ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**Розробка програмного забезпечення для вирішення окремих задач розпізнавання зображень на основі глибоких нейронних мереж мовою Python**»

Виконав: студент 5 курсу, групи ППЗ–51
спеціальності
121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Студент Щеголь А. Г.

(прізвище та ініціали)

Керівник Треньов І.М.

(прізвище та ініціали)

Рецензент Зінченко О.В.

(прізвище та ініціали)

Нормоконтроль Трінтіна Н.А.

(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпеченняСтупінь вищої освіти -«Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
Негоденко О.В.
« » 2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

ЩЕГОЛЮ АНАТОЛІЮ ГЛІБОВИЧУ
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка програмного забезпечення для вирішення окремих задач розпізнавання зображень на основі глибоких нейронних мереж мовою Python»

Керівник роботи: Треньов М.Г., асистент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №22.
2. Строк подання студентом роботи «1» червня 2023 року
3. Вхідні дані до роботи
Методи створення правильного user experience;
Науково-технічна література з питань, пов'язаних з дослідженням і подальшої оптимізації параметрів інтерфейсу та подальшого протипування додатку;

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).
4.1 Поняття ui/ux дизайну та його роль в сучасних веб-застосунках.
4.2 Вимоги до інтерфейсу користувача.
4.3 UX-проектування та протипуання інтерфейсу користувача.
4.4 Тестування користувацького інтерфейсу.

5. Перелік демонстраційного матеріалу (назва основних слайдів)

1. Мета, об’єкт та предмет дослідження
2. Аналіз існуючих аналогів
3. Програмні засоби реалізації
4. Аналіз існуючих веб-сервісів університету (ДУТ)
5. Діаграма варіантів використання
6. Алгоритм тренування моделі
7. Архітектура моделі RESNET-50 (CNN)
8. Діаграма класів
9. Тестування інтерфейсу користувача
10. Дякую за увагу

6. Дата видачі завдання «25» лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	25.02.23 – 27.02.23	Виконано
2	Дослідження аналогів	05.04.2023	Виконано
3	Аналіз та вибір інструментів для розробки прототипу	06.04.2023	Виконано
4	Дослідження та прототипування	07.04.2023	Виконано
5	Вступ, висновки, реферат	17.04.2023	Виконано
6	Розробка обов’язкових демонстраційних матеріалів	18.04.2023	Виконано
7	Попередній захист роботи	26.05.2023	Виконано
8	Здача роботи	01.06.2023	Виконано

Студент _____ Щеголь А.Г.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Треньов М.Г.
(підпис). (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 58 с. 32 рис., 15 джерел.

Ключові слова: DATA, DATA PRE-PROCESSING, NETWORK ARCHITECTURE, MODEL TRAINING AND VALIDATION, RESULTS VISUALIZATION, PYTHON, AI, IMAGE RECOGNITION, DEEP NEURAL NETWORKS, MACHINE LEARNING

Об'єкт дослідження: процес розпізнавання об'єктів на зображеннях за допомогою глибоких нейронних мереж. Цей процес є актуальним у багатьох сферах, таких як комп'ютерне зорове сприйняття, медична діагностика, автономні автомобілі, робототехніка та інші.

Предмет дослідження: програмне забезпечення, призначене для розпізнавання об'єктів на зображеннях за допомогою глибоких нейронних мереж. Для реалізації цієї мети використовується мова програмування Python, яка має широкі можливості у галузі машинного навчання та обробки зображень.

Мета роботи: покращенні процесу розпізнавання об'єктів на зображеннях за допомогою глибоких нейронних мереж, реалізованих мовою Python. Дослідження спрямоване на розробку програмного забезпечення, яке забезпечить ефективне вирішення задач розпізнавання об'єктів на зображеннях з використанням потужних інструментів глибокого навчання.

Методи дослідження - Аналіз літератури, збір та підготовка даних, реалізація глибоких нейронних мереж, навчання та валідація моделі.

Програмне забезпечення, розроблене в рамках даної роботи, має на меті забезпечити точне та швидке розпізнавання об'єктів на зображеннях. Застосування глибоких нейронних мереж дозволяє досягти високої точності та надійності в розпізнаванні об'єктів, забезпечуючи значну перевагу перед традиційними методами.

Програмне забезпечення може бути використане в різних сферах, де потрібне автоматизоване розпізнавання об'єктів на зображеннях. Наприклад, воно може знайти застосування в системах відеоспостереження, медичних системах діагностики, автоматизованих системах контролю якості, системах безпеки та багатьох інших областях.

ЗМІСТ

ВСТУП

У сучасному світі зростає значущість розпізнавання об'єктів на зображеннях, що знайшло широке застосування в різних сферах, таких як комп'ютерне зорове сприйняття, автономні автомобілі, медична діагностика, безпека та багато інших. Одним з ключових інструментів для досягнення високої точності в розпізнаванні об'єктів є використання глибоких нейронних мереж.

Метою роботи є покращення процесу розпізнавання об'єктів на зображеннях за допомогою глибоких нейронних мереж, що дасть змогу досягти більш точних та ефективних результатів.

У ході роботи вивчені та застосовані сучасні методи глибокого навчання, зокрема з використанням архітектури мережі ResNet, популярних бібліотек, таких як TensorFlow, PyTorch, Keras, а також виконано аналіз літератури та перевірку наявних датасетів. Додатково проведені етапи підготовки даних, тренування та валідації моделі, аналіз отриманих результатів та їх візуалізація.

Результати дослідження та розроблене програмне забезпечення має практичне застосування в областях, де розпізнавання об'єктів на зображеннях є ключовим етапом. Дослідження спрямовано на збільшення ефективності та точності розпізнавання, що має велике значення для різних індустрій та сфер діяльності. Методи дослідження - методи кількісних та якісних досліджень, методи євристичного аналізу, методи проектування інтерфейсу користувача, методи побудування моделі пріоритизації, методи побудови інформаційної архітектури, методи тестування, реєстрації, верифікації та валідації поведінки користувача.

Наукова новизна роботи полягає у наступному:

1. Використання глибоких нейронних мереж для розпізнавання об'єктів на зображеннях. У даній роботі будуть застосовані сучасні архітектури мереж, зокрема ResNet, які відзначаються високою точністю і здатністю до вирішення складних задач розпізнавання.

2. Використання мови програмування Python для розробки програмного забезпечення. Python є широко використовуваною мовою в галузі глибокого навчання, оскільки вона надає зручний синтаксис, багатий набір бібліотек та інструментів для обробки зображень та нейронних мереж.

3. Вивчення та застосування сучасних методів глибокого навчання. У роботі будуть досліджені та використані найновітніші підходи до тренування та оптимізації глибоких нейронних мереж, що сприятиме досягненню кращих результатів у розпізнаванні об'єктів.

4. Аналіз літератури та датасетів. Робота включатиме огляд наукової літератури з питань розпізнавання об'єктів на зображеннях, а також вивчення та аналіз доступних датасетів для тренування та валідації моделі. Це дозволить забезпечити наукову обґрунтованість та достовірність отриманих результатів.

5. Розробка програмного забезпечення. У рамках роботи буде розроблено програмне забезпечення для розпізнавання об'єктів на зображеннях з використанням глибоких нейронних мереж. Це включатиме створення моделей, підготовку даних, тренування та валідацію моделі, а також аналіз та візуалізацію результатів. Практична значущість результатів дослідження:

1. Вдосконалення процесу розпізнавання об'єктів на зображеннях. Застосування глибоких нейронних мереж та розробленого програмного забезпечення дозволить досягти більш точного та швидкого розпізнавання об'єктів у зображеннях. Це може бути корисним у багатьох галузях, включаючи комп'ютерне зору, автоматизоване визначення об'єктів, медичну діагностику та інші області, де потрібна точна і швидка обробка зображень.

2. Покращення якості та ефективності системи обробки зображень. Використання сучасних методів глибокого навчання дозволить досягти більш високої точності та надійності у розпізнаванні об'єктів. Це може бути важливим фактором для систем, які вимагають високої якості обробки зображень, наприклад, в автономних автомобілях, системах безпеки, робототехніці тощо.

3. Застосування в реальному часі. Є важливим для багатьох застосувань, таких як відеоспостереження, віртуальна реальність, взаємодія з розумними пристроями тощо.

4. Переносимість та доступність. Розроблене програмне забезпечення, реалізоване на мові програмування Python, має високу переносимість та доступність для широкого кола користувачів. Це дозволяє використовувати розроблену систему на різних платформах та з різними обчислювальними ресурсами.

1 АНАЛІЗ ЗАДАЧ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

1.1 Визначення постановки задачі розпізнавання зображень

Задача розпізнавання зображень є важливою галуззю комп'ютерного зору, яка полягає в розробці алгоритмів та програмного забезпечення для автоматичного визначення та класифікації об'єктів на цифрових зображеннях. Вона включає в себе використання глибоких нейронних мереж, що є потужними інструментами для розпізнавання образів та вирішення складних завдань у сфері комп'ютерного зору. Задача розпізнавання зображень передбачає навчання моделей комп'ютерного зору для автоматичного виявлення об'єктів на зображеннях та їх класифікації відповідно до заданого набору категорій. Це включає в себе процеси попередньої обробки зображень, витягування ознак, тренування моделей та перевірку їх ефективності на нових невідомих зображеннях.

1.2 Огляд типових задач розпізнавання зображень та їх особливостей

У галузі розпізнавання зображень існує багато типових задач, кожна з яких має свої особливості і вимагає використання специфічних підходів. На рис.1 та рис.2 наведено аналіз аналогів, який включає: класифікацію зображень, локалізацію об'єктів та семантичну сегментацію.

Показники	Google Cloud Vision API	Microsoft Azure Computer Vision	Amazon Rekognition	Clarifai
Розпізнавання об'єктів	+	+	+	+
Класифікація зображень	+	+	+	+
Розпізнавання облич	+	+	+	+
Розпізнавання рукописного тексту	-	-	-	+
Аналіз вмісту зображень	+	+	+	+
Виявлення міток	+	+	+	+

Рис. 1 – Аналіз аналогів

Показники	Google Cloud Vision API	Microsoft Azure Computer Vision	Amazon Rekognition	Clarifai
Виявлення відомих облич	+	+	+	-
Виявлення небажаного вмісту	+	+	+	+
Розпізнавання тексту	+	+	+	+
Аналіз настрою	+	+	+	+

Рис. 2 – Аналіз аналогів

Класифікація зображень полягає в призначенні зображенням однієї або кількох категорій. Наприклад, класифікація зображень тварин може розподілити зображення на категорії "кіт", "собака" або "птах". Локалізація об'єктів передбачає визначення положення та меж області зображення, де знаходиться конкретний об'єкт. Це корисно для виявлення об'єктів, таких як обличчя, автомобілі або дорожні знаки. Семантична сегментація передбачає присвоєння кожному пікселю зображення певного класу або категорії. Це дозволяє точно визначити межі об'єктів на зображенні та здійснювати їх піксельну класифікацію. Задача виявлення об'єктів полягає в знаходженні та відмічанні областей зображення, які містять певні об'єкти. Це використовується для знаходження об'єктів на зображеннях, таких як автомобілі на дорозі або люди в натовпі.

1.3 Аналіз існуючих аналогів

1.3.1 Google Cloud Vision API

Vertex AI Vision - сервіс машинного навчання, що надається Google Cloud, спеціалізується на комп'ютерному зорі, таких як розпізнавання об'єктів, класифікація зображень та сегментація зображень.

