ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка веб-додатку для розпізнавання геометричних елементів зображень з використанням методів глибокого навчання»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Штучний інтелект»

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

_____ Олексій БОГОСЛАВЕЦЬ

Виконав: здобувач вищої освіти гр. ШІД-41

Олексій БОГОСЛАВЕЦЬ

Керівник: Євген ЧИЧКАРЬОВ
Д.т.н., професор

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма «Штучний інтелект»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Штучний інтелект
_____ Ольга ЗІНЧЕНКО
"___" _____ 2024 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Богославцю Олексію Олександровичу

1. Тема кваліфікаційної роботи: «Розробка веб-додатку для розпізнавання геометричних елементів зображень з використанням методів глибокого навчання»

керівник кваліфікаційної роботи Євген ЧИЧКАРЬОВ д.т.н., професор

затверджені наказом Державного університету інформаційно-комунікаційних технологій від « 27» лютого 2024 року № 36.

2. Строк подання кваліфікаційної роботи «31» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи Вибірка зображень, що містять різні геометричні елементи для тренування та тестування моделей глибокого навчання.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз існуючих методів і підходів до розпізнавання геометричних елементів.

2. Опис процесу створення та навчання нейронної мережі для розпізнавання фігур..

3. Процес перевірки, виправлення помилок та підвищення ефективності веб-додатку.

4. Перелік графічного матеріалу
Презентація PowerPoint.

5. Перелік графічного матеріалу
6. Дата видачі завдання «27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ зп	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Визначення актуальності підтримки процесів відновлення	27.02-05.11.23	Виконано
2.	Огляд літератури та сучасних підходів до розпізнавання зображень. Вибір найбільш підходящих методів для реалізації.	05.11-12.11.23	Виконано
3.	Проектування та створення архітектури нейронної мережі. Вибір та налаштування інструментів (TensorFlow, Keras).	13.11-19.11.23	Виконано
4.	Тренування моделі на підготовлених даних. Валідація та оцінка точності моделі.	20.11-25.11.23	Виконано
5.	Проведення функціонального та юзабіліті тестування. Виправлення виявлених помилок.	27.11-03.12.23	Виконано
6.	Аналіз та покращення продуктивності додатку. Оптимізація коду та алгоритмів.	04.12-10.12.23	Виконано
7.	Оформлення результатів дослідження.	11.12-20.12.23	Виконано
8.	Підготовка доповіді до захисту.	21.12-31.05.24	Виконано

Здобувач вищої освіти _____ Олексій БОГОСЛАВЕЦЬ

Керівник кваліфікаційної роботи _____ Євген ЧИЧКАРЬОВ

РЕФЕРАТ

Текстова частина кваліфікаційної роботи: 83 сторінок, 17 рисунки, 21 джерело.

Мета роботи – підвищення ефективності розробки веб-додатку для ефективного розпізнавання геометричних елементів зображень, заснованого на застосуванні методів глибокого навчання.

Об'єкт дослідження – процес розпізнавання геометричних елементів на зображеннях.

Предмет дослідження – методи глибокого навчання, алгоритми комп'ютерного зору та їх реалізація у веб-додатку для розпізнавання геометричних елементів зображень.

Методи дослідження – опрацювання літератури за даною темою, аналіз експлуатаційної документації.

Розроблений веб-додаток застосовує конволюційні нейронні мережі (CNN) для аналізу та обробки зображень. Основною метою є забезпечення високої точності розпізнавання різних геометричних форм, таких як кола, квадрати, трикутники та інші багатокутники. Для досягнення цієї мети було проведено детальне дослідження існуючих архітектур нейронних мереж та обрано оптимальну для задачі розпізнавання геометричних елементів.

Впровадження цього додатку має широкий спектр застосувань у різних галузях, включаючи архітектуру, будівництво, інженерію, медицину та освіту. Завдяки автоматизації процесу аналізу зображень, додаток значно знижує трудовитрати та підвищує точність і швидкість виконання задач.

Галузь використання – галузі математично-геометричних наук.

КЛЮЧОВІ СЛОВА: ВЕБ-ДОДАТОК, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, ГЕОМЕТРИЧНІ ЕЛЕМЕНТИ, ГЛИБОКЕ НАВЧАННЯ, АЛГОРИТМИ

ЗМІСТ

ВСТУП

Актуальність дослідження. У сучасному світі спостерігається стрімкий розвиток цифрових технологій, що сприяє зростанню обсягу візуальної інформації. Автоматичне розпізнавання та обробка цієї інформації є важливою задачею для багатьох галузей, таких як архітектура, будівництво, інженерія, медицина та освіта.

Методи глибокого навчання, зокрема конволюційні нейронні мережі, демонструють високу ефективність у розпізнаванні та класифікації візуальних об'єктів. Використання цих методів у веб-додатку забезпечує високу точність та надійність розпізнавання геометричних елементів на зображеннях.

Автоматизація процесів розпізнавання геометричних елементів сприяє зниженню трудовитрат та підвищенню продуктивності. Це особливо важливо у професійних сферах, де точність і швидкість аналізу зображень мають критичне значення.

Мета роботи – розробка веб-додатку для ефективного розпізнавання геометричних елементів зображень, заснованого на застосуванні методів глибокого навчання.

Об’єкт дослідження – процес розпізнавання геометричних елементів на зображеннях.

Предмет дослідження – методи глибокого навчання, алгоритми комп'ютерного зору та їх реалізація у веб-додатку для розпізнавання геометричних елементів зображень.

Наукові завдання:

визначення актуальності теми проектування;

проаналізувати методи розпізнавання геометричних елементів;

визначення вимог до розробки застосунку;

розробка інтерфейсу та графічного відображення застосунку;

проведення тестування застосунку.

Методи дослідження – опрацювання літератури за даною темою, аналіз експлуатаційної документації.

1 Основи та впровадження комп'ютерного зору

1 Технології комп'ютерного зору та їх розвиток

В останні десятиліття комп'ютерний зір та машинне навчання зазнали значного розвитку, що зробило їх важливими інструментами в багатьох галузях.

Сучасні технології дозволяють не лише автоматизувати рутинні завдання, але й забезпечують більш точний та ефективний аналіз даних. Ця документація присвячена програмі для обробки зображень в режимі реального часу, яка виконує виявлення форм та застосування фільтрів за допомогою бібліотеки OpenCV.

OpenCV-Python - це модуль OpenCV для мови програмування Python, який надає зручний та ефективний інтерфейс для роботи з функціями та алгоритмами комп'ютерного зору. Цей модуль є інтерфейсом Python для оригінальної бібліотеки OpenCV, що реалізована на мові програмування C++.

Основна мета OpenCV-Python - забезпечити зручний та інтуїтивний спосіб використання функцій OpenCV для розробки додатків з обробки зображень та відео на мові програмування Python. Це дозволяє розробникам використовувати всю потужність OpenCV, не виходячи за межі середовища Python.

Основні можливості OpenCV-Python включають в себе:

1. Завантаження та збереження зображень та відео: OpenCV-Python надає функції для завантаження та збереження зображень та відео з різних джерел, таких як файли, веб-камери та інші.
2. Обробка та аналіз зображень: Вона містить розширений набір функцій для обробки та аналізу зображень, таких як зміна розміру, фільтрація, виявлення облич, виявлення країв, бінаризація, вимірювання об'єктів тощо.
3. Робота з відео: OpenCV-Python дозволяє використовувати відеопотік з веб-камери або відеофайлу, а також виконувати різні операції над відео, такі як відстеження об'єктів, визначення руху тощо.
4. Машинне навчання: OpenCV-Python містить модуль ml, який надає функції для реалізації алгоритмів машинного навчання, таких як класифікація, кластеризація, регресія тощо.

5. Візуалізація даних: Вона має можливості для візуалізації даних та результатів обробки зображень та відео за допомогою графіків, діаграм тощо.

6. Інтеграція з іншими бібліотеками Python: OpenCV-Python інтегрується з іншими популярними бібліотеками Python, такими як NumPy, SciPy, Matplotlib тощо, що робить його потужним інструментом для роботи з даними та візуалізації результатів.

7. Багатомовна підтримка: OpenCV-Python підтримується для використання на різних мовах програмування, що дозволяє розробникам з усього світу використовувати його на своїх улюблених мовах.

OpenCV-Python є популярним вибором серед розробників, які працюють у галузі комп'ютерного зору, машинного навчання, обробки зображень та відео. Він надає зручний та потужний інтерфейс для виконання різноманітних завдань, що пов'язані з обробкою зображень, та використання алгоритмів машинного навчання.

OpenCV-Python є інтерфейсом Python до бібліотеки OpenCV, що дозволяє розробникам працювати з функціями комп'ютерного зору зручним та ефективним способом. Цей модуль став дуже популярним у спільноті Python завдяки своїй простоті використання та потужному функціоналу.

Основна функціональність OpenCV-Python охоплює різноманітні аспекти обробки зображень та відео. Він дозволяє завантажувати зображення та відео з різних джерел, таких як файли та веб-камери, і виконувати над ними різні операції, такі як зміна розміру, обрізка, фільтрація, виявлення облич, виявлення країв, бінаризація та багато інших. Він також дозволяє відтворювати та обробляти відеопотоки в реальному часі, що робить його ідеальним інструментом для розробки систем відеоспостереження, систем автоматичного водіння та інших застосувань, які потребують обробки великої кількості відеоданих.

OpenCV-Python також має вбудовані функції для реалізації алгоритмів машинного навчання, що дозволяє використовувати його для класифікації, кластеризації та інших завдань аналізу даних. Крім того, він інтегрується з іншими популярними бібліотеками Python, такими як NumPy та Matplotlib, що робить його

ідеальним інструментом для роботи з великими обсягами даних та візуалізації результатів.

Однією з великих переваг OpenCV-Python є його багатомовна підтримка, що дозволяє розробникам з усього світу використовувати його на своїх улюблених мовах програмування. Це робить його доступним та привабливим для широкого кола розробників та дослідників.

Загалом, OpenCV-Python є потужним та універсальним інструментом для обробки зображень та відео на мові програмування Python. Він надає широкий спектр функцій та можливостей для розв'язання різноманітних завдань у галузі комп'ютерного зору, машинного навчання та аналізу даних.

Давайте поговоримо про деякі додаткові можливості та особливості OpenCV-Python:

- Детектори особливостей: OpenCV-Python містить реалізації різних алгоритмів детекторів особливостей, таких як SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), ORB (Oriented FAST and Rotated BRIEF) та інших. Ці алгоритми корисні для виявлення та опису особливостей на зображеннях, що може бути використано для різних завдань, таких як відстеження об'єктів, реєстрація зображень тощо.
- Морфологічні операції: OpenCV-Python містить функції для виконання морфологічних операцій, таких як розширення, звуження, відкриття та закриття. Ці операції корисні для обробки зображень з малими деталями або шумами.
- Камера глибини: OpenCV-Python може працювати з камерами глибини, які надають додаткову інформацію про глибину об'єктів на зображенні. Це корисно для розробки систем реалістичної відтворення сцен, віртуальної реальності та інших застосувань.
- Робота з векторними та растровими форматами: OpenCV-Python підтримує різні формати зображень, включаючи векторні (наприклад, SVG) та растрові (наприклад, JPEG, PNG) формати. Вона надає зручний інтерфейс для завантаження, збереження та обробки зображень у різних форматах.

- Робота з камерами: OpenCV-Python дозволяє взаємодіяти з веб-камерами та іншими пристроями зображення. Вона може використовуватися для захоплення відеопотоку з камери, відтворення відео на екрані, а також для виконання різних операцій над кадрами в реальному часі.
- Робота з різними типами зображень: OpenCV-Python може працювати з різними типами зображень, включаючи кольорові, відтінки сірого та двійкові зображення. Це дозволяє розробникам використовувати його для різноманітних завдань обробки зображень.
- Ці можливості роблять OpenCV-Python потужним та універсальним інструментом для розробки різноманітних додатків, які вимагають обробки зображень та відео. Він надає широкий спектр функцій та інструментів для реалізації різноманітних завдань в області комп'ютерного зору та машинного навчання.

Нейронні мережі є однією з ключових технологій в галузі машинного навчання. Вони натхненні біологічними нейронами та мозком і складаються з штучних нейронів, організованих у шари.

Кожен нейрон отримує вхідні сигнали, обробляє їх і передає вихідний сигнал до інших нейронів.

TensorFlow - це відкрите програмне забезпечення для машинного навчання і глибокого навчання, розроблене компанією Google (рис. 1.1). Воно надає зручний і гнучкий інтерфейс для побудови та навчання різноманітних моделей штучного інтелекту, включаючи нейронні мережі, зокрема засновані на конволюційних та рекурентних архітектурах, а також інші алгоритми машинного навчання.

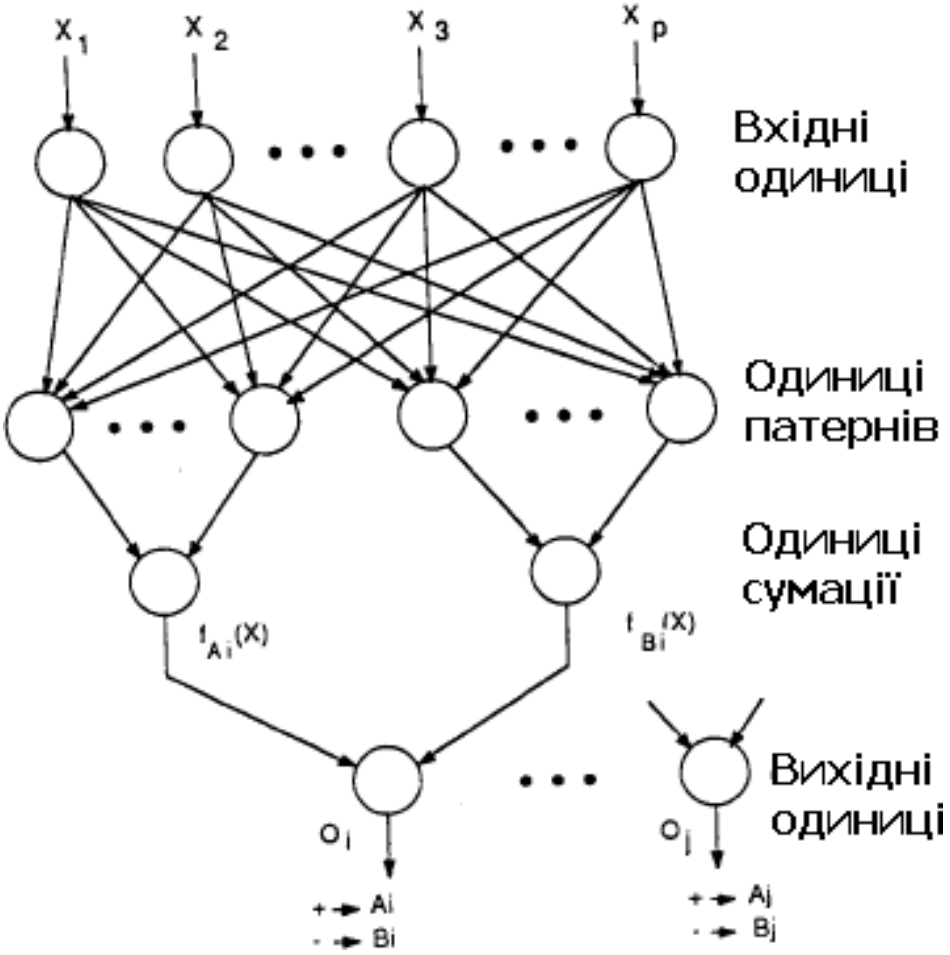


Рис. 1.1 TensorFlow програмне забезпечення для машинного навчання

Основні характеристики TensorFlow:

1. Граф обчислень: TensorFlow використовує граф обчислень для представлення операцій та їх зв'язків у моделі. Це дозволяє виконувати операції ефективно та розподілено на різних пристроях.

2. Модульність: TensorFlow надає широкий набір інструментів та функцій для розробки моделей машинного навчання, які можуть бути легко комбіновані та розширені.

3. Зручний інтерфейс: TensorFlow має зручний інтерфейс користувача, який дозволяє з легкістю створювати, навчати та оцінювати моделі машинного навчання.

4. Розподілений навчання: TensorFlow підтримує розподілене навчання моделей на кластерах серверів, що дозволяє прискорити обчислення та обробку великих обсягів даних.

5. Моделі глибокого навчання: TensorFlow надає багато вбудованих моделей глибокого навчання, таких як нейронні мережі згорткового та рекурентного типу, що дозволяє використовувати їх для різноманітних завдань, таких як класифікація зображень, обробка природних мов, генерація тексту тощо.

6. Підтримка різних мов програмування: TensorFlow має підтримку для різних мов програмування, зокрема Python, C++, Java та інших, що дозволяє використовувати його у різних середовищах та проектах.

