреби вдосконалити її.

* завантаження моделі та метаданих - завантажується файл із попередньо навченою моделлю, який також містить метадані: назви класів, розмір зображень для обробки, середнє значення та стандартне відхилення для нормалізації вхідних даних. Ці метадані забезпечують сумісність моделі з новими даними;
* ініціалізація моделі - на основі метаданих ініціалізується структура моделі. Це дозволяє адаптувати модель до специфіки конкретної задачі класифікації;
* підготовка тестових даних - кожне зображення із тестового набору проходить попередню обробку:
  + - Зміна розміру до стандартного (визначеного в метаданих);
    - Конвертація до формату тензора;
    - Нормалізація піксельних значень на основі середнього значення та стандартного відхилення;
* передбачення класу - оброблене зображення подається на вхід моделі для проведення передбачення. Модель повертає набір імовірностей для кожного класу, після чого вибирається клас із найбільшою імовірністю;
* обчислення точності - результати передбачення порівнюються з реальними мітками класів. На основі кількості правильних передбачень розраховується загальна точність моделі на тестовому наборі;
* виведення результатів - для кожного зображення відображається передбачений клас, реальний клас і коректність передбачення. У кінці підраховується загальна точність моделі, що дозволяє оцінити її якість.

## 3

### 3

## 3.4 Валідація

Тестування та оцінювання моделей штучних нейронних мереж (ШНМ) є важливим етапом розробки програмного забезпечення. Воно охоплює не лише перевірку якості самої моделі, але й тестування коду, який забезпечує її функціонування, інтеграцію та взаємодію із зовнішніми системами. Комплексний підхід до тестування дозволяє досягти високої якості програмного забезпечення, мінімізувати помилки та забезпечити його стійкість у реальних умовах.

### 3.4.1 Аналіз методів тестування та оцінювання

Основне завдання тестування моделі полягає у перевірці її здатності правильно обробляти невідомі дані. Для цього використовується тестовий набір даних, який повинен бути незалежним від навчального і валідаційного наборів. Тестування проводиться за допомогою таких метрик, як точність (accuracy), precision, recall, F1-міра, AUC-ROC у задачах класифікації або середнє абсолютне відхилення (Mean Absolute Error, MAE), середньоквадратична помилка (Mean Squared Error, MSE) та коефіцієнт детермінації (R²) у задачах регресії.

Також важливим є тестування моделі на змінених даних. Для цього застосовуються аугментації, такі як зміна яскравості, поворот зображень чи додавання шуму. Такий підхід дозволяє оцінити стійкість моделі до змін у входах і виявити можливі слабкі сторони.

**Тестування коду**

Якість програмного забезпечення, що забезпечує функціонування моделі, також потребує ретельної перевірки. Для цього застосовуються різні види тестування, які включають:

* **Unit-тестування**

Unit-тести (модульні тести) дозволяють перевірити правильність роботи окремих компонентів програмного забезпечення, таких як функції, методи чи класи. Наприклад, можна тестувати функції для попередньої обробки даних, обчислення метрик, реалізації аугментації чи інших важливих процесів. Модульне тестування дозволяє виявити потенційні дефекти на ранніх етапах розробки, що значно підвищує стабільність та якість програмного забезпечення. Завдяки автоматизації тестування зменшується ризик людських помилок, а можливість повторного виконання тестів забезпечує надійність у разі внесення змін до коду. У перспективі це також сприяє скороченню витрат на підтримку та розвиток системи.

* **Тестування продуктивності**

Цей тип тестування спрямований на оцінювання швидкодії програми, її здатності обробляти великі обсяги даних, працювати під високим навантаженням та витримувати стресові ситуації. Для цього застосовуються інструменти, які дозволяють імітувати різні сценарії роботи системи.

* **End-to-end тестування**

Цей тип тестування охоплює всі аспекти роботи програми – від введення даних до отримання кінцевого результату. Наприклад, end-to-end тестування може включати перевірку повного циклу роботи з даними: завантаження зображення, його аугментація, обробка моделлю та збереження результатів.