Він забезпечує платформу для тренування та розгортання власних моделей з використанням глибоких нейронних мереж.

Vertex AI Vision має декілька ключових особливостей:

Масштабованість: Сервіс розроблений для роботи з великими обсягами даних та ефективно розподіляє тренування моделей між багатьма обчислювальними ресурсами, що дозволяє прискорити тренування моделей. Це важливо для складних задач обробки зображень, які вимагають обробки великої кількості зображень.

Кастомізація: Vertex AI Vision надає можливість навчати власні моделі, пристосовані до конкретних потреб користувача. Сервіс надає різноманітні попередньо побудовані моделі, які можна налаштувати за допомогою трансферного навчання або навчити з нуля, використовуючи користувацькі набори даних. Це дозволяє адаптувати моделі до різних областей та досягати кращої продуктивності у завданнях розпізнавання об'єктів.

Розгортання: Після тренування моделей, їх можна легко розгорнути для використання у реальному часі. Це дозволяє застосовувати навчені моделі для розпізнавання об'єктів на зображеннях у виробничому середовищі. Такі застосування включають автоматичне тегування зображень, візуальний пошук та виявлення об'єктів у відеопотоках.

Vertex AI Vision надає повноцінне рішення для задач розпізнавання об'єктів на зображеннях, поєднуючи потужні алгоритми глибокого навчання, масштабованість, можливості кастомізації та інтеграцію з екосистемою Google Cloud. Сервіс є цінним інструментом для будь-яких додатків, що використовують глибокі нейронні мережі для розпізнавання об'єктів на зображеннях.

1.3.2 Microsoft Azure Computer Vision

Microsoft Azure Computer Vision - це послуга хмарних обчислень від Microsoft, яка надає широкий спектр можливостей комп'ютерного зору. Azure Cognitive Services пропонує ряд функцій для розпізнавання об'єктів на зображеннях, включаючи розпізнавання зображень, виявлення об'єктів та аналіз зображень.

API та служби забезпечують легку інтеграцію функцій розпізнавання об'єктів у програми без значного досвіду машинного навчання.

Крім цього, Azure надає можливість користувачам створювати власні моделі комп'ютерного зору за допомогою служби Custom Vision. Також дозволяє завантажувати та анотувати набори даних, навчати моделі з використанням алгоритмів глибокого навчання та розгортати навчені моделі для виведення висновків. Такий підхід до кастомізації дозволяє створювати системи розпізнавання об'єктів, оптимізовані для конкретних потреб розробників.

Платформа Azure Machine Learning підтримує різні фреймворки глибокого навчання, такі як TensorFlow або PyTorch, що дає розробникам можливість використовувати їх для навчання глибоких нейронних мереж для розпізнавання об'єктів.

Azure Machine Learning надає інструменти для попередньої обробки даних, оцінювання, спрощуючи робочий процес машинного навчання.

Значною та ключовою перевагою Azure є його масштабованість та інтеграція з іншими службами Azure. Він пропонує інфраструктуру та ресурси для роботи з великими обсягами даних та обчислювальними вимогами.

Також інтегрується з Azure Blob Storage для зберігання даних, Azure Functions для безсерверних обчислень та Azure Kubernetes Service для контейнерного розгортання. Це дозволяє розробникам безперешкодно розробляти та розгортати системи розпізнавання об'єктів з використанням глибоких нейронних мереж.

Microsoft Azure Computer Vision надає розширений набір інструментів, сервісів та інфраструктури для розробки додатків комп'ютерного зору.

Завдяки готовим API, можливості створення власних моделей, масштабованій інфраструктурі та інтеграції з іншими службами Azure, стає потужним інструментом для реалізації розпізнавання об'єктів на зображеннях з використанням глибоких нейронних мереж.

1.3.3 Amazon Rekognition

Amazon Rekognition - це хмарний сервіс комп'ютерного зору, що надається Amazon Web Services (AWS). Він пропонує широкий спектр можливостей для аналізу зображень і відео, включаючи виявлення об'єктів і сцен, аналіз облич, виявлення тексту і модерацію зображень.

Основні критерії Amazon Rekognition:

Масштабованість і продуктивність: Amazon Rekognition розроблено для роботи з великими наборами даних та здатний обробляти зображення та відео в режимі реального часу. Він використовує інфраструктуру AWS, що дозволяє програмам ефективно обробляти великі обсяги зображень.

Кастомізація та навчання: Amazon Rekognition дозволяє користувачам навчати власні моделі виявлення об'єктів, використовуючи власні набори даних. Він підтримує методи навчання з перенесенням, які дозволяють точно налаштовувати попередньо навчені моделі або навчати моделі з нуля для розпізнавання конкретних об'єктів.

Розгортання та інтеграція: Amazon Rekognition легко інтегрується з іншими сервісами AWS, такими як Amazon S3 для зберігання даних, Amazon Lambda для безсерверних обчислень та Amazon DynamoDB для управління базами даних. Це дозволяє розробникам створювати складні робочі процеси та використовувати додаткові ресурси та функціональні можливості AWS.

Amazon Rekognition є надійним та масштабованим сервісом для розпізнавання об'єктів та аналізу зображень. Він надає можливість розробникам кастомізувати та навчати моделі, ефективно працювати з великими обсягами даних і легко інтегруватись з іншими сервісами AWS.

Завдяки цим функціональним можливостям, Amazon Rekognition є цінним інструментом для розробки додатків, які вимагають розпізнавання об'єктів на зображеннях з використанням глибоких нейронних мереж.

1.3.4 Clarifai

Clarifai - це платформа машинного навчання, спеціалізована на комп'ютерному зорі, яка пропонує рішення для завдань розпізнавання об'єктів і класифікації зображень з використанням глибоких нейронних мереж. Основні критерії Clarifai включають:

Налаштування: Clarifai надає зручний інтерфейс та інструменти для навчання власних моделей, які можуть бути пристосовані до конкретних потреб користувача. Це включає анотування та маркування наборів даних, методи глибокого навчання та точне налаштування моделей.

Попередньо навчені моделі: Платформа містить колекцію попередньо навчених моделей, які можуть бути використані "з коробки" для широкого спектру завдань розпізнавання об'єктів. Це спрощує розпочаток роботи та покращує продуктивність. API та SDK: Clarifai надає API та SDK на різних мовах програмування, що дозволяє легко інтегрувати можливості платформи у наявні програми та робочі процеси. Це забезпечує зручність та гнучкість для розробників.

Масштабованість і продуктивність: Clarifai розроблений для ефективної роботи з великими обсягами даних. Він може обробляти і аналізувати велику кількість зображень, що робить його підходящим для додатків, які потребують надійного та масштабованого розпізнавання об'єктів.

Загалом, Clarifai надає комплексне рішення для задач розпізнавання об'єктів з використанням глибоких нейронних мереж. Налаштування моделей, попередньо навчені моделі, API та SDK, а також масштабованість роботи роблять його потенційним вибором для розробників, які зацікавлені в розпізнаванні об'єктів на зображеннях з використанням глибоких нейронних мереж.

2 ТЕХНОЛОГІЇ РЕАЛІЗАЦІЇ

2.1 Вимоги до програмного забезпечення

Python - це універсальна і широко використовувана мова програмування, яка може бути ефективно використана в проекті, орієнтованому на розпізнавання об'єктів за допомогою глибоких нейронних мереж.

Простота та виразність Python сприяють швидшій розробці та експериментам, що робить її ідеальною для створення прототипів та досліджень у проектах з розпізнавання об'єктів.

Python чудово інтегрується з іншими технологіями та фреймворками. Вона легко взаємодіє з бібліотеками глибокого навчання, інструментами обробки зображень та API для пошуку і зберігання даних. Ця можливість інтеграції дозволяє легко інтегрувати етапи попередньої обробки, методи доповнення даних і завдання постобробки в конвеєри розпізнавання об'єктів.

Таким чином, багата екосистема Python це зручність для читання, підтримка спільноти, можливості інтеграції та варіанти розгортання роблять її чудовим вибором для розробки проектів, орієнтованих на розпізнавання об'єктів за допомогою глибоких нейронних мереж.

Архітектура програмного забезпечення - це відповідальний процес, що полягає в проектуванні структури програмного забезпечення з урахуванням його функцій та взаємодії між його компонентами. Вона включає у себе визначення абстрактних компонентів, їх функціональних взаємозв'язків, розміщення та взаємодії між ними.

Архітектура програмного забезпечення підтримує вимоги до системи та забезпечує гнучкість та розширюваність, тому що її компоненти можуть бути змінені чи замінені без впливу на інші частини системи. Окрім того, архітектура програмного забезпечення допомагає забезпечити якість та ефективність розробки.

Для розробки програмного забезпечення, що підтримує прийняття рішень при виборі освітньої програми, можна використовувати архітектуру, що базується на моделі MVC (Model-View-Controller).

Ця модель дозволяє розділити програму на три окремі компоненти, які взаємодіють між собою:

1. Модель (Model) - це компонент, який відповідає за обробку даних та бізнес- логіку програми. Вона може включати базу даних, логіку обробки даних та алгоритми прийняття рішень.

2. Представлення (View) - це компонент, що відображає дані та результати обчислень користувачеві. Цей компонент може включати графічний інтерфейс користувача, веб-інтерфейс чи інші інтерфейси.

3. Контролер (Controller) - це компонент, що відповідає за зв'язок між моделлю та представленням. Він обробляє вхідні дані, передає

їх моделі для обробки та отримує відповідні дані з моделі для відображення користувачу через представлення.

Контролер взаємодіє з користувачем через представлення, отримуючи від нього запити на обробку даних та передаючи їх до моделі. Після отримання результату від моделі, контролер знову взаємодіє з представленням, передаючи йому необхідні дані для відображення результату користувачу.

Крім того, контролер може відповідати за перехід між різними сторінками та екранами програми, виконуючи необхідні дії для збереження стану програми та переходу до нового екрану.

Основна функціональність контролера повинна включати:

- Обробка вхідних даних від користувача
- Взаємодія з моделлю для обробки даних
- Взаємодія з представленням для відображення результатів
- Керування станом програми та перехід між екранами

Контролер повинен мати відповідну архітектуру, щоб забезпечити зручну розробку та підтримку. Також використовують паттерн проектування Model-View-Controller (MVC), що дозволяє розділити логіку програми на три основні компоненти: модель, представлення та контролер, що дозволяє зберігати їх незалежними та легко змінювати окремі компоненти без впливу на інші.

Архітектура програмного забезпечення (АПЗ) для розробки може бути побудована на основі Model-View-Controller (Модель-Вид-Контролер) або подібних архітектурних шаблонів.

Модель-Вид-Контролер (MVC) - це архітектурний шаблон, що відділяє компоненти програми на три окремі складові: модель, яка представляє дані та бізнес- логіку, вид, який відображає дані моделі користувачеві та контролер, який обробляє вхідні дані та забезпечує зв'язок між моделлю та видом. Цей підхід сприяє зменшенню залежності між складовими та полегшує розробку та модифікацію програмного забезпечення.

У контексті розробки програмного забезпечення підтримки прийняття рішень при виборі освітньої програми мовою Python, модель може включати базу даних з інформацією про доступні освітні програми, їх характеристики та вимоги. Вид може бути реалізований у вигляді веб-інтерфейсу, що дозволяє користувачеві вибирати критерії вибору та отримувати рекомендації щодо підходящих освітніх програм. Контролер може включати алгоритми для обробки вхідних даних користувача та взаємодії з моделлю для отримання необхідної інформації та підготовки рекомендацій.