7. Зручність використання: TensorFlow надає широкий набір інструментів для візуалізації даних, моніторингу навчання та ефективності моделей, а також інструменти для оптимізації та налаштування гіперпараметрів моделей .

Застосування TensorFlow:

- Розпізнавання облич: TensorFlow може бути використаний для розробки моделей для розпізнавання та класифікації облич на зображеннях;
- Аналіз тексту: TensorFlow може бути використаний для аналізу текстових даних, таких як класифікація тексту, машинний переклад, генерація тексту тощо;
- Обробка зображень: TensorFlow може бути використаний для обробки зображень, таких як фільтрація, сегментація, відновлення тощо;
- Прогнозування часових рядів: TensorFlow може бути використаний для прогнозування та аналізу часових рядів, таких як фінансові дані, метеорологічні дані тощо;
- Підтримка різних галузей: TensorFlow використовується у різних галузях, включаючи науку про матеріали, медицину, фінанси, транспорт, рекламу та інші.

TensorFlow є потужним інструментом для розробки та навчання моделей машинного навчання та глибокого навчання. Він надає широкий набір функцій та інструментів для створення ефективних моделей, які можуть бути використані в різних галузях та застосунках.

Однією з ключових особливостей TensorFlow є його граф обчислень, який дозволяє представляти операції та їх зв'язки у моделі у вигляді графу. Це дозволяє

виконувати операції ефективно та розподілено на різних пристроях, таких як центральний процесор (CPU) або графічний процесор (GPU), а також на кластерах серверів.

Однією з ключових переваг TensorFlow є його модульність. Він надає широкий спектр інструментів та функцій для розробки моделей машинного навчання, які можуть бути легко комбіновані та розширені відповідно до потреб користувача. TensorFlow підтримує різні мови програмування, зокрема Python, C++, Java та інші, що дозволяє використовувати його у різних середовищах та проектах.

Однією з найбільш популярних областей застосування TensorFlow є розпізнавання облич та обробка зображень. Він може бути використаний для розробки моделей, які автоматично розпізнають обличчя на фотографіях або відео, а також для аналізу зображень та витягнення інформації з них.

Крім того, TensorFlow знаходить широке застосування в інших областях, таких як обробка природних мов, аналіз текстів, прогнозування часових рядів, медична діагностика та біоінформатика, фінансовий аналіз, аналіз великих даних та багато інших.

Особливо варто відзначити можливості TensorFlow у розробці та навчанні різноманітних моделей глибокого навчання, таких як нейронні мережі згорткового та рекурентного типу. Він надає багато вбудованих моделей та архітектур для швидкого старту, а також інструменти для налаштування гіперпараметрів та оптимізації моделей.

У світі штучного інтелекту та машинного навчання TensorFlow є одним з найпопулярніших та потужних інструментів, що використовується для розв'язання різноманітних завдань та проблем у багатьох галузях. Він продовжує активно розвиватися та вдосконалюватися, надаючи користувачам нові можливості та інструменти для створення інтелектуальних систем та додатків. Також для полегшення роботи з TensorFlow була розроблена **Keras**.

Keras - це високорівнева бібліотека для машинного навчання, яка забезпечує простий та інтуїтивний інтерфейс для розробки та навчання моделей штучного

інтелекту. Розроблена як високорівневий інтерфейс до TensorFlow, Keras також підтримує інші фреймворки для машинного навчання, такі як Theano та Microsoft Cognitive Toolkit (CNTK), що дозволяє використовувати її в різних середовищах та з різними реалізаціями під задачі.

Основна мета Keras - це надати простий та інтуїтивний спосіб створення та навчання нейронних мереж. Вона розроблена з урахуванням принципів простоти, гнучкості та модульності, щоб зробити процес розробки моделей більш доступним та ефективним для розробників та дослідників.

Основні особливості Keras:

1. Простота використання: Keras має простий та інтуїтивний інтерфейс, що дозволяє швидко розгортати та навчати моделі без необхідності великої кількості коду.

2. Модульність: Keras побудована на принципах модульності, що дозволяє користувачам легко комбінувати різні шари та моделі, а також розширювати функціональність за допомогою власних рішень.

3. Загальна зручність: Keras надає різноманітні інструменти та функції для підтримки різних завдань машинного навчання, таких як класифікація, регресія, кластеризація, а також обробка тексту та зображень.

4. Інтеграція з TensorFlow: В якості високорівневого інтерфейсу до TensorFlow, Keras інтегрується з екосистемою TensorFlow, що дозволяє використовувати всі можливості та функціонал TensorFlow разом із зручним інтерфейсом Keras.

5. Широкий вибір шарів: Keras надає широкий вибір вбудованих шарів для побудови різноманітних типів нейронних мереж, включаючи повнозв'язані шари, згорткові шари, рекурентні шари, а також спеціалізовані шари для обробки тексту та зображень.

6. Можливість використання в навчанні та дослідженнях: Keras часто використовується в академічних дослідженнях та навчанні, оскільки вона надає простий та зручний інтерфейс для експериментування з різними архітектурами та методами машинного навчання.

Застосування Keras:

- Класифікація зображень: Keras часто використовується для розробки моделей для класифікації зображень, таких як розпізнавання облич або класифікація об'єктів на зображеннях;
- Обробка природних мов: Keras може бути використаний для обробки тексту та розробки моделей для машинного перекладу, аналізу сентименту, генерації тексту тощо;
- Рекомендаційні системи: Keras може бути використаний для розробки рекомендаційних систем, які аналізують поведінку користувачів та рекомендують вміст або продукти;
- Прогнозування часових рядів: Keras може бути використаний для прогнозування та аналізу часових рядів, таких як фінансові дані, метеорологічні дані тощо;
- Обробка звуку: Keras також може використовуватися для обробки звукових даних, таких як розпізнавання мови або аналіз аудіо сигналів.

Keras є потужним та популярним інструментом для розробки та навчання моделей машинного навчання. Він надає зручний та інтуїтивний інтерфейс, широкий вибір функцій та інструментів, а також можливість легко комбінувати та розширювати функціональність моделей. Завдяки своїй простоті та гнучкості, він став популярним вибором для розробників та дослідників у багатьох галузях та застосунках машинного навчання.

Розроблена Франсуа Шолеттем (François Chollet) і випущена у 2015 році, Keras став широко використовуваним інструментом в галузі машинного навчання та глибокого навчання.

Головною метою Keras є зробити процес розробки моделей машинного навчання доступним і простим для розробників, дослідників та ентузіастів. Однією з ключових особливостей Keras є його високорівневий інтерфейс, який дозволяє швидко створювати та налаштовувати моделі без необхідності великої кількості коду. Він має зрозумілу та інтуїтивну структуру, що робить процес розробки моделей більш приємним та ефективним.

Керас побудований на принципах модульності та простоти, що дозволяє користувачам легко комбінувати різні шари та моделі, а також розширювати функціональність за допомогою власних рішень. Він надає широкий вибір вбудованих шарів для побудови різноманітних типів нейронних мереж, включаючи повнозв'язані шари, згорткові шари, рекурентні шари, а також спеціалізовані шари для обробки тексту та зображень.

Однією з ключових переваг Keras є його інтеграція з фреймворком TensorFlow. В якості високорівневого інтерфейсу до TensorFlow, Keras інтегрується з екосистемою TensorFlow, що дозволяє використовувати всі можливості та функціонал TensorFlow разом із зручним інтерфейсом Keras. Це дозволяє розробникам використовувати найновітніші функції та можливості TensorFlow без необхідності вивчення його складних API.

Керас знаходить широке застосування в різних галузях та застосунках машинного навчання, включаючи класифікацію зображень, обробку природних мов, рекомендаційні системи, аналіз часових рядів та багато інших. Він часто використовується в наукових дослідженнях, стартапах та великих компаніях для розв'язання різноманітних завдань та проблем у галузі штучного інтелекту.

Загалом, Keras є потужним та популярним інструментом для розробки та навчання моделей машинного навчання. Він надає зручний та інтуїтивний інтерфейс, широкий вибір функцій та інструментів, а також можливість легко комбінувати та розширювати функціональність моделей. Завдяки своїй простоті та гнучкості, він став популярним вибором для розробників та дослідників у багатьох галузях та застосунках машинного навчання.

Системи розпізнавання обличчя використовуються сьогодні в усьому світі урядами та приватними компаніями, їх ефективність різна, і деякі системи раніше були списані через їх неефективність.

Отже, створення програми для розпізнавання людського обличчя є актуальною темою. Метою статті є дослідження теоретичних аспектів розробки системи розпізнавання людського обличчя та практична реалізація відповідного програмного комплексу.

Процедура розпізнавання обличчя просто вимагає, щоб будь-який пристрій, оснащений цифровою фотографічною технологією, генерував і отримував зображення та дані, необхідні для створення та запису біометричного малюнка обличчя людини, якого необхідно ідентифікувати. Розглянуто основні методи розпізнавання обличчя: геометричні методи, метод головних компонент, метод гнучкого порівняння на графах, метод Віоли-Джонса, бінарні шаблони, нейронні мережі.

Запропоновано реалізацію алгоритму роботи системи розпізнавання обличчя. У даній роботі проаналізовано наявні алгоритми та системи виявлення та розпізнавання обличчя, зважені їх переваги та недоліки. Проаналізовано на практиці відсоток точності розпізнавання людського обличчя та продуктивність, враховуючі такі фактори як освітлення, якість зображення, кількість облич на зображенні, реалізовано розпізнавання облич людей з використанням локальних бінарних шаблонів (Local Binary Patterns – LBP) за допомогою бібліотеки комп’ютерного зору OpenCV.

Спрощення машинного навчання. Ця тенденція заснована на розробці програм, щоб користувачі могли автоматизувати вибір моделей, параметрів або вдосконалити процес навчання клацанням миші одним кліком.

Мова программування Python найкраще підходить для таких завдань, оскільки він досить простий у порівнянні з іншими мовами. Більше того, він має відмінні показники обробки даних. Python - популярна мова програмування, створена в 1991 році Гвідо ван Россумом.

Вона має ефективні структури даних високого рівня та простий, але ефективний підхід до об’єктно-орієнтованого програмування. Елегантний синтаксис та динамічне введення тексту Python, поряд з інтерпретованою природою, роблять її ідеальною мовою для сценаріїв та швидкої розробки додатків у багатьох областях на більшості платформ.

Вона може бути використана для веб-розробки на стороні сервера, розробки програмного забезпечення та системних сценаріїв. Python працює на декількох платформах (Windows, Mac, Linux, Raspberry Pi тощо) і має простий англійський

синтаксис, що дозволяє розробникам писати програми з меншою кількістю рядків коду, ніж інші мови програмування.

Python можна вважати процедурною, об'єктно-орієнтованою або функціональною мовою. Однією з основних причин, чому Python використовується для машинного навчання, є те, що вона має багато фреймворків, які спрощують процес кодування та скорочують час розробки. Працюючи з Python, програмісту не потрібно приділяти великої уваги безпосередньому написанню коду: він може зосередити всю свою увагу на вирішенні більш складних проблем, пов’язаних з машинним навчанням.

Простий синтаксис Python допомагає програмісту перевірити складні алгоритми з мінімальним часом реалізації.

Ще однією перевагою Python у машинному навчанні є його гнучкість: наприклад, програміст має вибір між об’єктно-орієнтованим підходом та сценаріями. Python допомагає поєднувати різні типи даних.

Для того, чтобі стати фахівцем в Machine learning, потрібно знати не тільки як систематизувати та аналізувати дані, як працюють базові алгоритми, вивчати реальні кейси, а й знати мову програмування Python, розібратися

буде простіше. Вищезазначені фактори пояснюють, чому Python так широко використовується у машинному навчанні, його простота та гнучкість допомагає працювати над складними алгоритмами з мінімальним часом реалізації.

1.2 Виявлення форм та фільтрації зображень

Програма, яку ми розглядаємо в цій документації, використовує основи комп'ютерного зору для виявлення геометричних форм на зображенні та застосування кольорових фільтрів (рис. 1.2).

Рис. 1.2 Програма фільтрації зображень

CNN - це абревіатура від Convolutional Neural Network (конволюційна нейронна мережа), що є одним з найважливіших інструментів у глибокому навчанні та обробці зображень. Ці нейронні мережі використовуються для автоматичного впізнавання та класифікації зображень, а також для багатьох інших завдань у сферах комп'ютерного зору, обробки природних мов та інших областях.

Основна ідея CNN полягає у використанні шарів, що виконують операцію згортки (convolution) над вхідними даними. Ці шари фільтрують вхідні зображення за допомогою невеликих фільтрів (як правило, квадратних матриць), які рухаються по зображенню та виконують операцію згортки. Після цього застосовуються шари підсемплінгу (pooling), які зменшують розмірність зображення, зберігаючи при цьому важливі ознаки.

Ці операції згортки та підсемплінгу дозволяють CNN автоматично витягувати різноманітні характеристики зображення на різних рівнях абстракції. Починаючи з низькорівневих ознак, таких як границі та форми, і закінчуючи високорівневими ознаками, такими як обличчя чи об'єкти.

Ключовими перевагами CNN є автоматичне вивчення ознак із зображень, навіть при великій кількості даних, здатність до локалізації об'єктів на зображеннях, а також робота з даними у форматі зображень без необхідності попередньої обробки.

Ці нейронні мережі широко застосовуються у таких областях, як комп'ютерний зір, медичне зображення, автоматична розпізнавання облич, автомобільна промисловість (наприклад, для систем допомоги при водінні) та багато інших. Вони є ключовим інструментом у багатьох застосуваннях штучного інтелекту та глибокого навчання, і відіграють важливу роль у вирішенні багатьох складних завдань у сучасному світі.

Давайте розглянемо CNN більш детально.

1. Структура CNN:

- Вхідний шар (Input Layer): Приймає вхідні зображення або дані;
- Шари згортки (Convolutional Layers): Вони складаються з набору фільтрів, які застосовуються до вхідних даних для витягнення особливостей;
- Шари підсемплінгу (Pooling Layers): Використовуються для зменшення розмірності виходу з шарів згортки, зберігаючи при цьому важливі ознаки;
- Повнозв'язані шари (Fully Connected Layers): Останній шар або шари, які виконують класифікацію або регресію на основі витягнутих ознак;
- Функції активації (Activation Functions): Використовуються для нелінійності та активації нейронів у мережі;
- Шари знакування (Normalization Layers): Використовуються для нормалізації вихідних даних з шарів згортки.

2. Операції в CNN:

- Згортка (Convolution): Перемноження фільтра на кожен відрізок вхідних даних для витягнення ознак;
- Підсемплінг (Pooling): Зменшення розмірності виходу з попереднього шару шляхом обчислення статистичних показників (наприклад, максимуму) в певному вікні;

- Пряме поширення (Forward Propagation): Передача вхідних даних через мережу для отримання вихідних прогнозів;
- Зворотне поширення (Backpropagation): Обчислення похідних помилки та оновлення ваг нейронів для зменшення цієї помилки;
- Функція втрат (Loss Function): Функція, яка вимірює, наскільки вдало відповідають прогнози реальним значенням.

3. Архітектури CNN:

- LeNet-5: Одна з перших успішних архітектур CNN, використана для розпізнавання рукописних цифр;
- AlexNet: Відома архітектура, яка виборола перемогу в конкурсі ImageNet у 2012 році, що дозволило показати потужність CNN у класифікації зображень;
- VGG: Серія архітектур, відомих своєю глибиною та простотою структури;
- ResNet: Архітектура з використанням блоків з нейронними з'єднаннями, що дозволяє навчити глибокі мережі без проблеми з втратою градієнту;
- Inception (GoogleNet): Архітектура з використанням "Inception modules", які дозволяють використовувати різні розміри фільтрів в одному шарі.

4. Застосування CNN:

- Розпізнавання облич: Використовується для ідентифікації та відокремлення облич на зображеннях;
- Класифікація зображень: Використовується для класифікації об'єктів на зображеннях за їхніми характеристиками;
- Обробка медичних зображень: Використовується для діагностики захворювань на основі медичних зображень;
- Розпізнавання об'єктів: Використовується для автоматичного розпізнавання об'єктів на зображеннях або відео;
- Загалом, CNN є потужним інструментом для різноманітних завдань обробки зображень та аналізу даних і відіграє важливу роль у багатьох сферах науки та технологій.

Опишемо її компоненти та принципи роботи.

1. Конволюційні шари (Convolutional Layers):

- Це основний компонент CNN. Вони складаються з набору фільтрів (ядро згортки), які рухаються по вхідному зображенню, виконуючи операцію згортки;
- Кожен фільтр виділяє певні особливості на зображенні, такі як границі, форми або текстури;
- Результатом операції згортки є карта ознак, яка показує ступінь відповідності кожної області зображення певній ознаці.