* **Методи автоматизації тестування**

У сучасній розробці програмного забезпечення значна увага приділяється автоматизації тестування. Це дозволяє зменшити вплив людського фактора, підвищити швидкість перевірки та забезпечити стабільність результатів. Для цього використовуються інструменти, такі як PyTest або Unittest для Python. Автоматизовані тести можуть бути інтегровані у процес розробки за допомогою інструментів CI/CD (наприклад, GitHub Actions, Jenkins), що забезпечує регулярну перевірку кожного нового внесеного коду.

Різні методи тестування забезпечують перевірку програмного забезпечення з різних сторін, і їх поєднання є найефективнішим підходом. Наприклад, unit-тести забезпечують базову стабільність окремих компонентів, інтеграційні тести перевіряють їх взаємодію, а енд-то-енд тести гарантують функціональність системи в цілому.

### 3.4.2 Тестування програмного забезпечення

У процесі розробки програмного забезпечення тестування відіграє ключову роль у забезпеченні його функціональності, ефективності та стійкості. Використання різних підходів до тестування дозволяє перевірити як окремі модулі системи, так і їхню інтеграцію. У цій роботі були реалізовані методи модульного, інтеграційного, системного та автоматизованого тестування для досягнення високої якості кінцевого продукту.

**Модульне тестування (Unit Testing)**

Модульне тестування є важливою складовою процесу розробки програмного забезпечення, яка спрямована на забезпечення коректності роботи окремих компонентів системи. Цей вид тестування дозволяє перевірити функціональність кожного модуля незалежно від інших, що дає змогу виявляти помилки на ранніх етапах розробки, знижуючи витрати на їх виправлення в майбутньому.

У даному проекті програмне забезпечення складається з трьох основних модулів, кожен з яких відповідає за конкретну функціональність:

1. Модуль обробки відео (Video Processing Module)   
   Цей модуль відповідає за прийом, обробку та аугментацію відео-кадрів. Основними завданнями цього модуля є вибір області інтересу (ROI), застосування серії трансформацій (поворот, зміна яскравості, додавання шуму тощо) та збереження результатів разом з відповідними анотаціями.  
   Для цього модуля створені тести, які перевіряють функціональність основних компонентів, таких як функція apply\_augmentations для застосування трансформацій до зображень та функція save\_augmented\_images для збереження результатів. Зокрема, перевіряється коректність розмірів зображень, правильність генерації анотацій та реакція системи на крайні значення налаштувань.
2. Модуль класифікації об’єктів (Object Classification Module)   
   Цей модуль забезпечує класифікацію зображень на основі заздалегідь навчених моделей нейронних мереж. Основними завданнями цього модуля є завантаження збережених контрольних точок (checkpoint) моделі, її ініціалізація відповідно до метаданих та подальше передбачення класів для зображень.   
   У цьому модулі тести перевіряють коректність завантаження метаданих та конфігурації моделі, правильність роботи трансформацій для підготовки вхідних даних, а також адекватність результатів класифікації для вхідних зображень. Додатково тестується функціональність обчислення точності класифікації для окремих папок з даними.
3. Модуль тренування нейронної мережі (Model Training Module)   
   Цей модуль забезпечує процес навчання та валідації нейронної мережі на основі кастомного набору даних. Основними функціями є підготовка даних, нормалізація зображень, тренувальний процес з оптимізацією параметрів мережі та збереження результатів.  
   Для цього модуля тести охоплюють різні аспекти роботи: перевірка розрахунку статистичних параметрів нормалізації (середнього значення та стандартного відхилення), коректність поділу даних на тренувальний та валідаційний набори, правильність роботи алгоритмів оптимізації та функцій втрат. Крім того, тестується здатність модулю коректно працювати з порожніми наборами даних або аномальними значеннями.

Усі тести реалізовані із застосуванням бібліотеки unittest та принципів модульності. Крім того, для забезпечення ізоляції тестів та уникнення залежності від зовнішнього середовища застосовано техніки мокування, що дозволяють імітувати поведінку об'єктів та функцій.

Для забезпечення якісного тестування програмного забезпечення було використано інструмент Coverage, який дозволяє оцінити рівень покриття вихідного коду тестами. Цей інструмент надає детальну статистику про кількість перевірених рядків коду, а також їх співвідношення до загальної кількості рядків у програмі. Згідно з отриманими результатами, загальне покриття тестами становить 64,59% (425 перевірених рядків із 658). Інструмент також надав інформацію щодо відсутності перевірених розгалужень у коді (branches-covered= 0), що свідчить про можливість подальшої оптимізації тестування.