Крім того, для забезпечення ефективності та масштабовності програмного забезпечення можна використовувати інші архітектурні підходи, такі як мікросервісна архітектура.

Зараз відбувається збільшення обсягів даних та їх складності, тому розробка архітектури програмного забезпечення є критично важливою. Вона дозволяє створити структуру програмного продукту, яка забезпечує ефективну роботу з даними, взаємодію з користувачем та іншими системами.

Одним з ключових аспектів архітектури є вибір підходу до розробки програмного продукту. Наприклад, монолітний підхід передбачає створення єдиного блоку програмного забезпечення, що містить усі компоненти системи. Цей підхід є простим та зручним для малих проектів, але може стати неефективним при збільшенні обсягу даних.

Інший підхід - мікросервісна архітектура, що передбачає створення незалежних компонентів програмного забезпечення, які взаємодіють між собою за допомогою API. Цей підхід дає більшу гнучкість та можливість масштабування системи, але потребує більшої складності в розробці та управлінні системою.

Для розробки системи підтримки прийняття рішень, важливо вибрати архітектуру, що найкраще відповідає вимогам проекту. Також, варто враховувати принципи модульності, зручності розширення та тестування, а також ефективність роботи з даними та забезпечення безпеки системи.

Для розробки такої архітектури рекомендаційної системи можна використовувати підходи мікросервісної архітектури. У цьому випадку, кожен компонент системи буде представлений окремим мікросервісом, який буде відповідати за свої функціональні можливості. Наприклад, можна виділити окремі мікросервіси для збереження даних про користувачів, товари, історії їх взаємодії, алгоритми рекомендацій тощо.

Кожен мікросервіс можна буде розробляти та масштабувати окремо від інших компонентів системи. За необхідності, зможуть бути додані нові мікросервіси, або вже існуючі можна буде змінювати без впливу на решту системи.

Також для забезпечення взаємодії між мікросервісами можна використовувати протокол RESTful API, який дозволяє створювати простий та ефективний інтерфейс для обміну даними між компонентами системи.

Загальна архітектура може бути побудована на основі моделі "клієнт-сервер". Клієнтська частина може бути реалізована у вигляді веб-інтерфейсу або мобільного додатку, який буде взаємодіяти з серверною частиною через RESTful API. Серверна частина, у свою чергу, буде складатися з окремих мікросервісів, кожен з яких буде виконувати свою функцію.

Для забезпечення безпеки можна використовувати протокол HTTPS, а також механізми аутентифікації та авторизації, що дозволять забезпечити доступ до даних тільки авторизованим користувачам.

2.2 PyCharm

PyCharm - інтегроване середовище розробки (IDE) для Python, що надає засоби для ефективного створення програмного забезпечення. Використовується у дипломній роботі для розробки програмного забезпечення для розпізнавання зображень на основі глибоких нейронних мереж з використанням Python. Забезпечує інтеграцію з науковими бібліотеками, зручну налагоджувач, автодоповнення та перевірку коду, підтримку систем контролю версій та доступ до розширень через плагіни.

2.3 Numpy

Numpy - це бібліотека для мови Python, яка надає засоби для роботи з масивами та математичними операціями. Вона використовується у дипломній роботі для розробки програмного забезпечення для розпізнавання зображень на основі глибоких нейронних мереж з використанням Python. Numpy дозволяє ефективно працювати з масивами даних, виконувати математичні операції, включаючи броадкастінг, та інтегрується з іншими бібліотеками, що дозволяє легко реалізувати алгоритми розпізнавання зображень на основі глибоких нейронних мереж.

2.4 TensorFlow

TensorFlow - це популярна бібліотека для машинного навчання та глибокого навчання, розроблена компанією Google. Вона використовується у дипломній роботі для розробки програмного забезпечення з метою розпізнавання зображень на основі глибоких нейронних мереж за допомогою мови програмування Python. Інтерфейс TensorFlow і еталонна реалізація були відкритими за ліцензією Apache 2.0, і система доступна для завантаження на www.tensorflow.org. Система включає детальну документацію, низку навчальних посібників і низку прикладів, що демонструють, як використовувати систему для різноманітних завдань машинного навчання.

Інтерфейс TensorFlow і еталонна реалізація були відкритими за ліцензією Apache 2.0, і система доступна для завантаження на www.tensorflow.org. Система включає детальну документацію, низку навчальних посібників і низку прикладів, що демонструють, як використовувати систему для різноманітних завдань машинного навчання. Приклади включають моделі для класифікації рукописних цифр із набору даних MNIST («привіт, світ» алгоритмів машинного навчання), класифікації зображень із набору даних CIFAR-10, виконання мовного моделювання за допомогою повторюваних оренда мережі LSTM, навчальні вектори вбудовування слів [8].

Система включає інтерфейси для визначення обчислень Tensor-Flow у Python і C++, з часом будуть додані інші інтерфейси у відповідь на бажання як внутрішніх користувачів Google, так і ширшої спільноти з відкритим кодом.

Досить багато моделей машинного навчання перенесли на TensorFlow. Зокрема, зосереджуємося на перенесенні найсучаснішої згорткової нейронної мережі для розпізнавання зображень під назвою Inception. Ця система розпізнавання зображень класифікує зображення розміром 224 × 224 пікселя за однією з 1000 міток (наприклад,
«гепард», «сміттєвоз» тощо).

Така модель містить 13,6 мільйона параметрів, які можна вивчати, і 36 000 операцій, виражених у вигляді графа Tensor-Flow. Виведення на одному зображенні вимагає 2 мільярдів операцій множення-додавання.

Після побудови всіх необхідних математичних операцій у TensorFlow зібрати та налагодити всі 36 000 операцій у правильну структуру графа виявилося складним завданням. Перевірка правильності є складним завданням, оскільки система за своєю суттю є стохастичною і має намір поводитися певним чином лише в очікуванні — можливо, після годин обчислень.

Враховуючи ці обставини, ми визнали наступні стратегії критично важливими для перенесення моделі Inception на TensorFlow:

1. Створіть інструменти, щоб отримати уявлення про точну кількість параметрів у даній моделі.

Такі інструменти демонструють виявлені тонкі недоліки в специфікації архітектури складної мережі. Зокрема, ми змогли ідентифікувати операції та змінні, створені неправильно через автоматичну трансляцію в математичній операції через вимір.

2. Почніть з малого та збільшуйте масштаб. Перша згорточна нейронна мережа, яку ми перенесли з нашої попередньої системи, була невеликою мережею, використаною на наборі даних CIFAR-10. Налагодження такої мережі з'ясувало тонкі граничні випадки в окремих операціях (наприклад, максимальне об'єднання) у системі машинного навчання, які було б практично неможливо розшифрувати в більш складних моделях.

3.Завжди переконайтеся, що мета (функція втрат) збігається між системами машинного навчання, коли навчання вимкнено. Встановлення нульової швидкості навчання допомогло нам визначити неочікувану поведінку в тому, як ми випадково ініціалізували змінні в моделі. Таку помилку було б важко виявити в динамічній навчальній мережі.

4. Зробіть відповідність реалізації однієї машини перед тим, як налагоджувати розподілену реалізацію. Ця стратегія допомогла нам окреслити та усунути розбіжності в продуктивності навчання між системами машинного навчання. Зокрема, ми виявили помилки через умови змагання та неатомарні операції, які неправильно вважалися атомарними.

5. Стережіться числових помилок. Числові бібліотеки несумісні в тому, як вони обробляють нескінченні значення з плаваючою комою. Згорткові нейронні мережі особливо сприйнятливі до чисельної нестабільності та мають тенденцію до регулярних розбіжностей під час експериментів та фаз налагодження. Захист від такої поведінки шляхом перевірки нескінченних значень з плаваючою комою дозволяє виявляти помилки в реальному часі на відміну від ідентифікації розбіжної поведінки пост-хок.

6. Проаналізуйте частини мережі та зрозумійте величину чисельної похибки. Запуск підрозділів нейронної мережі паралельно на двох системах машинного навчання надає точний метод гарантувати, що числовий алгоритм є ідентичним у двох системах.

З огляду на те, що такі алгоритми працюють із точністю до числа з плаваючою комою, важливо передбачити та зрозуміти величину очікуваної числової помилки, щоб судити про те, чи правильно реалізовано даний компонент (наприклад, розрізняти «в межах 1e-2, великий !» і «в межах 1e-2: чому це так неправильно?!»).

Перевірка складних математичних операцій у присутності природно стохастичної системи є досить складною. Стратегії, описані вище, виявилися безцінними для здобуття довіри до системи та, зрештою, для створення моделі Inception у TensorFlow. Кінцевим результатом цих зусиль стало 6-кратне підвищення швидкості під час навчання порівняно з нашою існуючою реалізацією моделі DistBelief і таке підвищення швидкості виявилося незамінним у навчанні нового класу моделей розпізнавання зображень більшого масштабу.

Паралельне навчання даних

Одним із простих методів прискорення SGD є розпаралелювання обчислення градієнта для міні-пакетів між елементами міні-пакетів. Наприклад, якщо ми використовуємо розмір міні-пакета з 1000 елементів, ми можемо використовувати 10 реплік моделі, щоб кожна обчислити градієнт для 100 елементів, а потім об’єднати градієнти та застосувати оновлення до параметрів синхронно, щоб бути точно так, якби ми запускали послідовний алгоритм SGD із розміром пакету 1000 елементів. У цьому випадку граф TensorFlow просто містить багато копій тієї частини графіка, яка виконує основну частину обчислень моделі, і один клієнтський потік керує всім циклом навчання для цього великого графіка.

Цей підхід також можна зробити асинхронним, коли граф TensorFlow має багато реплік тієї частини графа, яка виконує основну частину обчислень моделі, і кожна з цих реплік також застосовує оновлення параметрів до параметрів моделі асинхронно. У цій конфігурації існує один клієнтський потік для кожної репліки графа. Цей асинхронний підхід також описано в [10].

Модель паралельного навчання

Паралельне навчання моделі, коли різні частини обчислення моделі виконуються на різних обчислювальних пристроях одночасно для однієї групи прикладів, також легко виразити в TensorFlow. На рисунку 8 показаний приклад рекурентної глибокої моделі LSTM, яка використовується для навчання

послідовності, розпаралеленої на трьох різних пристроях.

Одночасні кроки конвеєрної обробки обчислень моделі

Інший поширений спосіб отримати краще використання для навчання глибоких нейронних мереж — конвеєрне обчислення моделі на тих самих пристроях, виконуючи невелику кількість одночасних кроків на одному наборі пристроїв. Це показано на малюнку 9. Це дещо схоже на асинхронний паралелізм даних, за винятком того, що паралелізм відбувається в межах того самого пристрою (пристроїв), а не копіювання графа обчислень на різних пристроях. Це дозволяє «заповнити прогалини», коли обчислення однієї групи прикладів може бути не в змозі повністю використовувати повний паралелізм на всіх пристроях у будь-який час протягом одного кроку.

Продуктивність

Майбутня версія офіційного документа матиме розділ всебічної оцінки продуктивності як однієї машини, так і розподілених реалізацій. Деякі інструменти працюють разом із основним механізмом виконання графів TensorFlow.