2. Підсемплінг (Pooling):

- Після кожного шару згортки може використовуватись шар підсемплінгу для зменшення розмірності карти ознак;
- Найпоширенішою операцією підсемплінгу є максимальне пулінг, де вибирається максимальне значення в певному вікні зображення;
- Підсемплінг допомагає зменшити обсяг обчислень та параметрів, а також узагальнити виявлені ознаки.

3. Повнозв'язані шари (Fully Connected Layers):

- Після декількох шарів згортки та підсемплінгу, карта ознак перетворюється у вектор, який подається на вхід повнозв'язаному шару;
- Повнозв'язаний шар використовується для класифікації або регресії, вирішуючи задачу, для якої навчалась мережа

4. Функції активації (Activation Functions):

- Після кожного шару нейронів використовується функція активації, яка дозволяє нейронам вивчати нелінійні залежності в даних;
- Популярними функціями активації є ReLU (Rectified Linear Activation), Sigmoid та Tanh.

5. Зворотній розповсюдження помилки (Backpropagation):

- Це алгоритм, за допомогою якого мережа вчиться за допомогою коригування ваг нейронів у відповідь на втрати, отримані під час навчання;

- Під час навчання CNN обчислюється градієнт втрат відносно параметрів мережі, і цей градієнт використовується для оновлення ваг.

6. Архітектурні варіанти CNN:

- У CNN існує безліч архітектурних варіантів, які можуть включати різні кількості та типи шарів;
- Деякі популярні архітектури включають LeNet, AlexNet, VGG, ResNet, Inception і багато інших.

7. Застосування CNN:

- CNN широко використовуються у різних сферах, включаючи комп'ютерний зір, обробку медичних зображень, автомобільну промисловість, рекомендаційні системи та інші області;
- Вони успішно використовуються для класифікації зображень, розпізнавання облич, виявлення об'єктів та багато інших завдань обробки зображень та аналізу даних.

Загалом, CNN є потужним інструментом для аналізу зображень та вирішення різноманітних завдань у сфері комп'ютерного зору та штучного інтелекту. Їхній успіх полягає в їхній здатності ефективно витягати та використовувати ознаки зображень, що робить їх важливим інструментом для багатьох сучасних застосувань.

Ця документація присвячена програмі для обробки зображень в режимі реального часу, яка виконує виявлення форм та застосування фільтрів за допомогою бібліотеки OpenCV.

Основна мета цього проекту — створити інтерактивну програму, яка може ідентифікувати геометричні форми в кадрі, використовуючи алгоритми комп'ютерного зору, а також забезпечити користувача можливістю застосовувати кольорові фільтри до відеопотоку з веб-камери.

Програма має два основні режими роботи:

режим визначення форм та режим фільтрації. У режимі визначення форм програма аналізує контури на зображенні та ідентифікує трикутники, квадрати, прямокутники, пентагони та кола.

Це досягається завдяки використанню функції **detect_shape**, яка базується на алгоритмі апроксимації контурів. У режимі фільтрації користувач може налаштовувати кольоровий діапазон фільтра, використовуючи трекбари, щоб виділяти певні кольори на зображенні. Взаємодія користувача з програмою здійснюється за допомогою графічного інтерфейсу, створеного на основі бібліотеки OpenCV.

Інтерфейс включає декілька кнопок, що дозволяють перемикати режими, зберігати зображення, переглядати історію виявлених форм та завершувати роботу програми.

Крім того, програма логує всі виявлені форми з координатами їх центру, зберігаючи цю інформацію у файл **shapes_log.txt**, що дозволяє відстежувати історію виявлених об’єктів.

Проект має декілька важливих аспектів, які роблять його корисним для вивчення та застосування у реальних сценаріях.

По-перше, він демонструє, як використовувати OpenCV для обробки зображень та відеопотоків у реальному часі.

По-друге, програма ілюструє принципи роботи з контурними об'єктами, їх апроксимацією та класифікацією за геометричними формами.

По-третє, проект показує, як можна створювати інтерактивні інтерфейси для програм комп'ютерного зору, що забезпечують користувача зручними засобами керування функціональністю програми.

Ця документація містить детальний опис всіх аспектів програми, включаючи структуру коду, пояснення алгоритмів та методів, інструкції з встановлення та налаштування, а також приклади використання.

Вона призначена для розробників, які бажають розширити свої знання у галузі комп'ютерного зору та взаємодії з користувачем.

1.3 Налаштування та встановлення програмного забезпечення

Вимоги:

Для запуску програми з виявлення форм та фільтрації зображень, необхідно мати наступне програмне забезпечення та пакети:

1. Python 3.6 або новіша версія
2. OpenCV (cv2)
3. Numpy

Інструкція з встановлення

1. Встановлення Python:

2. Якщо у вас ще не встановлено Python, завантажте та встановіть останню версію з офіційного сайту Python.

3. Створення віртуального середовища (рекомендується): Використання віртуального середовища дозволяє уникнути конфліктів між різними версіями пакетів. Для створення віртуального середовища виконайте наступні команди в терміналі або командному рядку:

python -m venv venv

source venv/bin/activate # для Linux/Mac

venv\Scripts\activate # для Windows

Встановлення OpenCV та Numpy: Встановіть необхідні пакети за допомогою pip: pip install opencv-python-headless

pip install numpy

Скачування та налаштування програми:

2. Інструкції з встановлення

Скачайте файл з кодом програми (наприклад, detector.py) і збережіть його в зручному місці на вашому комп'ютері, або десь у папці де потім буде зручно запустити нашу програму, сам файл можна назвати будь-як….

Запуск програми:

Щоб запустити нашу програму нам необхідно відкрити термінал або командний рядок, також це можна зробити (відкрити термінал) в VSCode або PyCharm, також в інших редакторах коду. Перейдіть до директорії, де збережено файл detector.py

Перейти до директорії можна за допомогою команди: cd назва папки

наприклад ми хочемо перейти до папки desktop:

cd desktop

Після чого тиснемо Enter

Щоб запустити нашу програму необхідно написати наступну команду: На Windows:

python detector.py

На MacOS:

python3 detector.py

1. Використання програми:

- Після запуску програми відкриється вікно з відеопотоком з вашої веб-камери.
- Використовуйте трекбари в верхній частині вікна для налаштування діапазону кольорів для фільтрації.
- Для взаємодії з програмою за допомогою миші використовуйте наступні кнопки:
- Exit: завершення роботи програми.
- Mode: перемикання між режимами виявлення форм та фільтрації.
- Save: збереження поточного кадру.
- History: перегляд історії виявлених форм.

Усунення неполадок

- Проблеми з веб-камерою:
- Якщо програма не може знайти або відкрити веб-камеру, переконайтеся, що вона підключена та працює належним чином. Спробуйте використовувати іншу програму для перевірки функціональності веб-камери.
- Помилки імпорту модулів:
- Якщо при запуску програми з'являються помилки, пов'язані з імпортом модулів, переконайтеся, що всі необхідні пакети встановлені правильно. Виконайте повторну інсталяцію пакетів за допомогою pip install opencv-python-headless numpy.
- Програма не реагує на натискання миші:

- Переконайтеся, що натискаєте в межах активних зон кнопок на екрані. Якщо проблема залишається, перевірте код функції mouse_click на наявність помилок.

3. Огляд коду

Огляд коду

Програма для виявлення форм та фільтрації зображень використовує бібліотеку OpenCV для обробки відеопотоку з веб-камери. Вона дозволяє виявляти геометричні форми, такі як трикутники, квадрати, прямокутники, пентагони та кола, а також застосовувати кольорові фільтри до зображень. Розглянемо основні частини коду.

Ініціалізація та глобальні змінні

Програма починається з імпорту необхідних бібліотек та оголошення глобальних змінних. Встановлюються початкові значення для змінних exit_program, current_mode, save_image та history, які керують роботою програми.

Функція detect_shape

Функція detect_shape приймає контур і повертає тип форми на основі кількості його вершин (рис. 1.3). Вона використовує метод апроксимації контурів для зменшення кількості вершин до мінімально необхідної кількості, що дозволяє класифікувати форму як трикутник, квадрат, прямокутник, пентагон або коло.

```python
def detect_shape(contour):
    shape = "unidentified"
    perimeter = cv2.arcLength(contour, True)
    approx = cv2.approxPolyDP(contour, 0.04 * perimeter, True)

    if len(approx) == 3:
        shape = "triangle"
    elif len(approx) == 4:
        (x, y, w, h) = cv2.boundingRect(approx)
        aspect_ratio = w / float(h)
        if 0.95 <= aspect_ratio <= 1.05:
            shape = "square"
        else:
            shape = "rectangle"
    elif len(approx) == 5:
        shape = "pentagon"
    else:
        shape = "circle"

    return shape
```

Рис. 1.3 Detect_shape функція

Обробка подій миші

Функція mouse_click (рис. 1.4) обробляє події натискання миші, дозволяючи користувачу взаємодіяти з програмою. Вона відповідає за перемикання режимів, збереження зображень, перегляд історії та завершення програми [14].

```
def mouse_click(event, x, y, flags, param):
    global exit_program, current_mode, save_image
    if event == cv2.EVENT_LBUTTONDOWN:
        if 650 <= x <= 780 and 50 <= y <= 100:
            exit_program = True
        elif 650 <= x <= 780 and 150 <= y <= 200:
            current_mode = "filter" if current_mode == "shape" else "shape"
        elif 650 <= x <= 780 and 250 <= y <= 300:
            save_image = True
        elif 650 <= x <= 780 and 350 <= y <= 400:
            show_history()
```

Рис. 1.4 Mouse_click функція

1.4 Створення та отримання значень трекбарів

Функції **create_trackbar** та **get_trackbar_values** відповідають за створення та зчитування значень трекбарів для налаштування кольорових фільтрів (рис. 1.5). Це дозволяє користувачу динамічно змінювати параметри фільтрації.

```python
def create_trackbar(window_name):
    cv2.createTrackbar('Low H', window_name, 0, 180, lambda x: None)
    cv2.createTrackbar('High H', window_name, 180, 180, lambda x: None)
    cv2.createTrackbar('Low S', window_name, 0, 255, lambda x: None)
    cv2.createTrackbar('High S', window_name, 255, 255, lambda x: None)
    cv2.createTrackbar('Low V', window_name, 0, 255, lambda x: None)
    cv2.createTrackbar('High V', window_name, 70, 255, lambda x: None)

def get_trackbar_values(window_name):
    low_h = cv2.getTrackbarPos('Low H', window_name)
    high_h = cv2.getTrackbarPos('High H', window_name)
    low_s = cv2.getTrackbarPos('Low S', window_name)
    high_s = cv2.getTrackbarPos('High S', window_name)
    low_v = cv2.getTrackbarPos('Low V', window_name)
    high_v = cv2.getTrackbarPos('High V', window_name)
    return (low_h, high_h, low_s, high_s, low_v, high_v)
```

Рис. 1.5 create_trackbar, get_trackbar_values функції

Обробка режиму виявлення форм

Функція process_shape_mode обробляє зображення для виявлення форм (рис. 1.6). Вона створює маску для кольорового фільтра, перетворює зображення в сірий колір, виконує гістограмну рівність, розмиває та бінаризує зображення, а потім виявляє контури та визначає їх форми.

```
def process_shape_mode(frame, mask):
    masked = cv2.bitwise_and(frame, frame, mask=mask)
    gray = cv2.cvtColor(masked, cv2.COLOR_BGR2GRAY)
    equalized = cv2.equalizeHist(gray)
    blurred = cv2.GaussianBlur(equalized, (5, 5), 0)
    _, thresh = cv2.threshold(blurred, 30, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_AP

    for contour in contours:
        if cv2.contourArea(contour) < 500:
            continue

        x, y, w, h = cv2.boundingRect(contour)
        if not (50 <= x <= 600 and 50 <= y <= 400):
            continue

        shape = detect_shape(contour)
        M = cv2.moments(contour)
        if M["m00"] != 0:
            cX = int(M["m10"] / M["m00"])
            cY = int(M["m01"] / M["m00"])
        else:
            cX, cY = 0, 0

        cv2.drawContours(frame, [contour], -1, (0, 255, 0), 2)
        cv2.putText(frame, shape, (cX - 20, cY - 20), cv2.FONT_HERSHEY_SIMPLEX, 0
        log_shape(shape, cX, cY)
```

Рис. 1.6 Process_shape_mode функція

Запуск та основний цикл

Основний цикл програми виконує захоплення кадрів з веб-камери, застосовує кольорові фільтри та виявляє форми в режимі реального часу. Він також обробляє взаємодію користувача через інтерфейс та завершує роботу при натисканні відповідної кнопки. Лістинг зображений на Рис. 1.7.

```
cap = cv2.VideoCapture(0)
cv2.namedWindow("Frame")
cv2.setMouseCallback("Frame", mouse_click)
create_trackbar("Frame")
save_image = False

while True:
    ret, frame = cap.read()
    if not ret:
        break

    lower_black = np.array([0, 0, 0])
    upper_black = np.array([180, 255, 50])
    lower_blue = get_trackbar_values("Frame")[:3]
    upper_blue = get_trackbar_values("Frame")[3:]

    mask_black = cv2.inRange(cv2.cvtColor(frame, cv2.COLOR_BGR2HSV), lower_black,
    mask_blue = cv2.inRange(cv2.cvtColor(frame, cv2.COLOR_BGR2HSV), lower_blue, u
    mask = cv2.bitwise_or(mask_black, mask_blue)

    if current_mode == "shape":
        process_shape_mode(frame, mask)
        mode_text = "Mode: Shape Detection"
    else:
        filtered_frame = process_filter_mode(frame, mask)
        frame = filtered_frame
        mode_text = "Mode: Filter"

    cv2.rectangle(frame, (50, 50), (600, 400), (255, 0, 0), 2)
    cv2.rectangle(frame, (650, 50), (780, 100), (0, 0, 255), -1)
    cv2.putText(frame, 'Exit', (660, 85), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 25
    cv2.rectangle(frame, (650, 150), (780, 200), (0, 255, 0), -1)
    cv2.putText(frame, 'Mode', (660, 185), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 2
    cv2.rectangle(frame, (650, 250), (780, 300), (255, 255, 0), -1)
    cv2.putText(frame, 'Save', (660
```

Рис. 1.7 Лістинг захоплення кадрів з веб-камери

Основний цикл

4. Детальний Огляд Функцій

У цьому підрозділі ми розглянемо ключові функції програми для виявлення форм та фільтрації зображень, їх призначення та взаємодію між собою.

Код написаний з великою увагою до деталей для досягнення оптимальної ефективності та функціональності.

Функція detect_shape

Ця функція відповідає за визначення форми контуру на основі кількості його вершин. Вона використовує метод апроксимації контурів для зменшення кількості вершин до мінімально необхідної кількості, що дозволяє класифікувати форму як трикутник, квадрат, прямокутник, пентагон або коло.

def detect_shape(contour): shape = "unidentified"

perimeter = cv2.arcLength(contour, True)

approx = cv2.approxPolyDP(contour, 0.04 * perimeter, True)

1. detect_shape - це функція, що приймає контур та визначає його форму.
2. perimeter = cv2.arcLength(contour, True) - обчислює довжину контуру, використовуючи функцію cv2.arcLength().
3. approx = cv2.approxPolyDP(contour, 0.04 * perimeter, True) - використовується для зменшення кількості вершин у контурі за допомогою методу апроксимації Поліномом Дуги. Другий аргумент - це точність апроксимації, а третій - флаг замкненого контуру.

if len(approx) == 3: shape = "triangle" elif len(approx) == 4: (x, y, w, h) = cv2.boundingRect(approx) aspect_ratio = w / float(h) if 0.95 <= aspect_ratio <= 1.05: shape = "square" else: shape = "rectangle" elif len(approx) == 5: shape = "pentagon" else: shape = "circle"

1. Ця частина коду визначає форму контуру залежно від кількості його вершин.
2. Якщо кількість вершин дорівнює 3, це трикутник, 4 - квадрат або прямокутник, 5 - п'ятикутник, в іншому випадку це коло.
3.

Ці умови базуються на загальних характеристиках кожної геометричної фігури.

Чому так? Це дозволяє класифікувати форму кожного контуру на основі його характеристик. Це корисно для подальшої обробки, наприклад, розпізнавання об'єктів на зображеннях.