Отриманий показник покриття дозволяє зробити висновок, що більшість основних функцій програмного забезпечення перевірено, проте є простір для вдосконалення тестування. Для підвищення якості коду та виявлення потенційних недоліків планується додати тести для складних сценаріїв і розгалужень. Таким чином, інструмент Coverage виступає важливим компонентом процесу забезпечення якості розробки програмного забезпечення.

**Тестування продуктивності**

## 

ві

# ЕКОНОМІЧНА ЧАСТИНА

У даному розділі дипломної роботи проводиться економічне обґрунтування фінансової доцільності та ефективності розробки програмного забезпечення. Зокрема розраховується комплексний показник якості проектного рішення, який показує його переваги в порівнянні з аналогами. А також на основі показника якості та ціни споживання проектного рішення та його аналога визначається коефіцієнт конкурентоздатності, який показує спроможність даного проектного рішення конкурувати з аналогами.

Програмне забезпечення призначене для автоматизації дослідження та розробки штучних нейронних мереж, здатне підвищити продуктивність та знизити експлуатаційні витрати в різних галузях економіки. Однак для ефективного впровадження потрібні відповідні ресурси, включаючи апаратне забезпечення, розробку програмного забезпечення, підготовку даних та навчання персоналу.

## 4.1. Розрахунок витрат на розробку програмного забезпечення

Витрати на розробку і впровадження програмних засобів (К) включають:

К = К1,+К2 (4.1)

де K1 - витрати на розробку програмних засобів, грн.

К2 - витрати на відлагодження і дослідну експлуатацію програми рішення задачі на ПК, грн.

Витрати на розробку програмних засобів включають:

* витрати на оплату праці розробників;
* витрати на відрахування у спеціальні державні фонди (Вф);
* витрати на куповані вироби (Кв);
* витрати на придбання спецобладнання для експериментальних  
  робіт (Об);
* накладні витрати (Н);
* інші витрати (Ів).

Витрати на оплату праці розробників проекту визначаються за формулою:

(4.2)

де nij - чисельність розробників і-ої спеціальності j-ro тарифного

розряду, які приймають участь в проектуванні, чол.;

tij - час, який затрачений на розробку проекту співробітника і-ої

спеціальності j-ro тарифного розряду, днів;

Cij - денна заробітна плата і-ої спеціальності j-ro тарифного розряду,

грн.;

(4.3)

де Сij - основна місячна заробітна плата розробника і-ої спеціальності

j-гo тарифного розряду, грн.;

h - коефіцієнт, що визначає розмір додаткової заробітної плати;

р - середня кількість робочих днів у місяці (21 день).

*Таблиця 4.1.*

**Вихідні дані для розрахунку витрат на оплату праці**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Посада виконавців | Місячний оклад, грн. | Середньоденна ставка, грн./дні |
| 1 | Доцент | 12128,97 | 577,57 |
| 2 | Консультант з економіки | 10000 | 476,19 |
| 3 | Студент | 2000 | 95,23 |

*Таблиця 4.2.*

**Розрахунок витрат на оплату праці**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Спеціальність розробника | Час розробки, дні | Денна заробітна плата, грн. | Витрати на розробку, грн. |
| 1 | Доцент | 5 | 577,57 | 2877,85 |
| 2 | Консультант з економіки | 1 | 476,19 | 476,19 |
| 4 | Студент | 85 | 95,23 | 8094,55 |
|  | Разом |  |  | 11 448,59 |

Величину відрахувань у спеціальні державні фонди визначають у процентному співвідношенні від суми основної та додаткової заробітної плати. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 22 %від суми заробітної плати:

Вф = 22,0: 100\*З (4.4)

Вф = 0,22\*11448,59= 2518,6 грн.