TensorBoard: Візуалізація графових структур і зведеної статистики. Щоб допомогти користувачам зрозуміти структуру їхніх обчислювальних графіків, а також зрозуміти загальну поведінку моделей машинного навчання, створили TensorBoard, додатковий інструмент візуалізації для TensorFlow, який включено до випуску з відкритим кодом.

Візуалізація обчислювальних графіків. Багато обчислювальних графіків для глибоких нейронних мереж можуть бути досить складними. Наприклад, обчислювальний граф для навчання моделі, подібної до моделі Inception від Google глибокої згорткової нейронної мережі, яка мала найкращі показники класифікації на конкурсі ImageNet 2014, має понад 36 000 вузлів у своєму обчисленні TensorFlow.

Граф та деякі глибокі рекурентні моделі LSTM для мовного моделювання мають понад 15 000 вузлів.

Через розмір і топологію цих графіків наївні методи візуалізації часто створюють захаращені та величезні діаграми.

Щоб допомогти користувачам побачити основну організацію графіків, алгоритми в TensorBoard згортають вузли в блоки високого рівня, виділяючи групи з однаковими структурами.

Система також відокремлює вузли високого ступеня, які часто виконують функції бухгалтерського обліку, в окрему область екрана. Це зменшує візуальний безлад і зосереджує увагу на основних розділах графіка обчислень.

Візуалізація є інтерактивною: користувачі можуть панорамувати, масштабувати та розгортати згруповані вузли, щоб детально ознайомитися з ними. Приклад візуалізації для графіка моделі глибокого згорткового зображення наведено на малюнку.

Візуалізація зведених даних.

Під час навчання моделям машинного навчання користувачі часто хочуть мати можливість вивчити стан різних аспектів моделі та те, як цей стан змінюється з часом. З цією метою TensorFlow підтримує набір різни підсумкових операцій, які можна вставити в графік, включаючи скалярні підсумки (наприклад, для вивчення загальних властивостей моделі, таких як значення функції втрат, усереднене за колекцією прикладів, або час, витрачений на виконання обчислювального графіка), підсумки на основі гістограм (наприклад, розподіл значення ваги на рівні нейронної мережі) або підсумки на основі зображень (наприклад, візуалізація вагових коефіцієнтів фільтра, отриманих у згортковій нейронній мережі).

Знімок екрана візуалізації підсумкових значень у TensorBoard показано на рис.2.1.

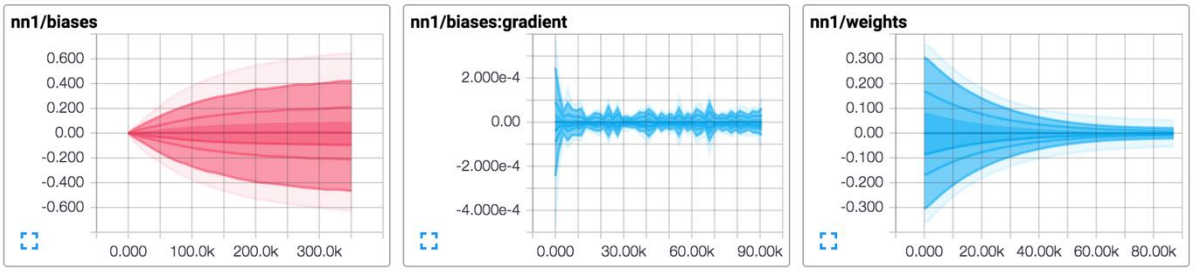


Рис.2.1 – Візуалізації підсумкових значень

Зазвичай обчислювальні графи встановлюються так, що підсумкові вузли включені для моніторингу різноманітних цікавих значень, і час від часу під час виконання навчального графа також виконується набір підсумкових вузлів, на додаток до звичайного набору вузлів, які виконується, а програма-драйвер клієнта записує підсумкові дані до файлу журналу, пов’язаного з навчанням моделі.

Програма TensorBoard налаштована на перегляд цього файлу журналу для отримання нового підсумку записів і може відображати цю підсумкову інформацію та те, як вона змінюється з часом (з можливістю вибору вимірювання «часу» як відносного часу стіни з початку виконання програми TensorFlow, абсолютного часу або «кроків»). ”, числова міра кількості запусків графа, які відбулися з початку виконання програми TensorFlow).

2.5 Matplotlib

Matplotlib - це бібліотека для візуалізації даних у мові програмування Python. Вона використовується в дипломній роботі для створення графіків, діаграм та інших візуальних елементів, що допомагають аналізувати та відображати результати розпізнавання зображень.

2.6 Tkinter

Tkinter - це стандартний набір інструментів для створення графічного інтерфейсу користувача (GUI) у мові програмування Python. Використовується в дипломній роботі для створення інтерактивного інтерфейсу програмного забезпечення для зручного взаємодії з користувачем та візуалізації результатів розпізнавання зображень.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Налаштування програмного середовища та створення структури застосунку

Для розробки програмного забезпечення для вирішення задач розпізнавання зображень на основі глибоких нейронних мереж у мові Python виконуються наступні кроки: встановлення та налаштування середовища Python, створення необхідної структури каталогів та файлів для розробки програмного забезпечення. На рис.3 наведено діаграма класів на рис.3.1 діаграма варіантів використання.

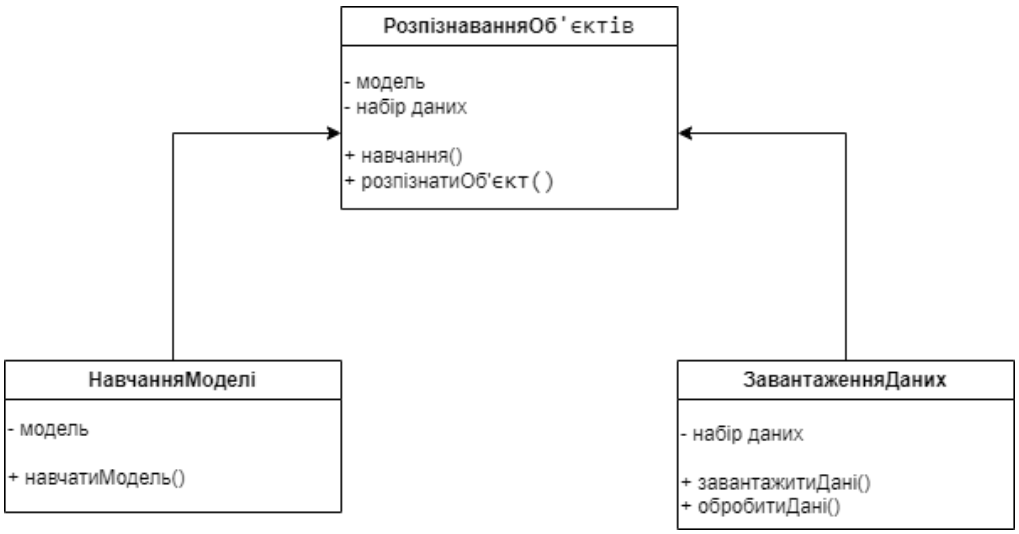


Рис.3 – Діаграма класів

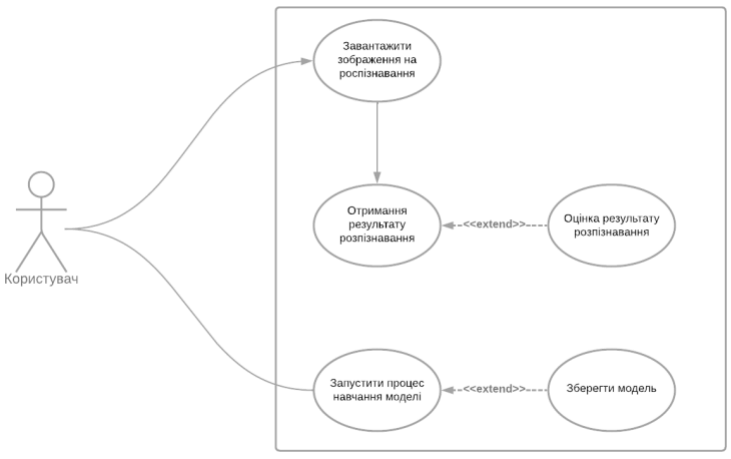


Рис.3.1 – Діаграма варіантів використання

3.2 Підготовка даних для навчання мережі

Для навчання глибокої нейронної мережі використовується датасет "food101" з бібліотеки TensorFlow. Перед підготовкою даних, датасет завантажується напряму з використанням функції tensorflow_datasets.builder("food101"). Після завантаження, дані розбиваються на тренувальний та валідаційний набори. Для покращення якості навчання, дані піддаються попередній обробці, включаючи масштабування, нормалізацію та аугментацію. Процес навчання моделі на наборі даних “food101” в блокноті Jupyter наведено на рис.3.2 як екранні форми.

```
Epoch 1/10
296/296 [==============================] - 290s 948ms/step - loss: 4.6483 - accuracy: 0.0125 - val_loss: 4.6217 - val_accuracy: 0.0149
Epoch 2/10
296/296 [==============================] - 283s 951ms/step - loss: 4.6099 - accuracy: 0.0160 - val_loss: 4.5923 - val_accuracy: 0.0171
Epoch 3/10
296/296 [==============================] - 284s 954ms/step - loss: 4.5862 - accuracy: 0.0186 - val_loss: 4.5707 - val_accuracy: 0.0199
Epoch 4/10
296/296 [==============================] - 292s 980ms/step - loss: 4.5679 - accuracy: 0.0210 - val_loss: 4.5533 - val_accuracy: 0.0240
Epoch 5/10
296/296 [==============================] - 288s 967ms/step - loss: 4.5528 - accuracy: 0.0234 - val_loss: 4.5387 - val_accuracy: 0.0287
Epoch 6/10
296/296 [==============================] - 286s 961ms/step - loss: 4.5397 - accuracy: 0.0254 - val_loss: 4.5258 - val_accuracy: 0.0322
Epoch 7/10
220/296 [=====================>........] - ETA: 54s - loss: 4.5296 - accuracy: 0.0271
```

Рис.3.2 – Екранні форми

Форма спроби розпізнати об’єкти на вибірці зображень наведено на рис.3.3.

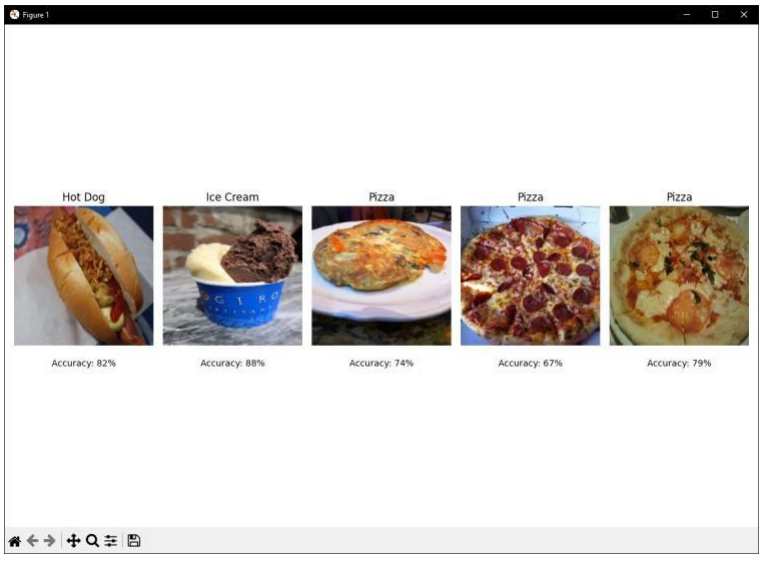


Рис.3.3 – Екранні форми

Форма завантаження, розпізнавання та оцінки передбачення наведено на рис.3.4

Рис.3.4 – Екранні форми

TensorFlow — це інтерфейс для вираження алгоритмів машинного навчання та реалізація для виконання таких алгоритмів. Обчислення, виражене за допомогою TensorFlow, можна виконувати з невеликими змінами або без змін на різноманітних гетерогенних системах, починаючи від мобільних пристроїв, таких як телефони та планшети, і закінчуючи великомасштабними розподіленими системами із сотень машин і тисяч обчислювальних пристроїв, таких як карти GPU.