Цей підхід до класифікації форм може бути розширений або модифікований відповідно до потреб конкретного застосування. Наприклад, можна додати умови для визначення інших геометричних фігур або вдосконалити умови для визначення квадратів і прямокутників залежно від їх відношення сторін.

def detect_shape(contour):

shape = "unidentified" perimeter = cv2.arcLength(contour, True) approx = cv2.approxPolyDP(contour, 0.04 * perimeter, True) if len(approx) == 3: shape = "triangle" elif len(approx) == 4: (x, y, w, h) = cv2.boundingRect(approx) aspect_ratio = w / float(h) if 0.95 <= aspect_ratio <= 1.05: shape = "square" else: shape = "rectangle" elif len(approx) == 5: shape = "pentagon" else: shape

= "circle" return shape

Обробка подій миші

Функція mouse_click відповідає за обробку подій натискання миші користувачем. Вона реалізує інтерфейс взаємодії з програмою, дозволяючи користувачеві змінювати режими роботи, зберігати зображення та переглядати історію виявлених форм.

- Ця функція відповідає за обробку подій натискання миші користувачем.
- Умова event == cv2.EVENT_LBUTTONDOWN: Перевірка, чи відбулося натискання лівою кнопкою миші.
- Умови 650 <= x <= 780 та 50 <= y <= 100: Перевірка, чи координати миші знаходяться в межах заданої області на інтерфейсі.

Ця функція забезпечує взаємодію користувача з програмою через мишу, дозволяючи виконувати різні дії, такі як вихід з програми, зміна режиму роботи, збереження зображення та перегляд історії виявлених форм.

def mouse_click(event, x, y, flags, param): global exit_program, current_mode, save_image if event == cv2.EVENT_LBUTTONDOWN: if 650 <= x <= 780 and 50

<= y <= 100: exit_program = True elif 650 <= x <= 780 and 150 <= y <= 200: current_mode = "filter" if current_mode == "shape" else "shape" elif 650 <= x <= 780 and 250 <= y <= 300: save_image = True elif 650 <= x <= 780 and 350 <= y

<= 400: show_history()

Створення та отримання значень трекбарів

Функції create_trackbar та get_trackbar_values відповідають за створення та отримання значень трекбарів, що використовуються для налаштування кольорових фільтрів. Це важливий аспект програми, оскільки дозволяє користувачеві динамічно контролювати параметри фільтрації зображень.

- create_trackbar ця функція створює трекбари для налаштування кольорових фільтрів вікна з іменем window_name.
- Для кожного трекбару створюється за допомогою
  cv2.createTrackbar.
- Параметри ('Low H', window_name, 0, 180, lambda x: None)
  вказують на назву трекбару, вікно, початкове значення,
  максимальне значення та зв'язану функцію (в даному випадку, функція відсутня).

Ця функція важлива для налаштування параметрів кольорових фільтрів, які використовуються для виділення об'єктів на зображенні.

Вона створює трекбари, що дозволяють користувачу динамічно змінювати значення параметрів фільтрації, таких як нижні та верхні межі для кольору в просторі HSV (відтінок, насиченість та значення).

Процес створення трекбарів дозволяє користувачеві в реальному часі спостерігати вплив змін параметрів на зображення, що спрощує налаштування фільтрів під конкретні потреби.

Наприклад, трекбари для налаштування нижніх та верхніх меж для компонентів HSV (відтінок, насиченість та значення) дозволяють користувачеві вибрати діапазон кольорів, які вважаються об'єктами інтересу.

Це може бути корисним для виявлення об'єктів певного кольору на зображеннях або для виконання інших завдань обробки зображень, які вимагають виділення певних кольорів.

def create_trackbar(window_name): cv2.createTrackbar('Low H', window_name, 0, 180, lambda x: None) cv2.createTrackbar('High H', window_name, 180, 180, lambda x: None) cv2.createTrackbar('Low S', window_name, 0, 255, lambda x: None)

cv2.createTrackbar('High S', window_name, 255, 255, lambda x: None) cv2.createTrackbar('Low V', window_name, 0, 255, lambda x: None) cv2.createTrackbar('High V', window_name, 70, 255, lambda x: None)

def get_trackbar_values(window_name): low_h = cv2.getTrackbarPos('Low H', window_name) high_h = cv2.getTrackbarPos('High H', window_name) low_s = cv2.getTrackbarPos('Low S', window_name) high_s = cv2.getTrackbarPos('High S', window_name) low_v = cv2.getTrackbarPos('Low V', window_name) high_v= cv2.getTrackbarPos('High V', window_name) return (low_h, high_h, low_s, high_s, low_v, high_v)

Функція process_shape_mode(frame, mask)

def process_shape_mode(frame, mask): masked = cv2.bitwise_and(frame, frame, mask=mask) gray = cv2.cvtColor(masked, cv2.COLOR_BGR2GRAY) equalized = cv2.equalizeHist(gray) blurred = cv2.GaussianBlur(equalized, (5, 5),

0) _, thresh = cv2.threshold(blurred, 30, 255, cv2.THRESH_BINARY) contours, _= cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE) for contour in contours: if cv2.contourArea(contour) < 500: continue x, y, w, h = cv2.boundingRect(contour) if not (50 <= x <= 600 and 50 <= y <= 400): continue shape = detect_shape(contour) M = cv2.moments(contour) if M["m00"] != 0: cX

= int(M["m10"] / M["m00"]) cY = int(M["m01"] / M["m00"]) else: cX, cY = 0, 0 cv2.drawContours(frame, [contour], -1, (0, 255, 0), 2) cv2.putText(frame, shape,

(cX - 20, cY - 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

log_shape(shape, cX, cY)

1. masked = cv2.bitwise_and(frame, frame, mask=mask): Застосування маски до вихідного зображення за допомогою побітового І.

2. gray = cv2.cvtColor(masked, cv2.COLOR_BGR2GRAY): Перетворення зображення у відтінки сірого кольору.

3. equalized = cv2.equalizeHist(gray): Гістограмне вирівнювання градаціїв сірого зображення.

4. blurred = cv2.GaussianBlur(equalized, (5, 5), 0): Застосування гаусівського розмиття для зменшення шуму.

5. _, thresh = cv2.threshold(blurred, 30, 255, cv2.THRESH_BINARY): Порогова бінаризація для виділення контурів.

6. contours, _ = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE): Знаходження контурів на бінаризованому зображенні.

7. Цикл for contour in contours: Ітерація по всім знайденим контурам.

8. Умова if cv2.contourArea(contour) < 500: Перевірка площі контуру для відсіву дрібних об’єктів.

9. Умова if not (50 <= x <= 600 and 50 <= y <= 400): Перевірка положення контуру на зображенні.

10. shape = detect_shape(contour): Визначення форми контуру за допомогою функції detect_shape.

11. Обчислення моментів контуру для знаходження центру маси.

12. cv2.drawContours(frame, [contour], -1, (0, 255, 0), 2): Малювання контуру на вихідному зображенні.

13. cv2.putText(frame, shape, (cX - 20, cY - 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2): Додавання тексту з назвою форми на вихідне зображення.

14. log_shape(shape, cX, cY): Логування форми та її координат.

1.5 Висновки до розділу 1

Досліджено розвиток технологій комп'ютерного зору, що дозволило окреслити їх значення в сучасних інформаційних системах. Встановлено, що розвиток алгоритмів машинного навчання та глибокого навчання сприяє покращенню точності та швидкості обробки зображень. Показано, що комп'ютерний зір стає невід'ємною частиною багатьох галузей, таких як медицина, автомобілебудування, промисловість та безпека.

Розглянуто методи виявлення форм та фільтрації зображень, які є основними компонентами систем комп'ютерного зору. Проаналізовано ефективність різних алгоритмів, таких як детектори країв та контурів. Виявлено, що застосування методів фільтрації дозволяє покращити якість зображень та підвищити точність розпізнавання об'єктів. Простежено, що вибір алгоритму залежить від конкретних вимог до системи та типу зображень.

Описано процес налаштування та встановлення програмного забезпечення для обробки зображень з використанням OpenCV-Python. Визначено, що коректне налаштування середовища розробки та встановлення всіх необхідних бібліотек є критичними для успішної роботи системи. Показано, що OpenCV-Python забезпечує зручний та ефективний інтерфейс для роботи з функціями та алгоритмами комп'ютерного зору, що спрощує розробку додатків.

Розглянуто методи створення та отримання значень трекбарів, які використовуються для налаштування параметрів алгоритмів в реальному часі. Виокремлено, що трекбари дозволяють динамічно змінювати параметри фільтрації та виявлення форм, що підвищує гнучкість та зручність використання системи. Встановлено, що інтерактивність трекбарів сприяє швидшому та точнішому налаштуванню параметрів для досягнення оптимальних результатів.

2 Реалізація програмного забезпечення

2.1 Основний цикл програми

Основний цикл програми є центральною частиною, яка забезпечує безперервну обробку відеопотоку з вебкамери в реальному часі.

Цей цикл виконує кілька важливих завдань, таких як захоплення кадрів, обробка зображень, взаємодія з користувачем через графічний інтерфейс і відображення результатів.

Весь процес можна порівняти з роботою промислових систем обробки зображень, які використовуються в різних галузях, таких як автоматизований контроль якості, медична діагностика, системи безпеки і розпізнавання об’єктів.

Загалом, системи комп'ютерного зору є невід'ємною частиною багатьох сучасних технологій. Наприклад, в автомобільній промисловості, автономні транспортні засоби використовують подібні цикли для обробки відеопотоку з численних камер, щоб виявляти дорожні знаки, пішоходів і інші транспортні засоби.

У медичній сфері, системи обробки зображень допомагають лікарям аналізувати рентгенівські знімки, МРТ і інші медичні зображення для виявлення захворювань. У виробництві, камери використовуються для автоматичного контролю якості продукції, де кожен кадр аналізується для виявлення дефектів або відхилень від стандартів.

Основний цикл програми нашого проекту забезпечує аналогічні функції на базовому рівні, дозволяючи користувачеві виявляти форми і фільтрувати зображення в реальному часі. Розглянемо цей цикл детально, пояснюючи кожен рядок коду та його функціональність.

```
cap = cv2.VideoCapture(0)
cv2.namedWindow("Frame")
cv2.setMouseCallback("Frame", mouse_click)
create_trackbar("Frame")
save_image = False
```

Рис. 2.1 Cap = VideoCapture

1. cap = cv2.VideoCapture(0): Ініціалізація захоплення відео з вебкамери. Аргумент 0 вказує на першу підключену камеру (рис. 2.1).

2. cv2.namedWindow("Frame"): Створення вікна з назвою "Frame" для відображення відео.

3. cv2.setMouseCallback("Frame", mouse_click): Призначення функції mouse_click як обробника подій натискання миші для вікна “Frame".

4. create_trackbar("Frame"): Створення трекбарів для налаштування кольорових фільтрів у вікні “Frame".

5. save_image = False: Ініціалізація змінної для збереження зображення.

На рисунку 2.2 представлений основний цикл

```
while True:
    ret, frame = cap.read()
    if not ret:
        break

    lower_black = np.array([0, 0, 0])
    upper_black = np.array([180, 255, 50])
    lower_blue = get_trackbar_values("Frame")[:3]
    upper_blue = get_trackbar_values("Frame")[3:]

    mask_black = cv2.inRange(cv2.cvtColor(frame, cv2.COLOR_BGR2HSV), lower_black
    mask_blue = cv2.inRange(cv2.cvtColor(frame, cv2.COLOR_BGR2HSV), lower_blue,
    mask = cv2.bitwise_or(mask_black, mask_blue)
```

Рис 2.2 Основний цикл

1. ret, frame = cap.read(): Захоплення поточного кадру з вебкамери. Ret вказує на успішність захоплення, а frame – це сам кадр.

2. if not ret: break: Перевірка успішності захоплення кадру. Якщо кадр не захоплено, цикл переривається.

3. lower_black = np.array([0, 0, 0]): Визначення нижньої межі для чорного кольору в просторі HSV.

4. upper_black = np.array([180, 255, 50]): Визначення верхньої межі для чорного кольору в просторі HSV.

5. lower_blue = get_trackbar_values("Frame")[:3]: Отримання нижніх значень кольору з трекбарів.

6. pper_blue = get_trackbar_values("Frame")[3:]: Отримання верхніх значень кольору з трекбарів.

7. mask_black = cv2.inRange(cv2.cvtColor(frame, cv2.COLOR_BGR2HSV), lower_black, upper_black): Створення маски для чорного кольору.

8. mask_blue = cv2.inRange(cv2.cvtColor(frame, cv2.COLOR_BGR2HSV), lower_blue, upper_blue): Створення маски для синього кольору.

9. mask = cv2.bitwise_or(mask_black, mask_blue): Об'єднання двох масок за допомогою побітового АБО.

Вибір режиму роботи та обробка кадру представлений на рисунку 2.3.

```
if current_mode == "shape":
    process_shape_mode(frame, mask)
    mode_text = "Mode: Shape Detection"
else:
    filtered_frame = process_filter_mode(frame, mask)
    frame = filtered_frame
    mode_text = "Mode: Filter"
```

Рис. 2.3 Вибір режиму роботи та обробка кадру

Обробка кадру:

1. if current_mode == "shape":: Перевірка, чи поточний режим – це режим виявлення форм

2. process_shape_mode(frame, mask): Виклик функції для обробки кадру у режимі виявлення форм.

3. mode_text = "Mode: Shape Detection": Встановлення тексту для відображення режиму роботи.

4. else: ...: Якщо режим не "shape", обробка кадру у режимі фільтрації.

5. filtered_frame = process_filter_mode(frame, mask): Виклик функції для обробки кадру у режимі фільтрації.

6. frame = filtered_frame: Оновлення кадру після фільтрації.

7. mode_text = "Mode: Filter": Встановлення тексту для відображення режиму роботи.

На рисунку 2.4 відображено інтерфейс користувача.

```
cv2.rectangle(frame, (50, 50), (600, 400), (255, 0, 0), 2)
cv2.rectangle(frame, (650, 50), (780, 100), (0, 0, 255), -1)
cv2.putText(frame, 'Exit', (660, 85), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 25
cv2.rectangle(frame, (650, 150), (780, 200), (0, 255, 0), -1)
cv2.putText(frame, 'Mode', (660, 185), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 2
cv2.rectangle(frame, (650, 250), (780, 300), (255, 255, 0), -1)
cv2.putText(frame, 'Save', (660, 285), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 2
cv2.rectangle(frame, (650, 350), (780, 400), (255, 0, 255), -1)
cv2.putText(frame, 'History', (660, 385), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255
cv2.putText(frame, mode_text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,
```

Рис. 2.4 Інтерфейс користувача

1. cv2.rectangle(frame, (50, 50), (600, 400), (255, 0, 0), 2): Малювання синього прямокутника, що визначає робочу область для виявлення форм.

2. Створення інтерфейсних кнопок:

- cv2.rectangle(frame, (650, 50), (780, 100), (0, 0, 255), -1): Кнопка "Exit" для виходу з програми.
- cv2.putText(frame, 'Exit', (660, 85), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2): Напис "Exit" на кнопці.
- cv2.rectangle(frame, (650, 150), (780, 200), (0, 255, 0), -1): Кнопка "Mode" для зміни режиму.
- cv2.putText(frame, 'Mode', (660, 185), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2): Напис "Mode" на кнопці.
- cv2.rectangle(frame, (650, 250), (780, 300), (255, 255, 0), -1): Кнопка "Save" для збереження зображення.
- cv2.putText(frame, 'Save', (660, 285), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2): Напис "Save" на кнопці.
- cv2.rectangle(frame, (650, 350), (780, 400), (255, 0, 255), -1): Кнопка "History" для перегляду історії виявлених форм.

- cv2.putText(frame, 'History', (660, 385),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2): Напис "History" на кнопці.

3. cv2.putText(frame,mode_text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2): Відображення поточного режиму роботи у верхньому лівому куті кадру.

А також код для збереження зображень, які ми зберігали коли тиснули кнопку save

If save_image:
cv2.imwrite(f"saved_image_{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')

2.2 Взаємодія програми з користувачем

Взаємодія з користувачем в програмі є важливою складовою для забезпечення зручного та ефективного використання функціоналу (рис. 2.5). Цей розділ розглядає різноманітні способи взаємодії з програмою, включаючи керування через графічний інтерфейс, обробку введених даних та відображення результатів.

Графічний інтерфейс відображається у вікні програми і надає користувачеві можливість вибору режимів роботи, керування параметрами обробки та взаємодії з результатами.

На інтерфейсі розміщені різноманітні елементи керування, такі як кнопки для зміни режимів, трекбари для налаштування фільтрів та текстові поля для відображення інформації про поточний стан програми. Користувач може взаємодіяти з цими елементами за допомогою миші або клавіатури, щоб виконати різні дії.

Обробка введених даних

Користувач може вводити дані через графічний інтерфейс, наприклад, за допомогою трекбарів для вибору кольорових параметрів або кнопок для запуску

операцій. Ці дані обробляються програмою з урахуванням вказаних параметрів і використовуються для подальшої обробки відеоабо зображень.

Рис 2.5 Взаємодія користувача

Відображення результатів

Після обробки введених даних програма відображає результати на графічному інтерфейсі.