*Таблиця 4.3.*

**Розрахунок витрат на куповані вироби**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Найменування купованих виробів | Одиниця виміру | Ціна за одиницю виміру, грн. | Кількість купованих виробів | Сума, грн. |
| 1 | Папір (формат А4) | 500 листів | 163,00 | 1 | 163,00 |
| 3 | Зошит | Шт. | 17,00 | 1 | 17,00 |
| 6 | Флеш пам’ять USB  8GB | Шт. | 125,00 | 1 | 125,00 |
| Всього |  |  | 305,00 |  |  |

При розробці даного програмного забезпечення спеціальне обладнання не використовувалось, тому витрати на спеціальне обладнання відсутні.

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими процентами до витрат на оплату праці:

(4.5)

Н = 0,3 \* 11448,59 = 3434,5 (грн.)

Інші витрати відображають видатки, які не враховані в інших статтях витрат. Вони розраховуються за встановленими процентами до витрат на оплату праці:

Ів =10:100\*З (4.6)

Ів=0,1\* 11448,59 = 1144,8 (грн.)

Витрати на розробку програмного забезпечення розраховуються за формулою:

К1 =3 + Вф + Кв + Об + Н + Ів (4.7)

К1 = 11448,59 + 2518,6 + 305,00 + 3434,5 + 1144,8 = 18851,5 грн.

Витрати на відлагодження і дослідну експлуатацію програмного забезпечення визначаються за формулою:

K2 = SMr\* tBiд (4.8)

де SMr - вартість однієї машино-години роботи конкретного типу ПК, грн./год.;

tBід - машинний час, витрачений на відлагодження і дослідну

експлуатацію програмних засобів, год.

Загальна кількість днів роботи на ПК рівна 60 днів. Середній щоденний час роботи на ПК - 2 год., тому:

tвiд=60\*2=120 (год.)

За даними обчислювального центру НУ "Львівська Політехніка" для ПК типу IBM PC/AT SMг = 20,0 (грн.)

Отже:

К2= 20,0\*120 = 2400 (грн.)

*Таблиця 4.4.*

## 4.2 Кошторис витрат на розробку програмного забезпечення

|  |  |  |
| --- | --- | --- |
| № | Найменування елементів витрат | Сума витрат, грн. |
| 1 | Витрати на оплату праці | 11448,59 |
| 2 | Відрахування у спеціальні державні фонди | 2518,6 |
| 3 | Витрати на куповані вироби | 305,00 |
| 4 | Накладні витрати | 3434,5 |
| 5 | Інші витрати | 1144,9 |
| 6 | Витрати на відлагодження і дослідну експлуатацію програмного забезпечення | 2400,00 |
|  | Всього | 21251,6 |

## 4.3 Визначення експлуатаційних витрат

Для оцінки економічної ефективності розроблюваного програмного продукту слід порівняти його з аналогом, тобто відомим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість машино-годин

роботи ПК (за час дії програми):

(4.12)

де Еп - одноразові експлуатаційні витрати на проектне рішення (аналог), грн.;

Е1п - вартість підготовки даних для експлуатації проектного рішення (аналогу), грн.;

Е2п - вартість машино-годин роботи ПК для виконання проектного рішення (аналогу), грн.

Річні експлуатаційні витрати Веп визначаються за формулою:

(4.13)

де Nп - періодичність експлуатації проектного рішення (аналогу), раз/рік.

Вартість підготовки даних для роботи на ПК визначається за формулою:

(4.14)

де 1 - номери категорій персоналу, який приймає участь у підготовці даних (1=1,2,...L);

n1, - чисельність співробітників 1-ої категорії, чол.;

t1, - трудомісткість роботи співробітників 1-ої категорії по підготовці

даних, год.;

с1 — середнього денна ставка співробітника 1-ої категорії з врахуванням додаткової заробітної плати та відрахувань у спеціальні державні фонди, грн./год.

(4.15)

де - основна місячна заробітна плата працівника 1-ої категорії, грн.;

b - коефіцієнт, який враховує додаткову заробітну плату і відрахування у спеціальні державні фонди;

m - кількість робочих годин у місяці, год.

Для роботи з даними як для проектного рішення так і аналогу потрібен

один працівник, основна місячна заробітна плата якого складає: с° = 10000 грн. Тоді:

c1 = 10000 (1+0,57) / 21\*8 = 93,5 (грн./год)

Трудомісткість працівника по підготовці даних для проектного рішення складає 1 год., для аналога 1,5 год.