Система є гнучкою і може використовуватися для вираження широкого спектру алгоритмів, включаючи алгоритми навчання та логічного висновку для моделей глибокої нейронної мережі, і її використовували для проведення досліджень і для розгортання систем машинного навчання у виробництві в більш ніж десятку областей інформатики та інших галузей, включаючи розпізнавання мови, комп’ютерний зір, робототехніку, пошук інформації, обробку природної мови, вилучення географічної інформації та комп’ютерне відкриття ліків. У цьому документі описано інтерфейс TensorFlow і реалізацію цього інтерфейсу, яку ми створили в Google. API TensorFlow і еталонна реалізація були випущені як пакет із відкритим кодом під ліцензією Apache
2.0 у листопаді 2015 року.

Представляємо метод виявлення на основі базової системи Faster R-CNN. Моделі ініціалізуються моделями класифікації ImageNet, а потім точно налаштовуються на даних виявлення об'єктів. На відміну від VGG-16, ResNet не має прихованих fc-шарів. Для вирішення цієї проблеми приймаємо ідею та обчислюємо спільні карти функцій conv для повного зображення, використовуючи ті шари, кроки яких на зображенні не перевищують 16 пікселів (тобто conv1, conv2 x, conv3 x і conv4 x, всього 91 шар conv у ResNet-101.

Розглядаємо ці шари як аналогічні 13 шарам conv у VGG-16, і завдяки цьому ResNet і VGG-16 мають карти функцій conv з однаковим загальним кроком (16 пікселів). Рівні спільно використовують мережа регіональних пропозицій (RPN, що генерує 300 пропозицій) і мережа виявлення Fast R-CNN. Об'єднання ROI виконується перед conv5 1. У цій об'єднаній RoI функції всі рівні conv5 x і вище приймаються для кожного регіону, відіграючи роль FC-рівнів VGG-16. Остаточний шар класифікації замінюється двома однорідними рівнями (класифікація та коробкова регресія).

Для використання шарів BN після попереднього навчання ми обчислюємо статистику BN (середні значення та дисперсії) для кожного шару в наборі для навчання ImageNet. Потім шари BN фіксуються під час тонкого налаштування для виявлення об'єктів. Таким чином, шари BN стають лінійними активаціями з постійними зміщеннями та масштабами, а статистика BN не оновлюється шляхом тонкого налаштування. Ми виправляємо шари BN головним чином для зменшення споживання пам'яті під час навчання Faster R-CNN.

3.3 Реалізація шаблонів web сторінок

Для створення HTML документів та роботи з ними в Django потрібно в середині створеного проекта створити папку «templates», фреймворк сам розуміє що там лежать саме файли для генерації web сторінки.

Всередині цієї папки створимо ще одну задовільну папку, назвемо її «main», після цього є можливість створювати шаблони, вийшло шість сторінок сайту, тобто шість шаблонів, рисунок 3.5.

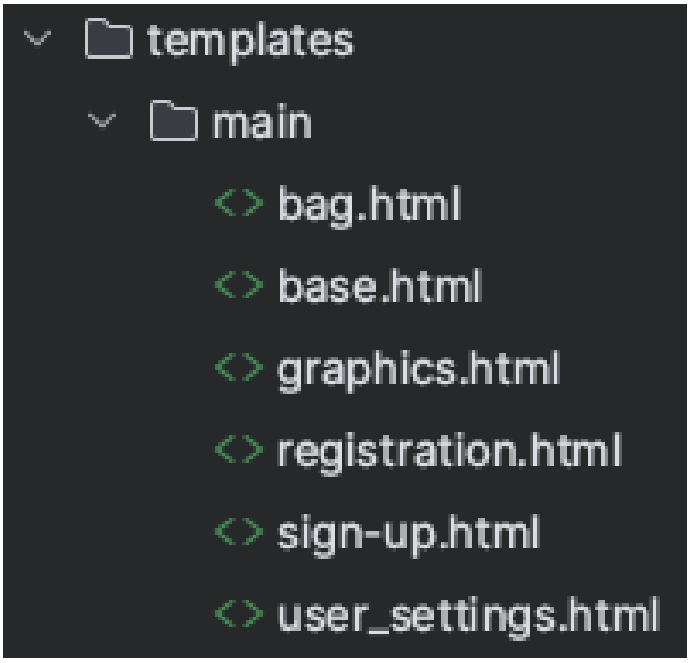


Рисунок 3.5 – папка із HTML шаблонами

Є головний шаблон від якого наслідуються всі інші, і цей шаблон має назву «base.html» всередині кожного із шаблонів використовується шаблонизатор Jinja2, завдяки якому можна робити багато зручних використання одного і того самого блоку коду. Весь базовий повторюваний код, такий як навігаціона полоска зверху, лежить в
«base.html» і відокремлений блок контенту що змінюється на кожній сторінці позначається з використанням Jinja2 кодом, рисунок 3.5.1.

Також за допомогою нього можна використовувати умови, в випадку навігаційної полоски перевіряємо чи користувач що потрапив на сторінку починав сесію з сервером, якщо так, то демонструємо нікнейм, і натиснувши на нього, користувач може переміститися на сторінку редагування профілю, рисунок 3.5.2. Також цикли напряму в HTML документі, при виклику генерації сторінки можемо передавати список або словник на сторінку.

На сторінці вже маніпулювати цими даними, нариклад на головній сторінці виводимо багато однакових елементів, в яких змінюється тільки контент, а сам елемент має одні і ті самі характеристики, щоб зменшити код і постійно не повторювати використовуються цикл, який проходить по кожному елементу зі словника що передається і створює об'єкти на сторінці, рисунок 3.5.3.

Ще додано додатковий блок, що відповідає за назву сторінки, який так само в залежності від сторінки динамічно змінює назву, рисунок 3.5.4.

```
{% block content %}
{% endblock %}
```

Рисунок 3.5.1 – блок контента в файлі «base.html»

```
{% if request.session.user_id %}
<a class="col-2 white_text" href="{% url 'user_settings' %}">{{ request.session.user_id }}</a>
{% else %}
<a class="col-2 btn btn-primary" href="{% url 'sign-up' %}">Sign Up</a>
{% endif %}
```

Рисунок 3.5.2 – використання умов з бібліотеки Jinja2

```
{% for token, info in tokens_with_prices.items %}
    <a href="graphics/{{ token }}" class="col-3 m-2 custom-border">
    <div class="d-flex justify-content-between" style="text-align: center;">
        <div style="padding-top: 10px;">
            <h6 class="white_text" style="border-bottom: 1px dashed white;">{{ token }}</h6>
            <h6 class="white_text">Price: {{ info.0 }}$</h6>
        </div>
        <div style="padding-top: 10px; border-left: 1px dashed white; padding-left: 5px; display: flex; align-items: center; justify-content: center;">
            <h5 class="{% if info.1 > 0 %}positive{% else %}negative{% endif %}" style="margin-left: 10px;">{{ info.1 }}%</h5>
        </div>
    </div>
    </a>
{% endfor %}
```

Рисунок 3.5.3 – використання циклів з бібліотеки Jinja2

```
<title>{% block title %}Main{% endblock %}</title>
```

Рисунок 3.5.4 – блок назви сторінки в файлі «base.html»

Всі наступні шаблони ініціалізуються в середині документу, рисунок 3.5.5, головний шаблон на який посилаються, і все що залишається, лише передавати нові блоки з контентом, назвою сторінки що відповідають за це.

```
{% extends 'main/base.html' %}
```

Рисунок 3.5.5 – наслідування з головного файлу «base.html»

Сторінка входу в системі містить в собі блок контенту що складається зі форми яка була створена в минулому пункті, і додаткові умови, якщо сторінка прийме змінну помилки, створить елемент і попередить користувача, що ввів не валідні дані, також виставляємо поля форми що передаємо, де це потрібно, рисунок 3.5.6.

```
{% block content %}
<body class="text-center bg-dark">
    <form class="form-signin mx-auto col-5" method="post">
        <div class="form-group">
            <h1 class="h3 mb-3 font-weight-normal white_text">Please sign in</h1>
            {% csrf_token %}
            {% if error %}
                <div class="alert alert-danger">
                    {{ error }}
                </div>
            {% endif %}
            {{ loginform.email }}
            {{ loginform.password }}
            <button class="btn btn-lg btn-primary btn-block" type="submit">Login</button>
            <a href="{% url 'registration' %}" class="btn btn-lg btn-primary btn-block" type="submit">Registration</a>
        </div>
    </form>
</body>
{% endblock %}
```

Рисунок 3.5.6 – сторінка входу в аккаунт «sign-up.html»

Сторінка реєстрації має схожу структуру як і сторінка логіна, але містить в собі форму реєстрації та всі поля що відносяться до неї, також умава на перевірку наявності помилок, рисунок 3.5.7.

```
{% block content %}
<body class="text-center bg-dark">
    <form class="form-signin mx-auto col-5" method="post">
        <div class="form-group">
            <h1 class="h3 mb-3 font-weight-normal white_text">Registration</h1>
            {% csrf_token %}
            {% if error %}
                <div class="alert alert-danger">
                    {{ error }}
                </div>
            {% endif %}
            {{ regform.nickname }}
            {{ regform.email }}
            {{ regform.password }}
            {{ regform.repeat_password }}
            <button class="btn btn-lg btn-primary btn-block" type="submit">Registrate</button>
        </div>
    </form>
</body>
{% endblock %}
```

Рисунок 3.5.7 – сторінка реєстрації нового аккаунту «registration.html»

Сторінка налаштування профілю користувача екранізує форму редагування нікнейма користувача, рисунок 3.5.8, форму редагування телеграм акаунту, рисунок 3.5.9, форму редагування пароля, рисунок 3.5.10 та форму редагування API ключів, рисунок 3.5.11. Також від кожної з цих форм виводиться помилка якщо щось пішло не так, і повідомлення про успішні зміни.

```
<form class="form-signin mx-auto col-5" method="post">
    <div class="form-group">
        <h1 class="h3 mb-3 font-weight-normal white_text">Edit nickname</h1>
        {% csrf_token %}
        {% if nickname_error %}
            <div class="alert alert-danger">
                {{ nickname_error }}
            </div>
        {% endif %}
        {{ edit_nickname_form.nickname }}
        <button class="btn btn-lg btn-primary btn-block" type="submit">Save</button>
    </div>
</form>
```

Рисунок 3.5.8 – форма зміни нікнейму в «user_settings.html»

```
<form class="form-signin mx-auto col-5" method="post">
    <div class="form-group">
       <h1 class="h3 mb-3 font-weight-normal white_text">Edit telegram account</h1>
        {% csrf_token %}
        {% if tg_account_error %}
            <div class="alert alert-danger">
                {{ tg_account_error }}
            </div>
        {% endif %}
        {{ edit_tg_account_form.tg_account }}
       <button class="btn btn-lg btn-primary btn-block" type="submit">Save</button>
   </div>
</form>
```