Це може бути відображення обробленого зображення, відзначення об'єктів або відображення інформації про режим роботи.

Результати надаються в реальному часі, що дозволяє користувачеві бачити ефекти внесених змін та приймати відповідні рішення.

Взаємодія з результатами

Користувач може взаємодіяти з відображеними результатами, наприклад, зберегти оброблене зображення, змінити режим роботи або переглянути додаткову інформацію. Це забезпечує користувачеві повний контроль над процесом обробки та дозволяє адаптувати програму під свої потреби.

Загальний підхід до взаємодії

Взаємодія з користувачем базується на принципах зручності та інтуїтивності.

Програма намагається надати користувачеві простий та зрозумілий інтерфейс, щоб спростити використання та забезпечити приємний досвід роботи.

Регулярне оновлення стану програми та взаємодія з користувачем допомагає підтримувати ефективну комунікацію та досягати поставлених цілей.

Основні принципи взаємодії з користувачем включають:

1. Інтуїтивність і простота використання

Програми повинні мати інтуїтивний інтерфейс, який легко зрозуміти навіть для неопытних користувачів. Простість використання дозволяє швидко оволодіти програмою та ефективно використовувати її функціонал.

2. Гнучкість та налаштовуваність

Користувачі повинні мати можливість налаштувати програму під свої потреби та вимоги. Це може включати налаштування параметрів обробки, вибір режимів роботи або зміну вигляду інтерфейсу.

3. Зручність взаємодії

Взаємодія з програмою повинна бути зручною та ефективною. Це означає швидкий доступ до потрібних функцій, зрозумілість управління та інтуїтивність дій.

4. Підтримка користувача

Програми повинні надавати зрозумілу та ефективну підтримку користувачам. Це може бути в формі документації, онлайн-довідки, чату підтримки або форуму спільноти.

5. Постійне вдосконалення

Взаємодія з користувачем – це процес постійного вдосконалення. Розробники повинні враховувати фідбек користувачів, вдосконалювати програму та впроваджувати нові функції для задоволення зростаючих потреб.

Загалом, ефективна взаємодія з користувачем – це ключовий елемент успішного розвитку програмного забезпечення. Розуміння потреб та вимог користувачів дозволяє створювати програми, які відповідають їхнім очікуванням та забезпечують приємний досвід.

2.3 Функціональність Button

У цьому підрозділі розглянуті кнопки, які використовуються у графічному інтерфейсі програми, а також їх функціональність. Кнопки відповідають за виклик певних операцій або зміну параметрів роботи програми.

1. Кнопка "Вихід"

Ця кнопка розташована у верхньому правому куті інтерфейсу та призначена для закриття програми. При натисканні на неї програма завершує свою роботу.

cv2.rectangle(frame, (650, 50), (780, 100), (0, 0, 255), -1)

cv2.putText(frame, 'Exit', (660, 85), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)

2. Кнопка "Режим"

Ця кнопка реалізує зміну режиму роботи програми між режимом виявлення форм та режимом фільтрації.

cv2.rectangle(frame, (650, 150), (780, 200), (0, 255, 0), -1)

cv2.putText(frame, 'Mode', (660, 185), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)

3. Кнопка "Зберегти"

Ця кнопка дозволяє зберегти поточний кадр або зображення під певним ім'ям у файловій системі.

cv2.rectangle(frame, (650, 250), (780, 300), (255, 255, 0), -1)

cv2.putText(frame, 'Save', (660, 285), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)

4. Кнопка "Історія"

Ця кнопка відкриває вікно з історією подій та дій користувача, які були виконані під час роботи програми.

cv2.rectangle(frame, (650, 350), (780, 400), (255, 0, 255), -1)

cv2.putText(frame, 'History', (660, 385), cv2.FONT_HERSHEY_SIMPLEX, 0.7,(255, 255, 255), 2)

Ці кнопки дозволяють користувачеві взаємодіяти з програмою та виконувати різноманітні дії в залежності від потреб і вимог. Реалізація функціональності кнопок забезпечує зручний та ефективний контроль за роботою програми.

3. Збереження зображень

Збереження зображень у програмі відіграє ключову роль для збереження результатів обробки та аналізу.

Вона надає користувачам можливість зафіксувати важливі моменти або результати виконаних операцій для подальшого використання, аналізу чи документування.

Розділ "Збереження зображень" розглядає цей процес у контексті програми обробки відеопотоку, надаючи інформацію про функціональність, реалізацію та взаємодію з користувачем.0

Функціональність збереження зображень

Кнопка "Зберегти", розташована на графічному інтерфейсі програми, надає можливість користувачам зберегти поточне зображення або кадр відеопотоку.

Коли кнопка натиснута, програма створює новий файл у файловій системі та зберігає у ньому зображення у форматі PNG. Це дозволяє користувачам зручно зберігати та документувати результати роботи програми для подальшого використання чи аналізу.

Реалізація збереження зображень

У програмі реалізація збереження зображень виконується за допомогою обробника подій для миші (рис. 2.6).

Коли користувач натискає на кнопку "Зберегти", відбувається збереження поточного кадру або зображення у файлову систему за допомогою функції cv2.imwrite().

Це забезпечує простий та ефективний спосіб збереження результатів роботи програми без необхідності додаткових операцій з файлами.

```
# Обробник події для миші
def mouse_click(event, x, y, flags, param):
    global save_image
    if event == cv2.EVENT_LBUTTONDOWN:
        # Перевірка чи було натиснуто кнопку "Зберегти"
        if 650 <= x <= 780 and 250 <= y <= 300:
            save_image = True

# Головний цикл програми
while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Обробка введених даних та відображення результатів

    # Відображення кнопки "Зберегти"
    cv2.rectangle(frame, (650, 250), (780, 300), (255, 255, 0), -1)
    cv2.putText(frame, 'Save', (660, 285), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,

    # Збереження зображення при натисканні кнопки "Зберегти"
    if save_image:
        cv2.imwrite(f"saved_image_{datetime.datetime.now().strftime('%Y%m%d_%H%M
        save_image = False

    cv2.imshow("Frame", frame)

    if exit_program or cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Рис. 2.6 Збереження зображення

2.4 Висновки до розділу 2

Розглянуто основний цикл програми, який є ключовим для безперервної обробки відеопотоку з вебкамери в реальному часі. Під час розгляду було виявлено, що цей цикл виконує кілька важливих завдань, таких як захоплення кадрів, обробка зображень та взаємодія з користувачем через графічний інтерфейс. Аналізуючи основний цикл програми, ми зрозуміли, що він дозволяє виконувати обробку зображень у реальному часі, що є важливим для багатьох сучасних технологій, таких як автономні транспортні засоби, медична діагностика та автоматизований контроль якості виробництва.

Досліджено взаємодію програми з користувачем через графічний інтерфейс. Виявлено, що ця взаємодія є важливою для забезпечення користувачам зручного та ефективного способу керування програмою. Аналізуючи розділ, ми зрозуміли, що взаємодія з користувачем є необхідною складовою будь-якої програми, оскільки вона дозволяє користувачам взаємодіяти з програмним забезпеченням та виконувати потрібні дії.

Розглянуто функціональність кнопок у веб-додатку для розпізнавання геометричних елементів зображень. Виявлено, що кнопки виконують різні функції, такі як запуск або призупинення обробки зображень, зміна параметрів фільтрів та інші. Аналізуючи цей розділ, ми зрозуміли, що функціональність кнопок є важливою для забезпечення користувачам зручного та ефективного способу взаємодії з програмою

3 РОЗРОБЛЕННЯ РЕКОМЕНДАЦІЙ ЩОДО ТЕСТУВАННЯ ТА ВДОСКОНАЛЕННЯ ДОДАТКУ

3.1 Структура та формат даних

Формат збереження даних відіграє важливу роль у забезпеченні доступності, зручності та ефективності використання інформації, яка генерується та обробляється програмою обробки відео потоку. У цьому розділі ми розглянемо різні аспекти формату збереження даних, його структуру та методи забезпечення цілісності та безпеки.

Важливість формату збереження даних.

Формат збереження даних визначає, як інформація буде представлена та організована у файловій системі. Правильно обраний формат дозволяє забезпечити доступність, зручність та ефективність використання даних, а також забезпечує сумісність з іншими програмами та системами.

Реалізація формату збереження даних.

У програмі обробки відео потоку дані можуть бути збережені у різних форматах, таких як текстові файли, зображення або відеофайли.

Наприклад, результати виявлення форм можуть бути збережені у текстовому форматі, де кожен рядок містить інформацію про тип форми та її координати.

Крім того, зображення кадрів відео потоку можуть бути збережені у форматі PNG або JPEG, щоб забезпечити зручність подальшого перегляду та аналізу.

Забезпечення цілісності та безпеки даних.

Під час збереження даних важливо забезпечити їх цілісність та безпеку. Це можна досягти шляхом використання алгоритмів хешування для перевірки цілісності даних, шифрування для захисту від несанкціонованого доступу та підпису для підтвердження автентичності даних.

Значення формату збереження даних.

Правильно обраний формат збереження даних є ключовим аспектом розробки програмного забезпечення, оскільки він визначає доступність, зручність та ефективність використання інформації.

Він також впливає на сумісність з іншими програмами та системами, що може мати велике значення для обміну даними та інтеграції з іншими системами.

Приклади формату збереження даних

У програмі обробки відеопотоку дані можуть бути збережені у таких форматах:

- **Текстові файли:** Дані про виявлені форми можуть бути збережені у текстовому форматі, де кожен рядок містить інформацію про тип форми та її координати.
- **Зображення:** Кадри відео потоку або результати обробки можуть бути збережені у форматі PNG або JPEG для подальшого аналізу або використання.
- **Відеофайли:** Відеозаписи відеопотоку можуть бути збережені у форматі AVI або MP4 для збереження відео аналізу та архівування результатів.

Приклад коду для збереження даних:

Збереження результатів у текстовий файл *def save_to_text_file(data, filename): with open(filename, "w") as file: for item in data: file.write(f"{item}\n")*

Збереження зображення у форматі PNG *def save_to_png(image, filename): cv2.imwrite(filename, image)*

Збереження відеозапису у форматі MP4 *def save_to_video(frames, filename):*

У цьому прикладі наведено функції для збереження даних у текстовий файл, зображення у форматі PNG та відеозапису. Кожна функція приймає вхідні дані та ім'я файлу, до якого вони будуть збережені.

Розділ: Формат збереження даних

Формат збереження даних є важливою складовою програмного забезпечення, оскільки він визначає, як інформація буде структурована та збережена для подальшого використання.

У цьому розділі ми розглянемо різні аспекти формату збереження даних для програми обробки відеопотоку.

Визначення формату збереження даних

Формат збереження даних визначає структуру та спосіб представлення

інформації у файловій системі.

Це може бути текстовий, бінарний, зображення, відео або будь-який інший формат, який найбільш відповідає потребам програми та її користувачів.

Види форматів збереження даних

У програмі обробки відеопотоку можуть використовуватися різні формати для збереження даних:

Текстові файли: Вони зручні для збереження текстової інформації, такої як логи, конфігураційні файли або результати обробки.

Бінарні файли: Це ефективний спосіб зберігання структурованих даних у бінарному форматі, що забезпечує економію місця та швидкість доступу.

Зображення: Зберігання кадрів відеопотоку у форматі зображень, таких як PNG або JPEG, дозволяє зручно архівувати та обробляти візуальну інформацію.

Відеофайли: Вони підходять для збереження великого обсягу відеоданих та використання їх для подальшого аналізу, відтворення або архівування.

Забезпечення цілісності та безпеки даних.

Під час збереження даних важливо забезпечити їх цілісність та безпеку. Це може бути досягнуто за допомогою резервного копіювання, шифрування, аутентифікації та авторизації, що забезпечує захист інформації від несанкціонованого доступу та змін.

Важливість вибору формату збереження даних.

Правильний вибір формату збереження даних є важливою складовою успішної роботи програми. Він визначає доступність, зручність та ефективність використання інформації, а також впливає на її безпеку та цілісність.

3.2 Інструменти та технології для глибокого навчання

Згорткова нейронна мережа (CNN), також відома як ConvNet, є спеціалізованим типом алгоритму глибокого навчання, в основному призначеним для задач, які потребують розпізнавання об'єктів, включаючи класифікацію, виявлення та сегментацію зображень. CNN застосовуються в різних практичних сценаріях, таких як автономні транспортні засоби, системи камер спостереження та

інші.

1. ШНМ відрізняються від класичних алгоритмів машинного навчання, таких як SVM і дерева рішень, своєю здатністю автономно видобувати ознаки у великих масштабах, оминаючи необхідність ручного інжинірингу ознак і тим самим підвищуючи ефективність.

2. Згорткові шари надають ШНМ незалежні від перекладу характеристики, що дозволяє їм ідентифікувати та вилучати шаблони та ознаки з даних незалежно від змін у положенні, орієнтації, масштабі чи перекладі.

3. Різноманітні попередньо навчені архітектури ШНМ, зокрема VGG-16, ResNet50, Inceptionv3 та EfficientNet, продемонстрували найвищу продуктивність. Ці моделі можна адаптувати до нових завдань з відносно невеликим обсягом даних за допомогою процесу, відомого як точне налаштування.

4. Крім завдань класифікації зображень, ШНМ є універсальними і можуть бути застосовані в ряді інших областей, таких як обробка природної мови, аналіз часових рядів і розпізнавання мови.

Ключові компоненти ШНМ

Згорткова нейронна мережа складається з чотирьох основних частин. CNN навчаються за допомогою наступних частин.

Вони допомагають ШНМ імітувати роботу людського мозку, розпізнаючи патерни та особливості зображень:

- Згорткові шари
- Випрямлений лінійний блок (скорочено ReLU)
- Об'єднання шарів
- Повністю з'єднані шари

Цей розділ занурюється у визначення кожного з цих компонентів на прикладі наступного прикладу класифікації рукописної цифри.

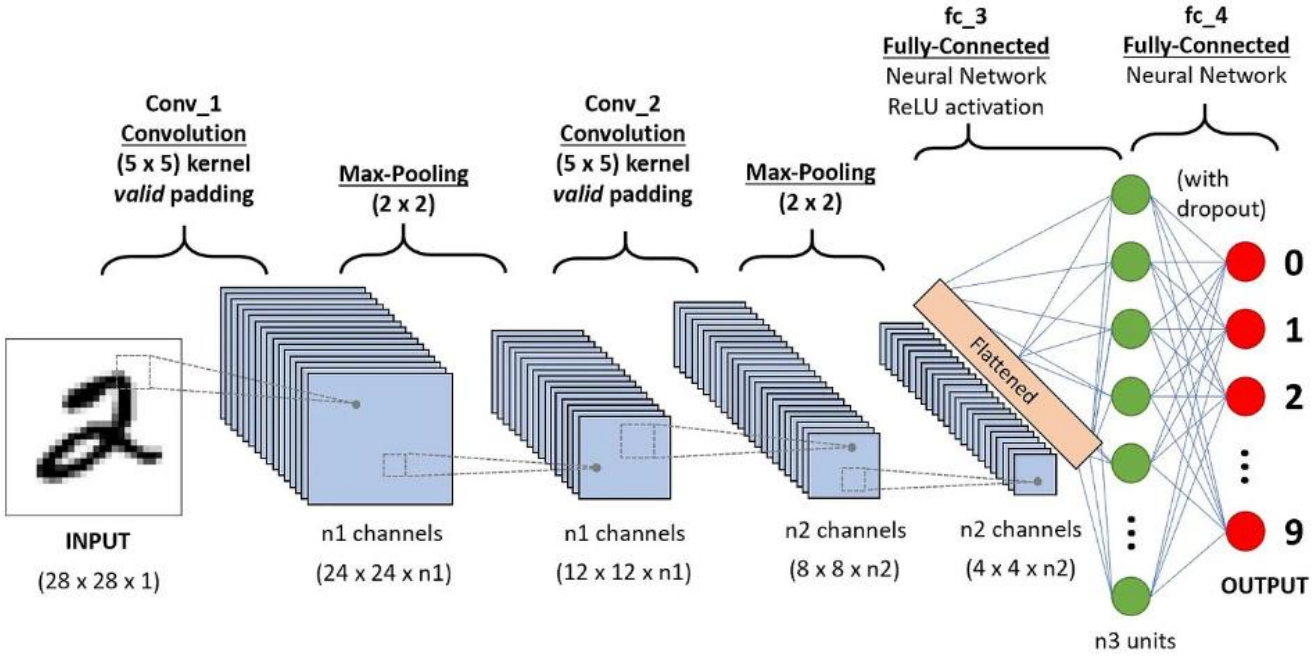


Рис. 3.1 Архітектура ШНМ, застосована для розпізнавання цифр

Це перший будівельний блок CNN. Як випливає з назви, основна математична задача, що виконується, називається згортка, яка полягає в застосуванні функції ковзного вікна до матриці пікселів, що представляє зображення. Функція ковзання, що застосовується до матриці, називається ядром або фільтром, і обидва ці поняття можна використовувати як взаємозамінні.

У шарі згортки застосовується кілька фільтрів однакового розміру, і кожен фільтр використовується для розпізнавання певного шаблону зображення, наприклад, вигину цифр, країв, всієї форми цифр тощо.

TensorFlow

TensorFlow — це бібліотека з відкритим кодом для швидких чисельних обчислень. Він був створений і підтримується Google і був випущений під ліцензією Apache 2.0 з відкритим кодом. API номінально призначений для мови програмування Python, хоча є доступ до основного API C++.

На відміну від інших числових бібліотек, призначених для використання в Deep Learning, як-от Theano, TensorFlow було розроблено для використання як у дослідженнях і розробках, так і в виробничих системах, серед яких не в останню чергу RankBrain у пошуку Google і цікавий проект DeepDream.

Він може працювати на системах з одним процесором і графічним процесором, а також на мобільних пристроях і великомасштабних розподілених системах із сотень машин. Обчислення описується в термінах потоку даних і операцій у структурі орієнтованого графа:

– вузли виконують обчислення та мають нуль або більше входів і виходів. Дані, які переміщуються між вузлами, відомі як тензори, які є багатовимірними масивами реальних значень.

– ребро визначає потік даних, розгалуження, цикл і оновлення стану. Спеціальні ребра можна використовувати для синхронізації поведінки в межах графа, наприклад, очікування завершення обчислень для кількох вхідних даних.

– операція — це назване абстрактне обчислення, яке може приймати вхідні атрибути та створювати вихідні атрибути. Наприклад, ви можете визначити операцію додавання або множення.

У контексті побудованої системи дана бібліотека застосовувалася для отримання базових шарів нейронних мереж, таких як Input, Dropout, Dense, LSTM, TimeDistributed, RepeatVector. А для регуляризації використовувалися відповідні regularizers.

Keras

Keras — це високорівневий API глибокого навчання, розроблений Google для впровадження нейронних мереж. Він написаний на Python і використовується для полегшення впровадження нейронних мереж. Він також підтримує численні серверні обчислення нейронної мережі.

Keras відносно легко освоїти та працювати з ним, тому що він забезпечує інтерфейс Python з високим рівнем абстракції, маючи можливість використовувати кілька серверних інтерфейсів для обчислень. Це робить Keras повільнішим, ніж інші фреймворки глибокого навчання, але надзвичайно зручним для початківців.

TensorFlow прийняв Keras як офіційний API високого рівня. Keras вбудовано в TensorFlow і може використовуватися для швидкого глибокого навчання, оскільки він надає вбудовані модулі для всіх обчислень нейронної мережі. У той же час, обчислення з використанням тензорів, обчислювальних графів, сеансів

тощо можна створювати на замовлення за допомогою Tensorflow Core API, який надає повну гнучкість і контроль над програмою та дозволяє реалізувати ідеї за відносно короткий час.

Keras — це API, створений таким чином, щоб його було легко освоїти. Keras був створений, щоб бути простим. Він пропонує послідовні та прості API, скорочує кількість дій, необхідних для реалізації загального коду, і чітко пояснює помилки користувача.

Час створення прототипу в Keras менше. Це означає, що ідеї можна реалізувати та розгорнути за більш короткий час. Keras також надає різноманітні варіанти розгортання залежно від потреб користувача.

Мови з високим рівнем абстракції та вбудованими функціями працюють повільно, тому створювати спеціальні функції в них може бути важко. Але Keras працює на основі TensorFlow і відносно швидкий. Keras також глибоко

інтегровано з TensorFlow, тому ви можете легко створювати персоналізовані робочі процеси.

Дослідницьке співтовариство Keras є великим і високорозвиненим. Доступна документація та допомога є набагато більшими, ніж інші структури глибокого навчання.

Keras комерційно використовується багатьма компаніями, як-от Netflix, Uber, Square, Yelp тощо, які розгорнули продукти у відкритому доступі, створені за допомогою Keras.

Opencv-python

OpenCV - це open source бібліотека комп'ютерного зору, яка призначена для аналізу, класифікації та обробки зображень. Широко використовується в таких мовах як C, C++, Python і Java.

Кожне зображення складається з набору пікселів. Піксель - це будівельний блок зображення. Якщо уявити зображення у вигляді сітки, то кожен квадрат у сітці містить один піксель, де точці з координатою (0, 0) відповідає верхній лівий кут зображення. Наприклад, уявімо, що у нас є зображення з роздільною здатністю 400x300 пікселів. Це означає, що наша сітка складається з 400 рядків і 300 стовпців.

У сукупності в нашому зображенні є 400*300 = 120000 пікселів.

У більшості зображень пікселі представлені двома способами: у відтінках сірого і в колірному просторі RGB. У зображеннях у відтінках сірого кожен піксель має значення між 0 і 255, де 0 відповідає чорному, а 255 відповідає білому. А значення між 0 і 255 приймають різні відтінки сірого, де значення ближче до 0 темніші, а значення ближче до 255 світліші:

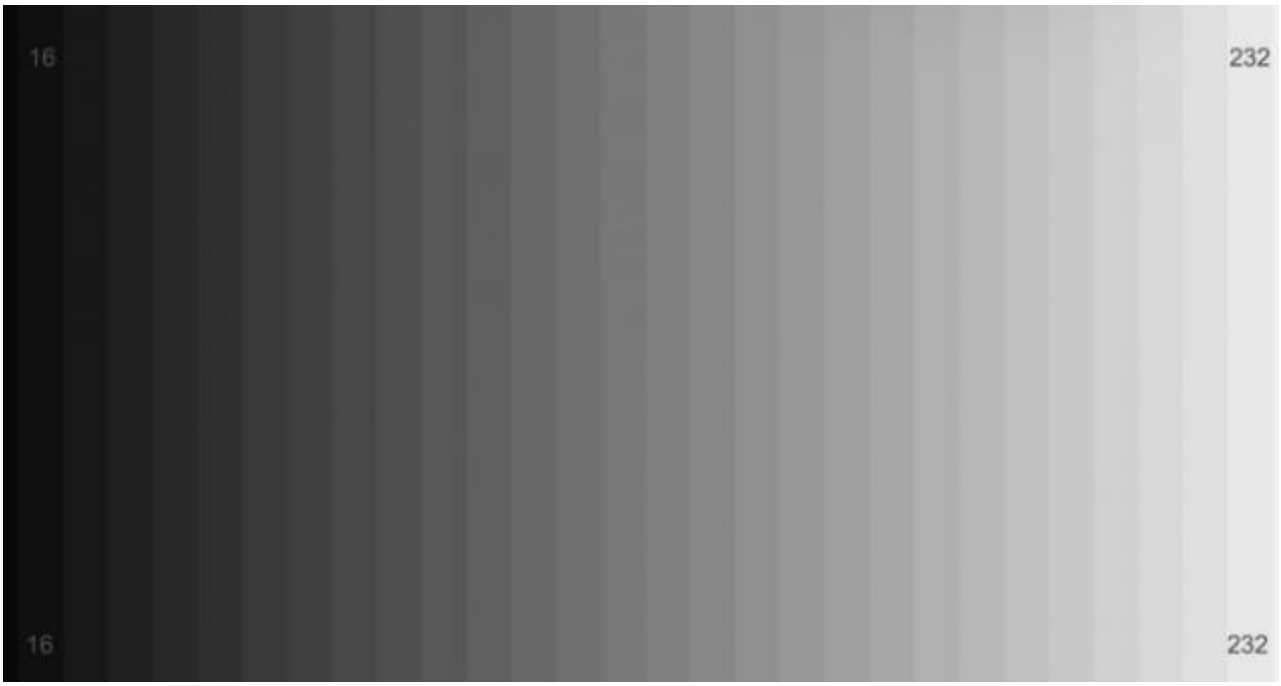


Рис. 3.2 Кольоровий простір RGB

Кожна з трьох компонент представлена цілим числом у діапазоні від 0 до 255 включно, що вказує як «багато» кольору міститься. Виходячи з того, що кожна компонента представлена в діапазоні [0,255], то для того, щоб уявити насиченість кожного кольору, нам буде достатньо 8-бітного цілого беззнакового числа. Потім ми об'єднуємо значення всіх трьох компонент у кортеж виду (червоний, зелений, синій). Наприклад, щоб отримати білий колір, кожна з компонент має дорівнювати 255: (255, 255, 255, 255). Тоді, щоб отримати чорний колір, кожна з компонент має дорівнювати 0: (0, 0, 0, 0).

Імпорт бібліотеки OpenCV.

Тепер перейдемо до практичної частини. Перше, що нам необхідно зробити -

це імпортувати бібліотеку. Є кілька шляхів імпорту, найпоширеніший - це використовувати вираз:

import cv2

Також можна зустріти таку конструкцію для імпорту цієї бібліотеки:

from cv2 import cv2

Загрузка, отображение и сохранение изображения.