*Таблиця 4.7.*

## 4.4 Розрахунок витрат на підготовку даних та реалізацію проектного рішення на ПК

|  |  |  |  |
| --- | --- | --- | --- |
| № | Час роботи співробітників, год. | Середньогодинна заробітна плата, грн./год. | Витрати , грн. |
|  | Проектне рішення | | |
| 1 | 1 | 93,5 | 93,5 |
|  | Аналог | | |
| 1 | 1,5 | 93,5 | 140,3 |

Витрати на експлуатацію ПК визначається за формулою:

(4.16)

де t - витрати машинного часу для реалізації проектного рішення (аналогу), год.;

SMГ - вартість однієї машино-години роботи ПК, грн./год.

Е2п = 1 \* 20,0 = 20 (грн.)

Е2а=1,5\*20,0 = 30,0 (грн.)

Еп=93,5+20,0 =113,5 (грн.)

Еа=140,3 + 30,0 = 170,3 ( грн.)

Веп= 113,5 \* 252 = 28602,0 (грн.)

Веа=170,3 \* 252 = 42915,6 (грн.)

## 4.5. Розрахунок ціни споживання проектного рішення

Ціна споживання - це витрати на придбання і експлуатацію проектного рішення за весь строк його служби:

Цсп=Цп+Вепрv (4.17)

де Цп - ціна придбання проектного рішення, грн.:

(4.18)

де Пр - норматив рентабельності;

Ко - витрати на прив'язку та освоєння проектного рішення на конкретному об'єкті, грн.;

Кк - витрати на доукомплектування технічних засобів на об'єкті, грн.;

Цп=21251,6 \*(1+0,3) = 27627,00 грн.

Benpv - теперішня вартість витрат на експлуатацію проектного рішення (за весь час його експлуатації), грн.:

(4.19)

де Веп - річні експлуатаційні витрати, грн.;

T - строк служби проектного рішення, років;

R - річна ставка дисконтування (приймається у розмірі 12%).

( грн)

(грн.)

Цсп = 27627+103110,21= 130737,61 (грн.)

Цса = 6000,00 + 154710,73 = 160710,73 (грн.)

## 4.6. Визначення показників економічної ефективності

Економічний ефект в сфері проектування рішення:

Епр=Ца-Цп (4.21)

Епр = 6000,00 – 27627= -21627 (грн.)

Річний економічний ефект в сфері експлуатації:

Екс=Веа-Веп (4.22)

Екс= 42915,6 – 28602,0 =14313,6 (грн.)

Додатковий економічний ефект у сфері експлуатації:

(грн.)

Сумарний ефект складає:

-21627 + 90931,44= 69304,44 (грн.)

*Таблиця 4.8.*

**Показники економічної ефективності проектного рішення**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування | Одиниці вимірювання | Значення показників | |
| Базовий варіант | Новий варіант |
| 1 | Капітальні вкладення | грн. | - | 21251,6 |
| 2 | Ціна придбання | грн. | 6000,00 | 27627 |
| 3 | Річні експлуатаційні витрати | грн. | 42915,60 | 28602,00 |
| 4 | Ціна споживання | грн. | 160710,73 | 130737,61 |
| 5 | Економічний ефект в сфері проектування | грн. | - | -21518,4 |
| 6 | Економічний ефект в сфері експлуатації | грн. | - | 14313,6 |
| 7 | Додатковий ефект в сфері експлуатації | грн. | - | 90931,44 |
| 8 | Сумарний ефект | грн. | 69304,44 | |

## Висновки до економічної частини

В даному розділі проведено розрахунок витрат на розробку проектного рішення. Здійснено порівняння з відомим аналогом, і цим показано, що дане проектне рішення має переваги в порівнянні з аналогами, зокрема: надійність, простота використання, гнучкість, зручність. Згідно проведеного економічного обґрунтування дане проектне рішення є конкурентоздатним. Крім того, отримано додатній економічний ефект у розмірі 69304,44 грн. і тому розробка і впровадження цього проектного рішення є економічно доцільними.

# Висновки

# Список літератури

# Додатки