Рисунок 3.5.9 – форма зміни телеграмм аккаунту в «user_settings.html»

```
<form class="form-signin mx-auto col-5" method="post">
    <div class="form-group">
       <h1 class="h3 mb-3 font-weight-normal white_text">Edit password</h1>
        {% csrf_token %}
        {% if password_error %}
            <div class="alert alert-danger">
                {{ password_error }}
            </div>
        {% endif %}
        {{ edit_password_form.old_password }}
        {{ edit_password_form.password }}
        {{ edit_password_form.repeat_password }}
       <button class="btn btn-lg btn-primary btn-block" type="submit">Save</button>
   </div>
</form>
```

Рисунок 3.5.10 – форма зміни пароля в «user_settings.html»

```
<form class="form-signin mx-auto col-5" method="post">
    <div class="form-group">
       <h1 class="h3 mb-3 font-weight-normal white_text">Edit Binance API keys</h1>
        {% csrf_token %}
        {% if key_error %}
            <div class="alert alert-danger">
                {{ key_error }}
            </div>
        {% endif %}
        {{ keys_form.api_key }}
        {{ keys_form.secret_key }}
        {{ keys_form.password }}
       <button class="btn btn-lg btn-primary btn-block" type="submit">Save</button>
   </div>
</form>
```

Рисунок 3.5.11 – форма зміни API ключів в «user_settings.html»

Сторінка детальної інформації про токен та аналітичні дані стосовно нього, на цю сторінку можна потрапити тільки з головної сторінки при натисканні на якийсь з валютних пар на ній. Ця сторінка виводить інформацію про зміни валютної пари, стоїть умова якщо знайшлося ущільнення, то його виведе на екран, рисунок 3.5.12.

Також нижче розташований цикл що за допомогою JavaScript виводить чортири графіка змін валюти за певний проміжок часу, щоб не повторювати код чотири рази, він теж винесений в цикл, рисунок 3.5.13.

Сторінка аналізу криптовалютного гаманця користувача містить в собі дані по результатам аналізу гаманця на біржі. Виводиться побудований графік по результатам змін, це фото документ що передається в бітовому вигляді і вимальовується відразу на сторінці. Також виводиться загальна вартість гаманця користувача і в циклі створюються елементи які в собі містять валютні пари в цьому гаманці і їх вартість на поточний момент, рисунок 3.5.14

```
<h4 class="white_text" style="text-align: center;">Graphics {{ token }} in different timeframes </h4>
<h5 class="white_text" style="margin-bottom: 20px;">Change price today: {{ token_changes.4 }}%</h5>
{% if token_changes.0 %}
    <h5 class="white_text">Support level on the price: {{ token_changes.0 }}$</h5>
    <h5 class="white_text" style="margin-bottom: 20px;">To support level: {{ token_changes.1 }}%</h5>
{% endif %}
{% if token_changes.2 %}
    <h5 class="white_text">Resistance level on the price: {{ token_changes.2 }}$</h5>
    <h5 class="white_text">To resistance level: {{ token_changes.3 }}%</h5>
{% endif %}
```

Рисунок 3.5.12 – умови відображення ущільнень в «graphics.html»

```
{% block content %}
<body class="bg-dark">
    <div class="container">
        <h5 class="white_text" style="...">Bag Changes</h5>
        <div style="text-align:center;">
            <img class="mb-4" src="{{ graphic }}">
        </div>
        <h5 class="white_text mb-2" style="...">Total: {{ total }}$</h5>
        <h5 class="white_text" style="...">Tokens in Binance spot bag:</h5>
        {% for token, amount in tokens_with_amounts.items %}
            <h6 class="white_text" style="...">{{ token }} = {{ amount }}$</h6>
        {% endfor %}
    </div>
</body>
{% endblock %}
```

Рисунок 3.5.13 – вигляд шаблону «bag.html»

```
{% for name, val in time_frame.items %}
<div class="col-6">
    <h5 class="white_text mt-4" style="...">Timeframe: {{ name }}</h5>
    <div class="tradingview-widget-container">
        <div id="tradingview_{{ name }}"></div>
        <script type="text/javascript">
            var container = document.getElementById("tradingview_{{ name }}");
            container.style.height = (window.innerHeight * 0.4) + "px";

            window.addEventListener('resize', function(event){
                container.style.height = (window.innerHeight * 0.5) + "px";
            });
        </script>
        <script type="text/javascript" src="https://s3.tradingview.com/tv.js"></script>
        <script type="text/javascript">
            new TradingView.widget({
                  "autosize": true,
                  "symbol": "BINANCE:{{ token }}",
                  "interval": "{{ val }}",
                  "timezone": "Etc/UTC",
                  "theme": "dark",
                  "style": "1",
                  "locale": "en",
                  "toolbar_bg": "#f1f3f6",
                  "enable_publishing": true,
                  "hide_top_toolbar": true,
                  "hide_legend": true,
                  "save_image": false,
                  "container_id": "tradingview_{{ name }}"
                });
        </script>
    </div>
</div>
{% endfor %}
```

Рисунок 3.5.14 – цикл відображення графіків в «graphics.html»

3.4 Розробка головних функцій системи

Фінальним етапом реалізації веб застосунку є збір всього розробленого в повноцінну працюючу систему, саме для цього в фреймворку Django інує файл «views.py». В цьому файлі створюється обробник запитів, приписується привязку конкретного інтернет адресу до конкретного шаблону і додається обробка та передача потрібної інформації.

Таким чином функція, що відповідає за генерацію головної сторінки, всередині себе викликає один раз на день функцію зі скриптів, що повертає список всіх доступних валютних пар на біржі.

Все записується в БД, звідки дістається ці значення, до кожної валютної пари використовується ще одна функція, яка отримує до кожної валютної пари, її вартість та відсоток змін за день, після чого повертається шаблон що пов'язаний з цією сторінкою і отримані дані, рисунок 3.6.1.

```
def base(request):
    now = timezone.localtime()
    current_time = now.time()
    target_time = datetime.time(1, 0, 0)
    if current_time < target_time or TokenName.objects.count() == 0:
        if TokenName.objects.count() != 0:
            TokenName.objects.all().delete()
        for token in TradingPairs().get_all_trading_pairs:
            TokenName.objects.create(token_name=token)

    tokens_name = TokenName.objects.all().values_list('token_name', flat=True)
    tokens_with_prices = Token().get_tokens_info(tokens_name)
    return render(request, 'main/base.html', {'tokens_with_prices': tokens_with_prices})
```

Рисунок 3.6.1 – функція що обробляє головну сторінку сайту

Далі йде функція, що поєднана зі сторінкою налаштувань користувача, вона є найбільшою серед всіх, за рахунок того, що вона в собі містить дуже багато умов і шаблонів які передаються. Так як, кожну форму потрібно перевіряти на валідацію, перевіряти значення з БД і записувати нові, а в кінці передається на сторінку великий обсяг даних, який вже потім на самій сторінці обробляється, рисунок 3.6.2.

```
content = {
    'edit_nickname_form': EditUserNicknameForm(instance=user),
    'edit_tg_account_form': EditUserTGForm(instance=user),
    'edit_password_form': EditUserPasswordForm(instance=user),
    'keys_form': APIKeysForm(instance=user),
    'nickname_error': nickname_error,
    'tg_account_error': tg_account_error,
    'password_error': password_error,
    'key_error': key_error,
    'success': success,
}
return render(request, 'main/user_settings.html', content)
```

Рисунок 3.6.2 – передача даних в шаблон сторінки редагування

Наступною йде функція, що аналізує використовуючи скрипт, задану валюту, та повертає цілий список значень різних показників, також передається словник з умовними позначеннями таймфреймів та передається на шаблон, рисунок 3.6.3.

```
def graphics(request, token):
    token_changes = Token(token).get_price_with_large_orders(token)
    content = {
        'token': token,
        'time_frame': {
            '1d': 'D',
            '1h': '60',
            '15m': '15',
            '5m': '5'
        },
        'token_changes': token_changes
    }
    return render(request, 'main/graphics.html', content)
```

Рисунок 3.6.3 – функція що обробляє сторінку з аналізом валютної пари

Функція, що обробляє сторінку з аналізом криптовалютного гаманця користувача, всередині себе обробляє багато запитів до БД та генерує за допомогою скриптів графік змін, який потім з усіма значеннями передається в шаблон, рисунок 3.6.4.

Функція, за допомогою якої користувач виходить з аккаунта, реалізована як перенаправлення на головну сторінку, і закінчення поточну сесію користувача, рисунок 3.6.5.

```
1 usage
def bag(request):
    total_db = Total()
    total_db_dates = []
    user = User.objects.filter(nickname=request.session.get('user_id')).first()
    config = UserTokensInfo().set_api_key(user.api_key).set_secret_key(user.secret_key)
    tokens_with_amounts = config.get_account_tokens_balance
    total = config.get_total_tokens_balance(tokens_with_amounts)
    graphic = config.get_total_tokens_graphic(Total.objects.filter(user=user).values('total', 'date'))

    for obj in Total.objects.filter(user=user).all():
        total_db_dates.append(obj.date.strftime('%Y-%m-%d'))

    if str(datetime.date.today()) not in total_db_dates:
        total_db.total = total
        total_db.user = user
        total_db.save()

    content = {
        'tokens_with_amounts': tokens_with_amounts,
        'total': round(total, 2),
        'graphic': graphic
    }
    return render(request, 'main/bag.html', content)
```

Рисунок 3.6.4 – функція що обробляє сторінку аналізом гаманця користувача

```
def log_out(request):
    request.session['user_id'] = None
    return redirect('home')
```

Рисунок 3.6.5 – функція що завершує сесію користувача

Функції реєстрації та входа в систему містять в собі багато перевірок на валідність даних які дістаються з БД, рисунок 3.6.6 та рисунок 3.6.7.

Крім цього, був розроблений додатковий функціонал для сповіщення користувача, завдяки телеграм-боту. Він містить в собі функціонал щоденного інформування користувача, зареєстрованого в системі, стосовно стану його крипто гаманця.