```
def loading_displaying_saving():
    img = cv2.imread('girl.jpg', cv2.IMREAD_GRAYSCALE)
    cv2.imshow('girl', img)
    cv2.waitKey(0)
    cv2.imwrite('graygirl.jpg', img)
```

Для завантаження зображення ми використовуємо функцію cv2.imread(), де першим аргументом вказується шлях до зображення, а другим аргументом, що є необов'язковим, ми вказуємо, в якому колірному просторі ми хочемо зчитати наше зображення. Щоб зчитати зображення в RGB - cv2.IMREAD_COLOR, у відтінках сірого - cv2.IMREAD_GRAYSCALE. За замовчуванням цей аргумент приймає значення cv2.IMREAD_COLOR.

Ця функція повертає 2D (для зображення у відтінках сірого) або 3D (для кольорового зображення) масив NumPy. Форма масиву для кольорового зображення: висота x ширина x 3, де 3 - це байти, по одному байту на кожну з компонент. У зображеннях у відтінках сірого все трохи простіше: висота x ширина.

За допомогою функції cv2.imshow() ми відображаємо зображення на нашому екрані. Як перший аргумент ми передаємо функції назву нашого вікна, а другим аргументом - зображення, яке ми завантажили з диска, однак, якщо ми далі не вкажемо функцію cv2.waitKey(), то зображення моментально закриється. Ця функція зупиняє виконання програми до натискання клавіші, яку потрібно передати першим аргументом. Для того, щоб будь-яка клавіша була зарахована, передається 0.

Доступ до пікселів і маніпулювання ними

Для того, щоб дізнатися висоту, ширину і кількість каналів у зображення

можна використовувати атрибут shape:

print("Висота:"+str(img.shape[0]))

print("Ширина:" + str(img.shape[1]))

print("Кількість каналів:" + str(img.shape[2]))

Важливо пам'ятати, що у зображень у відтінках сірого img.shape [2] буде недоступним, оскільки ці зображення представлені у вигляді 2D масиву.

Щоб отримати доступ до значення пікселя, нам просто потрібно вказати координати x і y пікселя, який нас цікавить. Також важливо пам'ятати, що бібліотека OpenCV зберігає канали формату RGB у зворотному порядку, в той час як ми думаємо в термінах червоного, зеленого і синього, то OpenCV зберігає їх у порядку синього, зеленого і червоного кольорів:

(b, g, r) = img[0, 0]

print("Червоний: {}, Зелений {}, Синій: {}".format(r, g, b))

Спочатку ми беремо піксель, який розташований у точці (0,0). Цей піксель, та й будь-який інший піксель, представлені у вигляді кортежу. Зауважте, що назви змінних розташовані в порядку b, g і r. У наступному рядку виводимо значення кожного каналу на екран. Як можна побачити, доступ до значень пікселів досить простий, так само просто можна і маніпулювати значеннями пікселів:

img[0, 0] = (255, 0, 0)

(b, g, r) = img[0, 0]

print("Червоний: {}, Зелений: {}, Синій: {}".format(r, g, b))

В першому рядку ми встановлюємо значення пікселя (0, 0) рівним (255, 0, 0), потім ми знову беремо значення даного пікселя і виводимо його на екран, в результаті мені на консоль вивелося наступне:

Червоний: 251, Зелений: 43, Синій: 65

Червоний: 0, Зелений: 0, Синій: 255

Розглянемо механічні алгоритми відпрацювання додатку. Першим пунктом розробимо діаграму прецедентів, показано на рис. 3.3.

Рис. 3.3 Діаграма прецедентів

Наступна реалізація містить в собі діаграму діяльності відтворену в додатку, вона показана на рисунку 3.4.

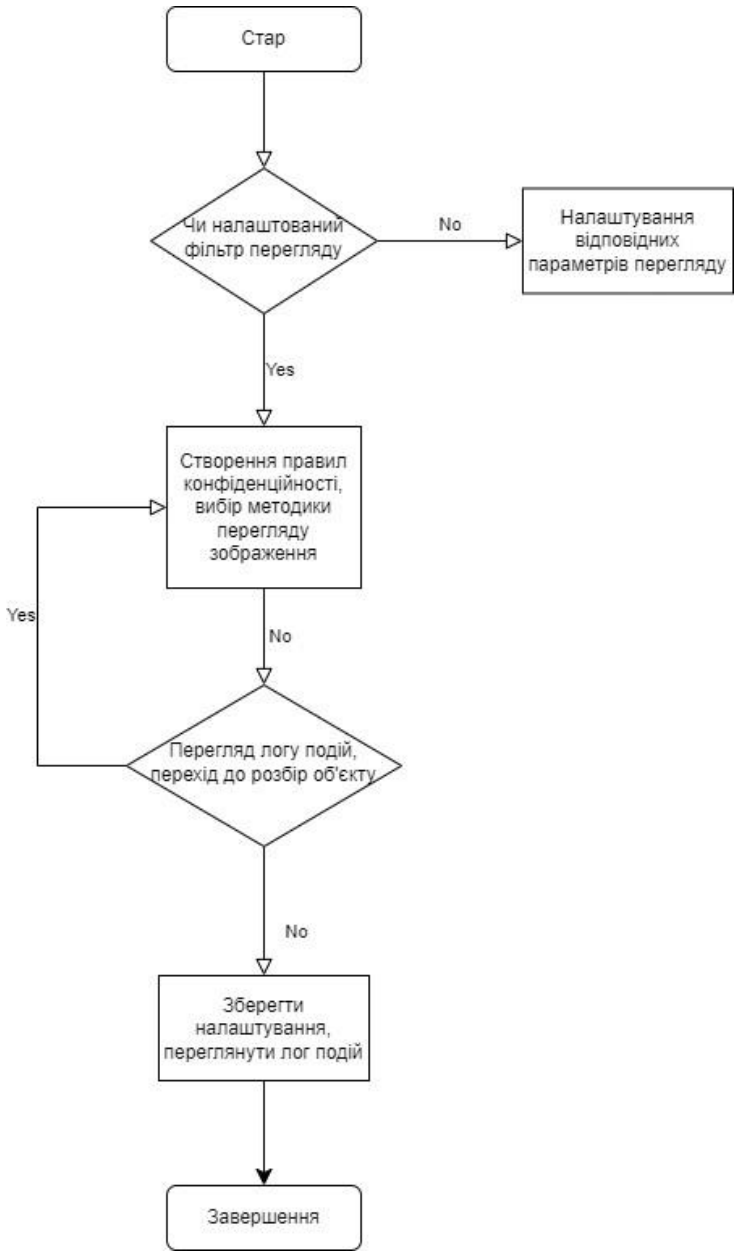


Рис. 3.4 Діаграма діяльності

3.3 Перевірка функціональності

Тестування програми є важливим етапом розробки, який дозволяє перевірити функціональність, надійність та продуктивність програмного забезпечення. У цьому розділі ми розглянемо методи та стратегії тестування для програми обробки відеопотоку, а також важливі аспекти тестування.

Методи тестування.

- Модульне тестування. Перевірка окремих модулів програми на коректність

їх роботи. Для цього можуть використовуватися різноманітні тести, включаючи тести одиничних випадків, тести меж та тести на винятки.

- Інтеграційне тестування. Перевірка взаємодії між різними модулями програми та їх коректність у складі єдиної системи.
- Системне тестування. Тестування програми як єдиного цілого з використанням різних тестових сценаріїв та вхідних даних.
- Валідація. Перевірка відповідності програми вимогам та специфікаціям.

Стратегії тестування.

- Функціональне тестування. Перевірка коректності виконання функцій програми, таких як виявлення форм, фільтрація зображення та інші.
- Візуальне тестування. Оцінка якості обробки відеопотоку та точності виявлення об'єктів шляхом візуального аналізу результатів.
- Навантажувальне тестування. Оцінка продуктивності програми при великому навантаженні або обробці великого обсягу даних.

Важливі аспекти тестування.

- Автоматизація тестування. Використання автоматизованих тестів для ефективного та повторюваного тестування програми.
- Тестування на різних платформах. Перевірка роботи програми на різних операційних системах та пристроях для забезпечення сумісності та відповідності стандартам.
- Тестування на входження. Перевірка коректності обробки некоректних або неправильних вхідних даних.

Переваги тестування програми.

- Забезпечення високої якості програмного забезпечення за рахунок виявлення та виправлення помилок на ранніх етапах розробки.
- Підвищення надійності та стабільності програми шляхом ідентифікації та виправлення дефектів.
- Забезпечення відповідності програми вимогам та очікуванням користувачів.

Приклади тестування.

- Тест виявлення форм. Створення тестових вхідних даних з різними формами

та перевірка коректності виявлення та класифікації цих форм програмою.

- Тест продуктивності. Виконання програми з великим обсягом відеоданих та оцінка

3.4 Проведення тестування

Процес тестування програмного забезпечення є важливою частиною розробки, що має на меті перевірити, чи відповідає програма вимогам, функціонує коректно та надійно. У даному розділі ми розглянемо етапи та методи тестування програми обробки відеопотоку.

Етапи тестування.

– Планування. Визначення обсягу тестування, складання плану тестування та розробка тестової документації.

– Підготовка тестових даних. Створення тестових наборів даних з відомими властивостями для використання під час тестування.

– Виконання тестів. Запуск тестів на програму та аналіз результатів їх виконання.

– Аналіз результатів. Оцінка результатів тестування, ідентифікація та виправлення дефектів, а також оцінка відповідності програми вимогам.

– Документування. Фіксація результатів тестування у вигляді звітів, логів та іншої документації.

Методи тестування.

– Ручне тестування. Виконання тестів вручну з використанням різних сценаріїв взаємодії з програмою.

– Автоматизоване тестування. Створення автоматизованих тестів, які виконуються автоматично за допомогою спеціальних інструментів.

– Тестування відповідності стандарта. Перевірка відповідності програми вимогам та стандартам безпеки, які встановлені для даного типу програмного забезпечення.

Види тестування.

– Функціональне тестування. Перевірка функціональності програми та її відповідності вимогам.

– Тестування навантаження. Визначення максимального обсягу роботи, який програма може витримати без втрати продуктивності.

– Тестування безпеки. Перевірка програми на наявність потенційних вразливостей та заходи для їх запобігання.

Забезпечення якості тестування.

– Повнота тестування. Перевірка всіх можливих аспектів програми та її функціональності.

– Тестування на реальних даних. Використання реальних або реалістичних наборів даних під час тестування для відтворення реальних умов використання програми.

Інструменти для тестування.

– Фреймворки для автоматизованого тестування. Наприклад, Selenium для веб-програм, PyTest для тестування Python-програм.

– Інструменти для аналізу коду. Наприклад, Pylint для перевірки Python-коду на виявлення потенційних помилок та недоліків.

– Інструменти для тестування безпеки. Наприклад, Burp Suite для виявлення потенційних вразливостей в веб-додатках.

Загальні кроки тестування програми обробки відеопотоку.

– Сценарії тестування. Визначення різних сценаріїв взаємодії з програмою, таких як виявлення форм, фільтрація зображення та інші.

– Створення тестових даних. Підготовка тестових вхідних даних для використання під час тестування.

– Виконання тестів. Виконання різних тестових сценаріїв на програмі та аналіз їх результатів.

– Виявлення та виправлення дефектів. Ідентифікація та усунення помилок та недоліків, виявлених під час тестування.

3.5 Виявлення та виправлення помилок

Пошук та виправлення помилок (debugging) є невід'ємною частиною розробки програмного забезпечення. У цьому розділі ми розглянемо процес виявлення, аналізу та виправлення помилок в програмі обробки відеопотоку.

Етапи процесу пошуку та виправлення помилок.

– Виявлення помилок. Ідентифікація аномальної поведінки програми, яка може бути пов'язана з помилкою.

– Відтворення помилок. Створення умов, які спричинюють виникнення помилок, для відтворення їх та аналізу.

– Аналіз помилок. Дослідження причин виникнення помилок та визначення їхнього впливу на роботу програми.

– Виправлення помилок. Внесення необхідних змін у програму для усунення помилок та відновлення її коректної роботи.

– Тестування виправлень. Перевірка внесених змін для підтвердження їхньої ефективності та відсутності нових помилок.

Інструменти для пошуку та виправлення помилок.

– Дебагери. Інструменти, які дозволяють крокувати по коду програми, дозволяючи аналізувати її стан у певні моменти виконання.

– Логування. Додавання до програми спеціальних виразів для запису інформації про стан виконання програми у вигляді лог-файлів.

– Моніторинг системи. Використання програмних або апаратних засобів для відстеження роботи програми та виявлення аномалій.

Кращі практики пошуку та виправлення помилок.

– Розуміння коду. Глибоке розуміння архітектури та логіки програми допомагає швидше виявляти та виправляти помилки.

– Систематичний підхід. Використання структурованих методів для виявлення та виправлення помилок дозволяє ефективно керувати цим процесом.

Важливі аспекти виправлення помилок.

– Приоритезація. Визначення важливості та впливу помилок на роботу

програми для визначення порядку їх виправлення.

– Контроль якості. Перевірка виправлених помилок через тестування для підтвердження їхнього кількісного розподілу.

3.6 Виявлення та виправлення помилок

Оптимізація коду є важливим аспектом розробки програмного забезпечення, оскільки дозволяє підвищити продуктивність, зменшити використання ресурсів та поліпшити ефективність програми. У цьому розділі ми розглянемо стратегії та методи оптимізації коду для програми обробки відеопотоку.

Етапи оптимізації коду.

– Аналіз профілювання. Визначення частин коду, які споживають найбільше ресурсів та потребують оптимізації.

– Вибір оптимальних алгоритмів. Використання ефективних алгоритмів та структур даних для зменшення обсягу обчислень та пам’яті.

– Паралелізація. Розподіл завдань між різними потоками або процесами для використання багатоядерних архітектур та прискорення виконання програми.

– Кешування даних. Зберігання проміжних результатів обчислень у пам'яті для їх подальшого використання та уникнення повторних обчислень.

– Використання оптимізованих бібліотек. Використання спеціалізованих бібліотек та інструментів, які пропонують швидкі та ефективні реалізації алгоритмів.

Стратегії оптимізації коду.

– Локалізація операцій. Виконання обчислень на місці та уникнення зайвих операцій та звернень до пам’яті.

– Мінімізація зайвого коду. Уникнення непотрібних операцій, перевірок та пере присвоєнь для зменшення обсягу коду.

– Використання ефективних структур даних. Використання оптимальних структур даних для зберігання та обробки інформації.

– Оптимізація звернень до пам'яті. Мінімізація звернень до пам'яті та використання локальних змінних та кешованих значень.

– Розпаралелювання завдань. Розподіл завдань між різними потоками або процесами для використання багатоядерних архітектур.

Інструменти для оптимізації коду.

– Профілювальні інструменти: Інструменти, які дозволяють аналізувати час виконання та використання ресурсів програми.

– Інструменти аналізу коду: Інструменти, які дозволяють виявляти недоліки та можливі місця для оптимізації в коді програми.

– Бібліотеки високої продуктивності: Використання спеціалізованих бібліотек для оптимізованих операцій, таких як математичні операції чи обробка відеоданих.

Переваги оптимізації коду.

– Підвищення продуктивності: Зменшення часу виконання програми та зменшення використання ресурсів.

– Економія ресурсів: Зменшення використання пам'яті та процесорного часу, що дозволяє ефективніше використовувати обмежені ресурси.

– Покращення користувацького досвіду: Зменшення часу очікування та поліпшення відзивчивості програми.

3.7 Висновки до розділу 3

Розглянуто інструменти та технології для глибокого навчання, які використовуються в програмному забезпеченні. Було виявлено, що для ефективного використання глибокого навчання необхідно використовувати потужні обчислювальні ресурси та спеціалізовані бібліотеки та фреймворки.

Розглянуто процес перевірки функціональності програмного забезпечення. Було виявлено, що для ефективної перевірки функціональності необхідно розробити тестові випадки, що охоплюють всі можливі сценарії використання програми.

Розглянуто процес проведення тестування програмного забезпечення. Було виявлено, що для ефективного тестування необхідно використовувати різні види тестів, такі як модульні тести, інтеграційні тести та системні тести.

Розглянуто процес виявлення та виправлення помилок в програмному забезпеченні. Було виявлено, що для ефективного виявлення та виправлення помилок необхідно використовувати систему управління помилками та дотримуватися принципів відкритості та відповідальності.

Розглянуто процес виявлення та виправлення помилок в програмному забезпеченні. Було виявлено, що для ефективного виявлення та виправлення помилок необхідно використовувати систему управління помилками та дотримуватися принципів відкритості та відповідальності.

ВИСНОВКИ

Досліджено розвиток технологій комп'ютерного зору, що дозволило окреслити їх значення в сучасних інформаційних системах. Встановлено, що розвиток алгоритмів машинного навчання та глибокого навчання сприяє покращенню точності та швидкості обробки зображень. Показано, що комп'ютерний зір стає невід'ємною частиною багатьох галузей, таких як медицина, автомобілебудування, промисловість та безпека.

Розглянуто методи виявлення форм та фільтрації зображень, які є основними компонентами систем комп'ютерного зору. Проаналізовано ефективність різних алгоритмів, таких як детектори країв та контурів. Виявлено, що застосування методів фільтрації дозволяє покращити якість зображень та підвищити точність розпізнавання об'єктів. Простежено, що вибір алгоритму залежить від конкретних вимог до системи та типу зображень.

Описано процес налаштування та встановлення програмного забезпечення для обробки зображень з використанням OpenCV-Python. Визначено, що коректне налаштування середовища розробки та встановлення всіх необхідних бібліотек є критичними для успішної роботи системи. Показано, що OpenCV-Python забезпечує зручний та ефективний інтерфейс для роботи з функціями та алгоритмами комп'ютерного зору, що спрощує розробку додатків.

Розглянуто основний цикл програми, який є ключовим для безперервної обробки відеопотоку з вебкамери в реальному часі. Під час розгляду було виявлено, що цей цикл виконує кілька важливих завдань, таких як захоплення кадрів, обробка зображень та взаємодія з користувачем через графічний інтерфейс. Аналізуючи основний цикл програми, ми зрозуміли, що він дозволяє виконувати обробку зображень у реальному часі, що є важливим для багатьох сучасних технологій, таких як автономні транспортні засоби, медична діагностика та автоматизований контроль якості виробництва.

Досліджено взаємодію програми з користувачем через графічний інтерфейс. Виявлено, що ця взаємодія є важливою для забезпечення користувачам зручного та ефективного способу керування програмою. Аналізуючи розділ, ми зрозуміли, що

взаємодія з користувачем є необхідною складовою будь-якої програми, оскільки вона дозволяє користувачам взаємодіяти з програмним забезпеченням та виконувати потрібні дії.

Розглянуто функціональність кнопок у веб-додатку для розпізнавання геометричних елементів зображень. Виявлено, що кнопки виконують різні функції, такі як запуск або призупинення обробки зображень, зміна параметрів фільтрів та інші. Аналізуючи цей розділ, ми зрозуміли, що функціональність кнопок є важливою для забезпечення користувачам зручного та ефективного способу взаємодії з програмою.

Розглянуто інструменти та технології для глибокого навчання, які використовуються в програмному забезпеченні. Було виявлено, що для ефективного використання глибокого навчання необхідно використовувати потужні обчислювальні ресурси та спеціалізовані бібліотеки та фреймворки.

Розглянуто процес перевірки функціональності програмного забезпечення. Було виявлено, що для ефективної перевірки функціональності необхідно розробити тестові випадки, що охоплюють всі можливі сценарії використання програми.

Розглянуто процес проведення тестування програмного забезпечення. Було виявлено, що для ефективного тестування необхідно використовувати різні види тестів, такі як модульні тести, інтеграційні тести та системні тести.

Розглянуто процес виявлення та виправлення помилок в програмному забезпеченні. Було виявлено, що для ефективного виявлення та виправлення помилок необхідно використовувати систему управління помилками та дотримуватися принципів відкритості та відповідальності.

Розглянуто процес виявлення та виправлення помилок в програмному забезпеченні. Було виявлено, що для ефективного виявлення та виправлення помилок необхідно використовувати систему управління помилками та дотримуватися принципів відкритості та відповідальності.

ПЕРЕЛІК ПОСИЛАНЬ

1. Matan O. Handwritten Character Recognition Using Neural Network Architectures. / O. Matan, R. Kiang, C. Stenard, et al. // Proceedings of the 4th US Postal Service Advanced Technology Conference. – 1990. – Vol. 9. – P. 1003-1011.
2. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение: пер. с англ. 2-е изд., испр. М.: ДМК Пресс, 2018. 652 с.
3. Шапиро Л., Стокман Дж. Компьютерное зрение пер. с англ. 3-е изд. М.: БИНОМ. Лаборатория знаний, 2015. 763 с.
4. Шапиро Л., Стокман Дж. Компьютерное зрение пер. с англ. 3-е изд. М.: БИНОМ. Лаборатория знаний, 2015. 763 с.
5. Serhiienko, A., Chychkarov, Y., Syrmamiikh, I., (2020). Information Technology of Hot-metal Ladle Car Handwritten Numbers Recognition from Photo Image. In: Computer Modeling and Intelligent Systems (CMIS-2020): Proceedings of the Third International Workshop (Zaporizhzhia, April 27 – May 1, 2020 y.). – Zaporizhzhia, 2020, Vol. I–2608, P. 900–912. http://ceur- ws.org/Vol-2608/paper67.pdf
6. Лузан, П. Г., Каленський, А. А., Пащенко, Т. М., Мося, І. А., & Ямковий, О. Ю. (2021). Методичні основи оцінювання якості підготовки фахівців у закладах фахової передвищої освіти: методичний посібник. Житомир: Полісся
7. Сергієнко, В.П., & Кухар, Л.О. (2011). Методичні рекомендації зі складання тестових завдань. Київ: НПУ.
8. Методичні рекомендації з організації тестового контролю освітньопрофесійної підготовки вчителя. (2004). Тернопіль: видавництво ТНПУ ім. В. Гнатюка, 100.
9. Сазонов, О.О., & Волкова, О.В. (2019). Методичні рекомендації з розроблення тестів. Київ: ДУТ.
10. Михайлов, К.М. (2000). Моделювання системи рейтингової оцінки знань. Вісник ХГТУ, 1(7), 343–346.
11. Михайлов, К.М., & Каленбет, Д.В. (2002). Деякі підходи до системи тестування. Вісник ХГТУ, 41, 503–507.

12. Мойсеюк, Н.Є.; Локшина, О.І. (ред.) (2001). Педагогіка: Навчальний посібник. Київ: (б.в.).

13. Огнівчук, Л. М. (2014). Оцінювання навчальних досягнень студентів вищих навчальних закладів на основі компетентністного підходу. Освітологічний дискурс, 3, 154-165. http://nbuv.gov.ua/UJRN/osdys_2014_3_18.

14. Олійник, М.М., & Романенко, Ю.А. (2001). Тест як інструмент кількісної діагностики рівня знань в сучасних технологіях навчання. Донецьк: ДонНУ. Омельяненко, С.В. (2008). Педагогіка: тестові завдання: Навчальний посібник. Київ: Знання,391.

15. Паращенко, Л. І., Леонський, В. Д., & Леонська, Г. І.; Ляшенко, О. І. (ред.). (2006). Тестові технології у навчальному закладі: Методичний посібник. Київ: ТОВ Майстерня книги.

16. ома, 111. Терещук, Г.В. (ред.). (2007). Удосконалення змісту й технологій оцінювання якості підготовки майбутніх фахівців відповідно до вимог Європейської асоціації якості освіти: Матеріали регіонального науково-практичного семінару. Тернопіль: ТНПУ ім. В.Гнатюка.

17. Bloom, B.S. (1956). Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain. New York: Longman, 216.https://www.uky.edu/~rsand1/china2018/texts/Bloom%20et%20al%20Taxonomy%20of%20Educational%20Objec tives.pdf

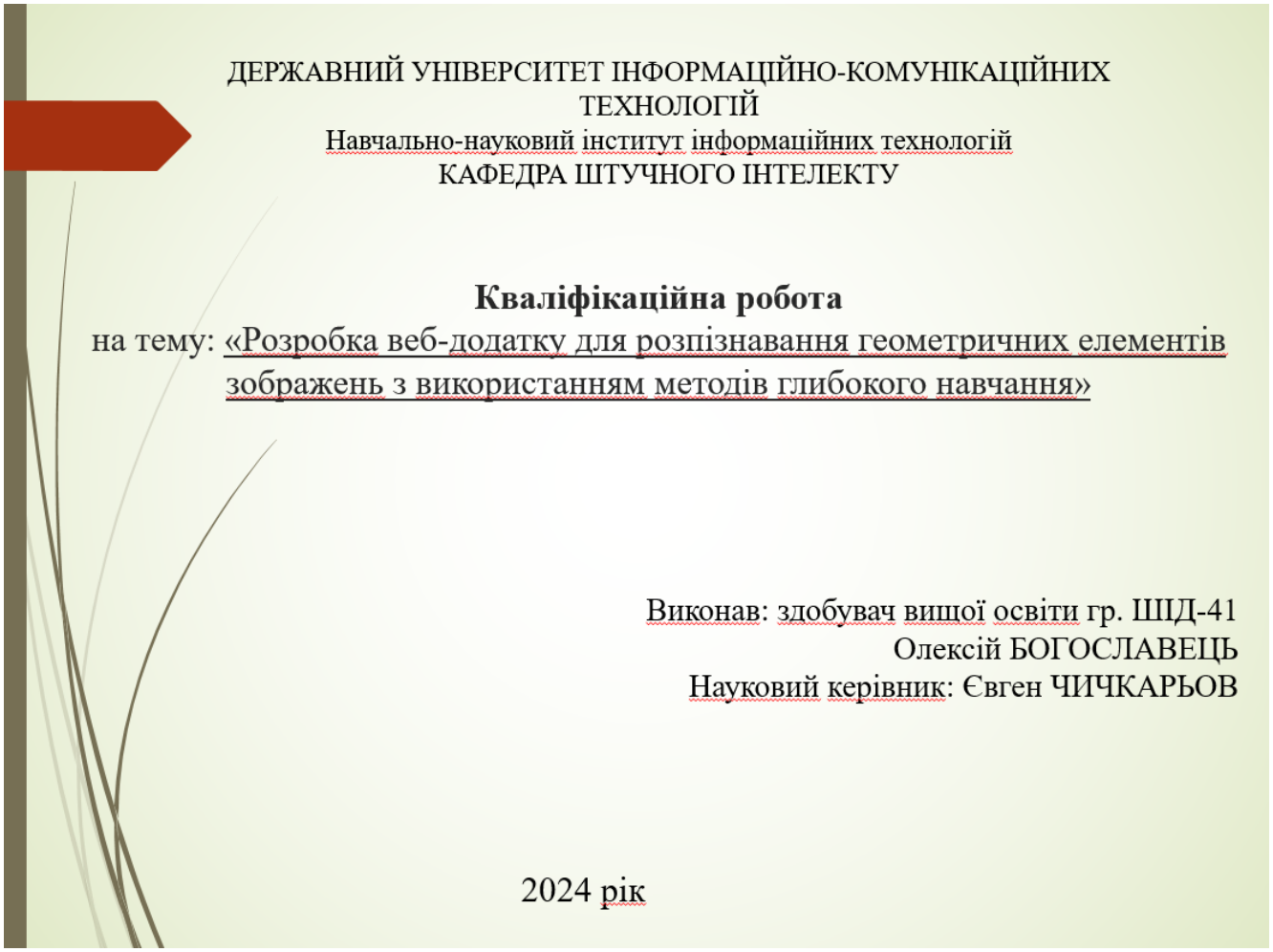
18. Hambleton, R. K., Zaal, J. N., & Pieters, H. J. (1991). Computerized adaptive testing: the ory, applications and standards. Kluwer Academic Publishers. Boston, MA, US.

19. Howard, Wainer. (1990). Computerized Adaptive Testing. A primer. Lawrence Erlbaum Associated, Publishers.

20. Livingston, S. A., & Zieky, M. J. (1982). Passing scores: A manual for setting standards of performance on educational and occupational tests. Princeton, NJ: Educational Testing Service.

21. Mehrens, W. A., & Lehman, I. J. (1991). Measurement and evaluation in education and psychology. New York: Holt; Rinehart &Winston, 3, 592.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



Мета роботи – підвищення ефективності розробки веб-додатку для ефективного розпізнавання геометричних елементів зображень, заснованого на застосуванні методів глибокого навчання.

Об'єкт дослідження – процес розпізнавання геометричних елементів на зображеннях.

Предмет дослідження – методи глибокого навчання, алгоритми комп'ютерного зору та їх реалізація у веб-додатку для розпізнавання геометричних елементів зображень.

Основні завдання кваліфікаційної роботи:

1. Визначення актуальності теми проектування;
2. Проаналізувати методи розпізнавання геометричних елементів;
3. Визначення вимог до розробки застосунку;
4. Розробка інтерфейсу та графічного відображення застосунку;
5. Проведення тестування застосунку.

Архітектура виявлення аномалій штучним інтелектом в реальному часі

Convolutional Neural Network (конволюційна нейронна мережа) є одним з найважливіших інструментів у глибокому навчанні та обробці зображень. Ці нейронні мережі використовуються для автоматичного впізнавання та класифікації зображень, а також для багатьох інших завдань у сферах комп'ютерного зору, обробки природних мов та інших областях.

Основна ідея CNN полягає у використанні шарів, що виконують операцію згортки (convolution) над вхідними даними. Ці шари фільтрують вхідні зображення за допомогою невеликих фільтрів (як правило, квадратних матриць), які рухаються по зображенню та виконують операцію згортки. Після цього застосовуються шари підсемплінгу (pooling), які зменшують розмірність зображення, зберігаючи при цьому важливі ознаки.

Взаємодія користувача

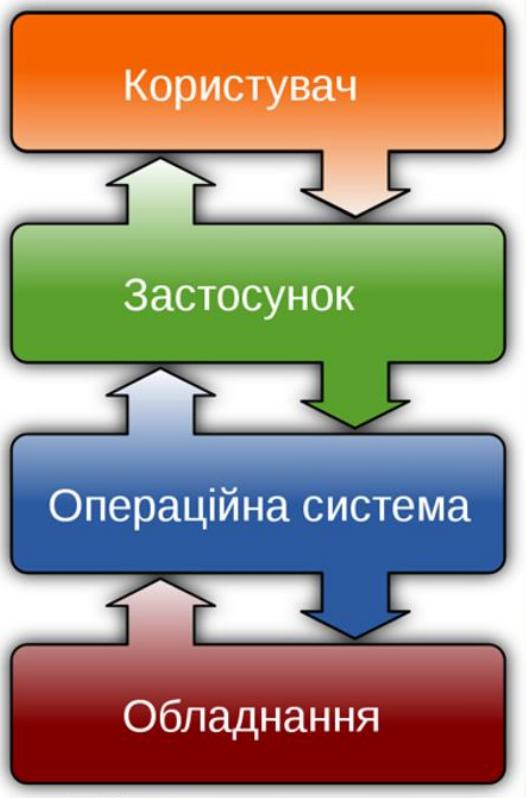


Рис. 1- Взаємодія користувача