Він присилає користувачеві детальну інформацію і згенерований графік змін. Також, при першому запуску, перевіряє чи ви зареєстровані в системі, завдяки тому, що бот використовує БД web застосунка. Також бот може прислати стан гаманця, за запитом, відправляючи йому потрібну команду.

```python
def sign_up(request):
    error = None
    if request.method == 'POST':
        login_form = LoginForm(request.POST)
        if login_form.is_valid():
            email = login_form.cleaned_data['email']
            password = login_form.cleaned_data['password']
            user = User.objects.filter(email=email).first()
            if user is not None:
                if user.check_password(password):
                    login(request, user)
                    request.session['user_id'] = user.nickname
                    user.last_login = timezone.now()
                    user.save()
                    return redirect('home')
                else:
                    error = 'Invalid password'
            else:
                error = f'User with email "{email}", not registered'

    content = {
        'loginform': LoginForm(),
        'error': error
    }
    return render(request, 'main/sign-up.html', content)
```

Рисунок 3.6.4 – функція що обробляє сторінку аналізом гаманця користувача

```python
def registration(request):
    error = None
    if request.method == 'POST':
        registration_form = RegistrationForm(request.POST)
        if registration_form.is_valid():
            nickname = registration_form.cleaned_data['nickname']
            email = registration_form.cleaned_data['email']
            password = registration_form.cleaned_data['password']
            repeat_password = registration_form.cleaned_data['repeat_password']
            if not User.objects.filter(nickname=nickname).first():
                if not User.objects.filter(email=email).first():
                    if password == repeat_password:
                        registration_form.save()
                        user = User.objects.filter(email=email).first()
                        login(request, user)
                        request.session['user_id'] = user.nickname
                        user.last_login = timezone.now()
                        user.save()
                        return redirect('home')
                    else:
                        error = 'Passwords do not match'
                else:
                    error = f'User with email "{email}", is registered'
            else:
                error = f'Nickname {nickname} if busy'

    content = {
        'regform': RegistrationForm(),
        'error': error
    }
    return render(request, 'main/registration.html', content)
```

Рисунок 3.6.4 – функція що обробляє сторінку аналізом гаманця користувача

3.5 Розробка коду для ініціалізації, навчання та оптимізації нейронної мережі

Для розробки коду ініціалізації, навчання та оптимізації глибокої нейронної мережі використовується бібліотека TensorFlow у мові Python. Починаючи з ініціалізації моделі, використовується архітектура ResNet50. Ваги моделі можуть бути завантажені з попередньо навченими вагами або використані випадкові ваги. Після компіляції моделі з оптимізатором Adam та функцією втрати SparseCategoricalCrossentropy, модель навчається на тренувальних даних протягом заданої кількості епох. Після навчання, модель оцінюється на валідаційному наборі даних для оцінки її точності.

Локалізація ImageNet

Завдання ImageNet Localization (LOC) вимагає класифікації та локалізації об’єктів. Класифікатори рівня зображення спочатку застосовуються для передбачення міток класу зображення, а алгоритм локалізації враховує лише передбачення обмежувальних рамок на основі передбачених класів. Ми приймаємо стратегію

«регресії за класом» (PCR), вивчаючи регресор обмежувальної рамки для кожного класу. Ми попередньо навчаємо мережі для класифікації ImageNet, а потім налаштовуємо їх для локалізації. Ми навчаємо мережі на наданому навчальному наборі ImageNet із 1000 класів.

Алгоритм локалізації базується на структурі RPN з кількома модифікаціями. На відміну від способу який не залежить від категорії, наш RPN для локалізації розроблений у формі для кожного класу. Цей RPN закінчується двома однорідними згортковими шарами 1×1 для бінарної класифікації (cls) і коробкової регресії (reg). Рівні cls і reg є окремими для кожного класу.

Зокрема, рівень cls має результат 1000 d, і кожен вимір є двійковою логістичною регресією для прогнозування того, бути чи не бути класом об’єктів; рівень reg має вихід 1000 × 4-d, що складається з коробкових регресорів для 1000 класів. Регресія обмежувальної рамки стосується кількох трансляційно-інваріантних «якірних» рамок у кожній позиції.

Як і під час навчання класифікації ImageNet, ми випадково відбираємо 224 × 224 кадри для збільшення даних та використовуємо міні-пакет із 256 зображень для точного налаштування. Щоб уникнути домінування негативних зразків, для кожного зображення випадковим чином відбирають 8 опорних точок, при цьому відібрані позитивні та негативні опорні значення мають співвідношення 1:1. Для тестування мережа наноситься на зображення повністю згортково.

Спочатку виконуємо тестування «оракула», використовуючи базовий клас істинності як класифікаційний прогноз. Папір VGG звітує про помилку кадрування центру 33,1% з використанням класів істинності землі. За тих самих налаштувань наш метод RPN із використанням мережі ResNet-101 значно зменшує похибку кадрування центру до 13,3%. Це порівняння демонструє чудову продуктивність нашого фреймворку. Зі щільним (повністю згортковим) і багатомасштабним тестуванням наш ResNet-101 має похибку 11,7% з використанням наземних класів істинності. Використовуючи ResNet-101 для прогнозування класів (помилка класифікації топ-5 4,6%, таблиця 4), помилка локалізації топ-5 становить 14,4%.

Проблема оптимізації навряд чи спричинена зникаючими градієнтами. Ці звичайні мережі навчені за допомогою BN, що гарантує, що сигнали, поширюються вперед, мають ненульову дисперсію. 34-шарова проста мережа все ще здатна досягти конкурентоспроможної точності, що свідчить про те, що розв’язувач певною мірою працює. Глибокі чисті мережі можуть мати експоненціально низькі швидкості конвергенції, які впливають на залишкові мережі.

Залишкові мережі не мають додаткових параметрів порівняно з

зменшення похибки навчання. Причина таких труднощів оптимізації буде досліджена в майбутньому.

Залишкові мережі. У першому порівнянні ми використовуємо відображення ідентичності для всіх ярликів і нульове доповнення. Тому вони не мають додаткових параметрів в порівнянні з простими аналогами.

Ми маємо три основні спостереження. По-перше, ситуація зворотна із залишковим навчанням.

У першому порівнянні ми використовуємо відображення ідентичності для всіх ярликів і нульове доповнення. Тому вони не мають додаткових параметрів в порівнянні з простими аналогами.

Ми маємо три основні спостереження. По-перше, ситуація зворотна із залишковим навчанням.

По-друге, порівняно з його простим аналогом, ResNet зменшує першу помилку на 3,5%, що є результатом успішно зменшеної помилки навчання. Це порівняння підтверджує ефективність залишкового навчання на надзвичайно глибоких системах.

Далі ми досліджуємо ярлики проекції.

Наведені вище результати базуються лише на мережі пропозицій (RPN) у Faster R-CNN. Для покращення результатів можна використовувати мережу виявлення (Fast R-CNN) у Faster R-CNN. Але ми помітили, що в цьому наборі даних одне зображення зазвичай містить один домінуючий об’єкт, а регіони пропозицій сильно накладаються одна на одну і, таким чином, мають дуже схожі об’єднані функції RoI. Як результат, орієнтоване на зображення навчання Fast R-CNN генерує зразки невеликих варіацій, які можуть бути небажаними для стохастичного навчання. Мотивуючись цим, у нашому поточному експерименті ми використовуємо оригінальний R-CNN, який орієнтований на RoI, замість Fast R-CNN.

Наша реалізація R-CNN така. Ми застосовуємо RPN для кожного класу, навчений, як описано вище, на навчальних зображеннях для прогнозування обмежувальних рамок для наземного класу істинності. Ці передбачені коробки відіграють роль залежних від класу пропозицій.

Для кожного навчального зображення 200 пропозицій з найвищим балом витягуються як навчальні зразки для навчання класифікатора R-CNN. Область зображення обрізається з пропозиції, деформується до 224 × 224 пікселів і подається в мережу класифікації, як у R-CNN. Виходи цієї мережі складаються з двох однорідних рівнів fc для cls і reg, також у формі для кожного класу. Ця мережа R-CNN налаштована на тренувальному наборі з використанням розміру міні-пакета 256 відповідно до ROI.

Для кожного навчального зображення 200 пропозицій з найвищим балом витягуються як навчальні зразки для навчання класифікатора R-CNN. Область зображення обрізається з пропозиції, деформується до 224 × 224 пікселів і подається в мережу класифікації, як у R-CNN. Виходи цієї мережі складаються з двох однорідних рівнів fc для cls і reg, також у формі для кожного класу. Ця мережа R-CNN налаштована на тренувальному наборі з використанням розміру міні-пакета 256 відповідно до ROI. Для тестування RPN генерує 200 пропозицій з найвищим балом для кожного передбаченого класу, а мережа R-CNN використовується для оновлення балів і позицій цих пропозицій.

Цей метод знижує п’ять основних помилок локалізації до 10,6%. Це результат нашої одномодельної перевірки. Використовуючи сукупність мереж як для класифікації, так і для локалізації, ми досягли п’ятірки найпоширеніших помилок локалізації 9,0% у тестовому наборі. Це число значно перевершує результати ILSVRC 14, демонструючи відносне зменшення помилки на 64%. Цей результат отримав 1-ше місце в завданні локалізації ImageNet в ILSVRC 2022 році.

3.6 Тестування та оцінка ефективності розробленого програмного забезпечення

Після навчання та оптимізації моделі, проводиться тестування розробленого програмного забезпечення. Для цього використовується тестовий набір даних, що містить зображення з об'єктами їжі. Результати тестування аналізуються, включаючи метрики точності та швидкодії. У разі виявлення можливих обмежень або покращень моделі, розробляються рекомендації для подальшого вдосконалення програмного забезпечення, наприклад, зміна архітектури моделі, оптимізація параметрів навчання або використання інших алгоритмів обробки зображень.

Тестування та валідація програмного забезпечення є важливим етапом в розробці будь-якої системи, включаючи систему рекомендацій.

Тестування програмного забезпечення має на меті виявлення помилок та недоліків у програмному коді та його функціональності. Це можна зробити шляхом ручного тестування, автоматичного тестування або їх комбінації.

Ручне тестування включає в себе тестування функціональності, тестування користувальницького інтерфейсу, тестування відповідності вимогам, тестування безпеки та тестування продуктивності. Автоматичне тестування може бути використане для автоматизації повторюваних тестів та тестування функцій, які важко або неможливо виконати вручну.

Валідація програмного забезпечення включає в себе перевірку правильності реалізації функціональності та відповідності вимогам, а також перевірку на безпеку та продуктивність. Це може бути зроблено шляхом проведення тестів на зразках даних, аналізу даних вводу та виводу, а також оцінювання ефективності програми в залежності від обсягу та складності даних.

У процесі тестування та валідації програмного забезпечення необхідно забезпечити максимально можливу покриття тестами та переконатися, що програмний код працює правильно, безпечно та ефективно з усіма можливими вхідними даними. Це допоможе підвищити якість та надійність програмного забезпечення та зменшити ризик непередбачених помилок та недоліків, що можуть виникнути у процесі роботи програм

Існує кілька методів тестування програмного забезпечення, які можуть бути використані для валідації розробленого програмного забезпечення. Нижче описані деякі з них:

1. Модульне тестування: цей метод полягає у тестуванні окремих модулів програмного забезпечення. Модулі можуть бути протестовані окремо від інших модулів з використанням заміни вхідних даних тестовими даними. Модульне тестування дозволяє виявляти помилки у маленьких модулях програми і виправляти їх раніше, ніж вони стануть проблемою в більшому контексті.
2. Інтеграційне тестування: цей метод полягає у тестуванні взаємодії між різними модулями програмного забезпечення. Тестування проводиться на рівні інтерфейсів між модулями, де можуть виникнути помилки при обміні даними.
3. Системне тестування: цей метод включає тестування всієї системи, яка складається з різних модулів, включаючи інтерфейси користувача та інші компоненти.

Системне тестування виконується для перевірки того, чи працює програма з точки зору відповідності вимогам та очікуванням користувачів.

4. Функціональне тестування: цей метод полягає у тестуванні функціональності програмного забезпечення відповідно до вимог, описаних у специфікації вимог.

5. Навантажувальне тестування: цей метод полягає у тестуванні програмного забезпечення під різними навантаженнями та обсягами даних для визначення його меж витривалості та продуктивності.

6. Тестування безпеки: цей метод включає тестування програмного забезпечення

ВИСНОВКИ

1. Проведено аналіз задач розпізнування зображень.
2. Визначено переваги та можливості сучасних програмних засобів.
3. Досліджено моделі ResNet та виявлено ефективніший з них для вирішення окремих задач розпізнавання зображень.
4. Проведено тестування та оцінка ефективності розробленого програмного забезпечення.