```
while True:
    ret, frame = cap.read()
    if not ret:
        break

    lower_black = np.array([0, 0, 0])
    upper_black = np.array([180, 255, 50])
    lower_blue = get_trackbar_values("Frame")[:3]
    upper_blue = get_trackbar_values("Frame")[3:]

    mask_black = cv2.inRange(cv2.cvtColor(frame, cv2.COLOR_BGR2HSV), lower_black
    mask_blue = cv2.inRange(cv2.cvtColor(frame, cv2.COLOR_BGR2HSV), lower_blue,
    mask = cv2.bitwise_or(mask_black, mask_blue)
```

Рис. 2- Основний цикл

```
if current_mode == "shape":
    process_shape_mode(frame, mask)
    mode_text = "Mode: Shape Detection"
else:
    filtered_frame = process_filter_mode(frame, mask)
    frame = filtered_frame
    mode_text = "Mode: Filter"
```

Рис. 3- Вибір режиму роботи та обробка кадру

Види форматів збереження даних

У програмі обробки відеопотоку можуть використовуватися різні формати для збереження даних:

- **Текстові файли:** Вони зручні для збереження текстової інформації, такої як логи, конфігураційні файли або результати обробки.
- **Бінарні файли:** Це ефективний спосіб зберігання структурованих даних у бінарному форматі, що забезпечує економію місця та швидкість доступу.
- **Зображення:** Зберігання кадрів відеопотоку у форматі зображень, таких як PNG або JPEG, дозволяє зручно архівувати та обробляти візуальну інформацію

```python
# Обробник події для миші
def mouse_click(event, x, y, flags, param):
    global save_image
    if event == cv2.EVENT_LBUTTONDOWN:
        # Перевірка чи було натиснуто кнопку "Зберегти"
        if 650 <= x <= 780 and 250 <= y <= 300:
            save_image = True

# Головний цикл програми
while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Обробка введених даних та відображення результатів

    # Відображення кнопки "Зберегти"
    cv2.rectangle(frame, (650, 250), (780, 300), (255, 255, 0), -1)
    cv2.putText(frame, 'Save', (660, 285), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,

    # Збереження зображення при натисканні кнопки "Зберегти"
    if save_image:
        cv2.imwrite(f"saved_image_{datetime.datetime.now().strftime('%Y%m%d_%H%M
        save_image = False

    cv2.imshow("Frame", frame)

    if exit_program or cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Рис. 4- Збереження зображення

Архітектура Штучної Нейронної Мережі

Архітектура Штучної Нейронної Мережі (ШНМ) для розпізнавання цифр зазвичай базується на згорткових нейронних мережах (Convolutional Neural Networks, CNN). Ця архітектура використовується для автоматичного впізнавання рукописних цифр зображень.

Вона містить кілька згорткових шарів, які використовуються для виявлення різних властивостей зображень, таких як лінії та форми. Після цього використовуються пулінгові шари для зменшення розміру зображення та зменшення обчислювальних витрат. На завершення, дані передаються через повнозв'язні шари для класифікації цифр.

Ця архітектура дозволяє досягти високої точності розпізнавання цифр та широко використовується у сучасних системах розпізнавання образів.

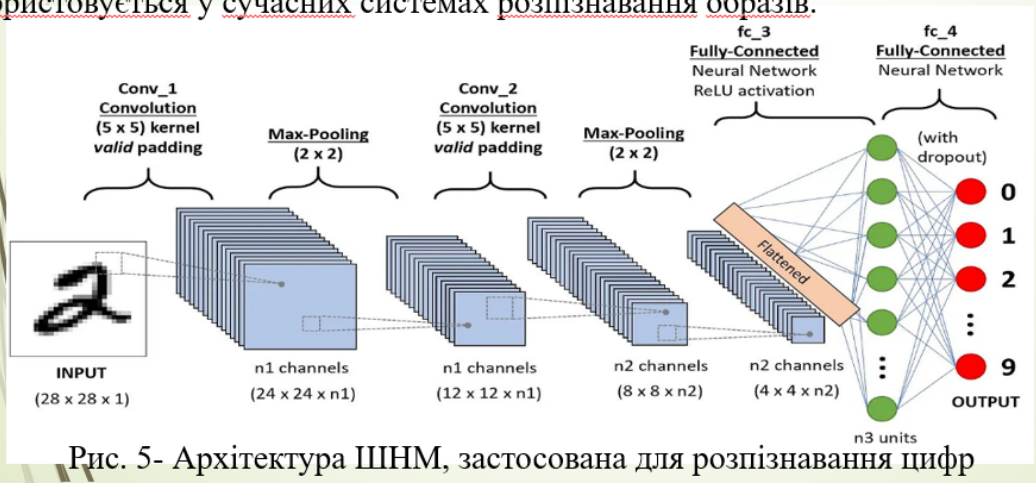


Рис. 5- Архітектура ШНМ, застосована для розпізнавання цифр

Діаграма прецедентів

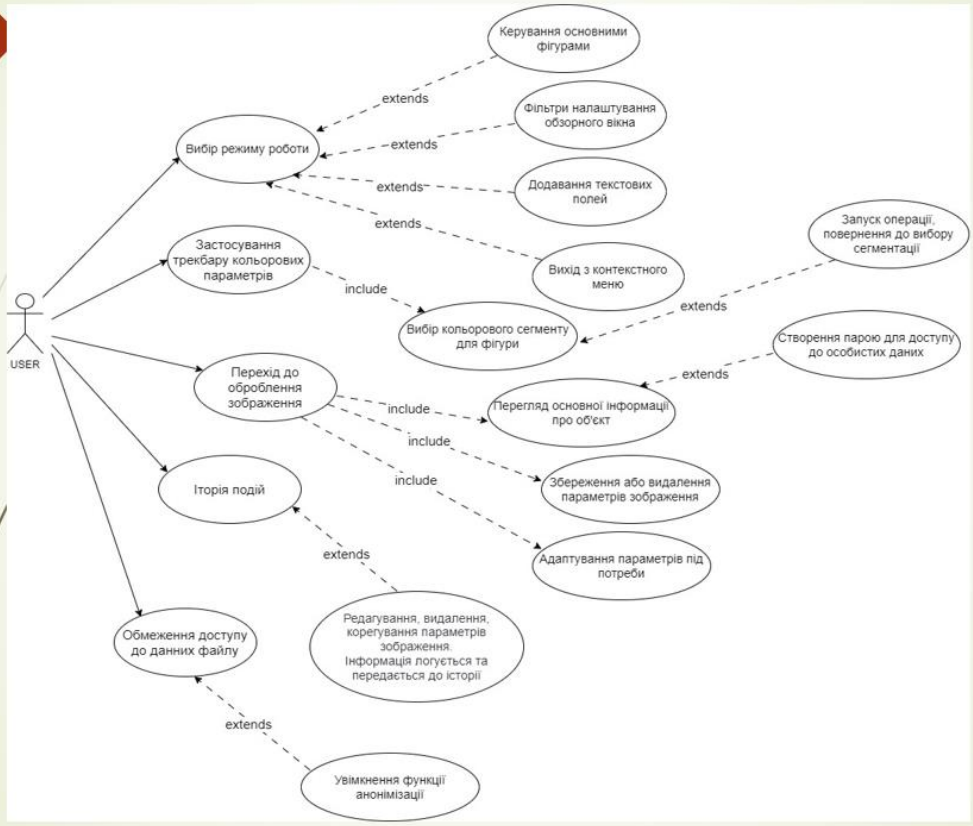


Рис. 6- Діаграма прецедентів

Діаграма діяльності

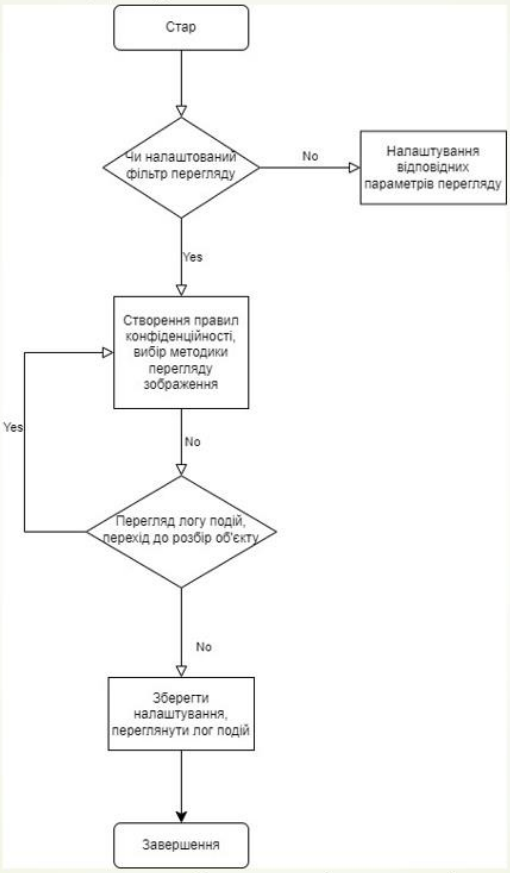


Рис. 7- Діаграма діяльності

ВИСНОВКИ

- Досліджено розвиток технологій комп'ютерного зору, що дозволило окреслити їх значення в сучасних інформаційних системах. Встановлено, що розвиток алгоритмів машинного навчання та глибокого навчання сприяє покращенню точності та швидкості обробки зображень.
- Розглянуто інструменти та технології для глибокого навчання, які використовуються в програмному забезпеченні.
- Розглянуто процес перевірки функціональності програмного забезпечення. Було виявлено, що для ефективної перевірки функціональності необхідно розробити тестові випадки, що охоплюють всі можливі сценарії використання програми.
- Розглянуто процес проведення тестування програмного забезпечення. Було виявлено, що для ефективного тестування необхідно використовувати різні види тестів, такі як модульні тести, інтеграційні тести та системні тести.
- Розглянуто процес виявлення та виправлення помилок в програмному забезпеченні. Було виявлено, що для ефективного виявлення та виправлення помилок необхідно використовувати систему управління помилками та дотримуватися принципів відкритості та відповідальності.

Дякую за увагу!
Доповідь закінчено