ПЕРЕЛІК ПОСИЛАНЬ

1. M Sivarajah, U., Kamal, M. M., Irani, Z., & Weerakkody, V. (2017). Critical analysis of Big Data challenges and analytical methods. Journal of Business Research, 70, 263-286.
URL: https://doi.org/10.1016/j.jbusres.2016.08.001

2. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute.
URL: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/big-data-the-next-frontier-for-innovation

3. Chen, H., Chiang, R. H., & Storey, V. C. (2012). Business intelligence and analytics: From Big Data to Big Impact. MIS Quarterly, 36(4), 1165-1188.
URL: https://www.jstor.org/stable/41703495

4. McAfee, A., & Brynjolfsson, E. (2012). Big data: The management revolution. Harvard Business Review, 90(10), 60-68.
URL: https://hbr.org/2012/10/big-data-the-management-revolution

5. An INTRANET-Based Web Application for College Management System Using Python with Django Web Framework [Електронний ресурс] / [N. Reddy, D. Tej, D. Koncha та ін.]. - 2023. - Режим доступу до ресурсу: https://www.researchgate.net/publication/368719592_An_INTRANET-Based_Web_Application_for_College_Management_System_Using_Python_with_Django_Web_Framework.

6. Посібник по Django для початківців [Електронний ресурс]. - 2017. - Режим доступу до ресурсу: https://codeguida.com/post/1039.

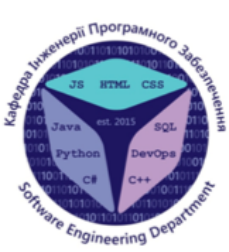
7. Розробка веб додатків з використанням Python і Django [Електронний ресурс]. - 2021. - Режим доступу до ресурсу: https://webcase.com.ua/uk/blog/razrabotka-veb-prilozhenij-s-ispolzovaniem-python-i-django/.

ДОДАТКИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



Розробка програмного забезпечення для вирішення окремих задач розпізнавання зображень на основі глибоких нейронних мереж мовою Python

Виконав студент 5 курсу

групи ППЗ-51
Шеголь Анатолій Глібович

Керівник роботи
Треньов Микита Георгійович

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** - покращення процесу розпізнавання об'єктів на зображеннях за допомогою глибоких нейронних мереж реалізованих мовою Python.
- **Об’єкт дослідження** - процес розпізнавання об'єктів на зображеннях за допомогою глибоких нейронних мереж.
- **Предмет дослідження** - програмне забезпечення для розпізнавання об'єктів на зображеннях за допомогою глибоких нейронних мереж.

ЗАДАЧІ БАКАЛАВРСЬКОЇ РОБОТИ

1. **Огляд літератури**
   Вивчення існуючих методів та алгоритмів розпізнавання об'єктів на зображеннях, особливостей глибоких нейронних мереж та їх використання в розпізнаванні об'єктів.

2. **Збір та підготовка даних**
   Зібрати відповідний набір даних для навчання та тестування моделі розпізнавання. Провести попередню обробку та підготовку даних для використання.

3. **Розробка та налаштування моделі**
   Розробити архітектуру глибокої нейронної мережі, виконати процес навчання моделі з використанням підготовлених даних. Налаштувати гіперпараметри моделі.

4. **Оцінка ефективності**
   Здійснити оцінку ефективності розробленої моделі на тестовому наборі даних. Виміряти показники точності, часу виконання та інші метрики.

5. **Вдосконалення результатів**
   Провести аналіз результатів та виявлення проблем. Запропонувати можливі покращення, які можуть бути виконані для підвищення ефективності розпізнавання об'єктів

АНАЛІЗ АНАЛОГІВ

Показники	Google Cloud Vision API	Microsoft Azure Computer Vision	Amazon Rekognition	Clarifai
Розпізнавання об'єктів	+	+	+	+
Класифікація зображень	+	+	+	+
Розпізнавання облич	+	+	+	+
Розпізнавання рукописного тексту	-	-	-	+
Аналіз вмісту зображень	+	+	+	+
Виявлення міток	+	+	+	+

АНАЛІЗ АНАЛОГІВ

Показники	Google Cloud Vision API	Microsoft Azure Computer Vision	Amazon Rekognition	Clarifai
Виявлення відомих облич	+	+	+	-
Виявлення небажаного вмісту	+	+	+	+
Розпізнавання тексту	+	+	+	+
Аналіз настрою	+	+	+	+

5

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1. **Наявність можливості навчання власної моделі:**
   Можливість користувачеві тренувати власну модель для розпізнавання об'єктів на зображеннях, що дозволить налаштувати розпізнавання під конкретні потреби.

2. **Розпізнавання об'єктів на зображеннях:**
   Програмне забезпечення має включати функціональність для точного та швидкого розпізнавання об'єктів на зображеннях.

3. **Інтерфейс користувача:**
   Програмне забезпечення повинно мати інтуїтивно зрозумілий інтерфейс користувача, побудований на Tkinter, що дозволить зручно взаємодіяти з програмою та налаштовувати параметри розпізнавання об'єктів.

4. **Збереження результатів:**
   Програма має мати можливість зберігати результати розпізнавання об'єктів для подальшого використання або аналізу.

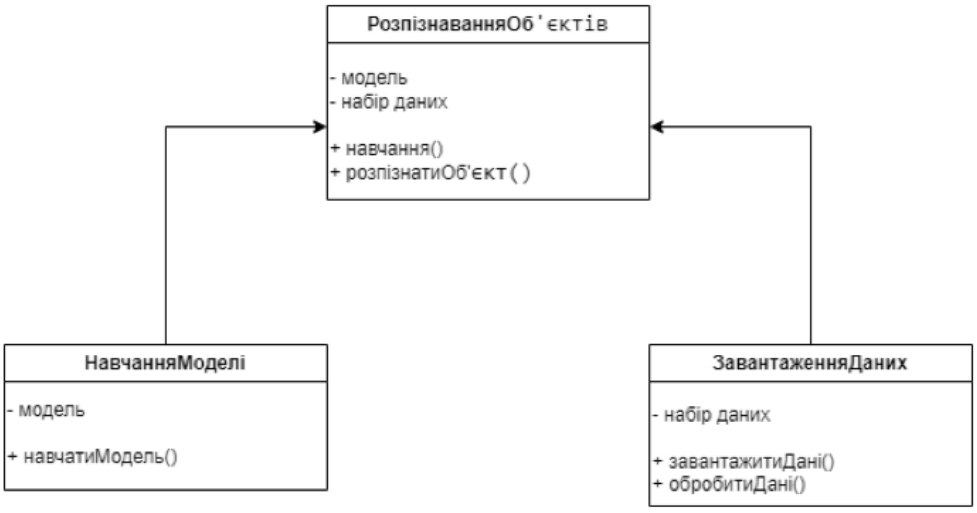
5. **Оцінка результатів розпізнавання користувачем:**
   Описує необхідність надання користувачу можливості оцінювати результати розпізнавання об'єктів на зображеннях, вказуючи, чи вірно вони були розпізнані чи ні.

6

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

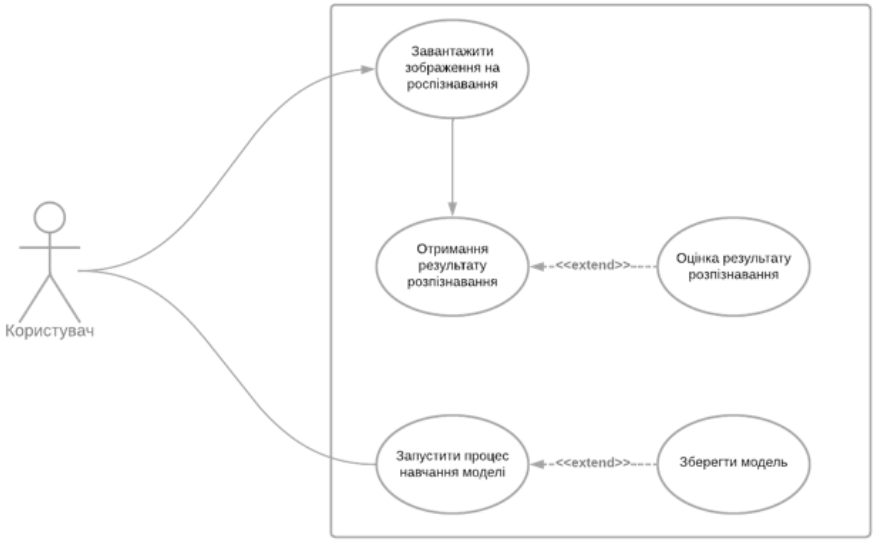
7

ДІАГРАМА КЛАССІВ



8

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



9

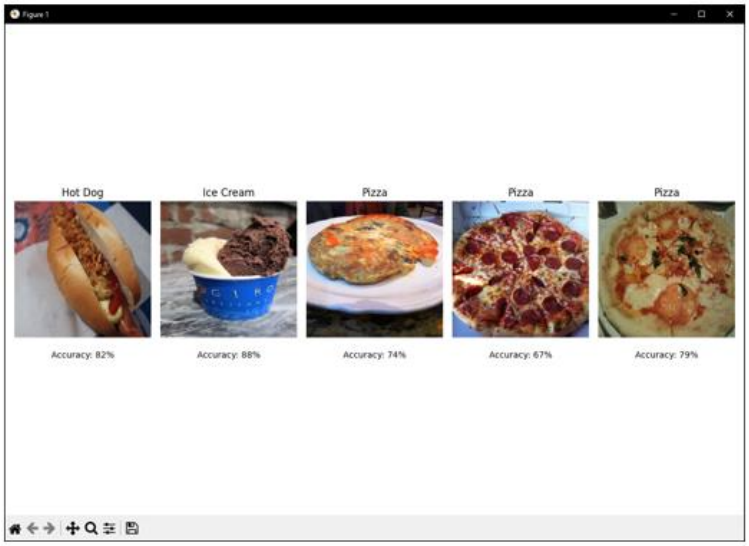
ЕКРАННІ ФОРМИ

```
Epoch 1/10
296/296 [==============================] - 290s 948ms/step - loss: 4.6483 - accuracy: 0.0125 - val_loss: 4.6217 - val_accuracy: 0.0149
Epoch 2/10
296/296 [==============================] - 283s 951ms/step - loss: 4.6099 - accuracy: 0.0160 - val_loss: 4.5923 - val_accuracy: 0.0171
Epoch 3/10
296/296 [==============================] - 284s 954ms/step - loss: 4.5862 - accuracy: 0.0186 - val_loss: 4.5707 - val_accuracy: 0.0199
Epoch 4/10
296/296 [==============================] - 292s 980ms/step - loss: 4.5679 - accuracy: 0.0210 - val_loss: 4.5533 - val_accuracy: 0.0240
Epoch 5/10
296/296 [==============================] - 288s 967ms/step - loss: 4.5528 - accuracy: 0.0234 - val_loss: 4.5387 - val_accuracy: 0.0287
Epoch 6/10
296/296 [==============================] - 286s 961ms/step - loss: 4.5397 - accuracy: 0.0254 - val_loss: 4.5258 - val_accuracy: 0.0322
Epoch 7/10
220/296 [=====================>........] - ETA: 54s - loss: 4.5296 - accuracy: 0.0271
```

Процес навчання моделі на наборі даних “food101” в блокноті Jupyter

10

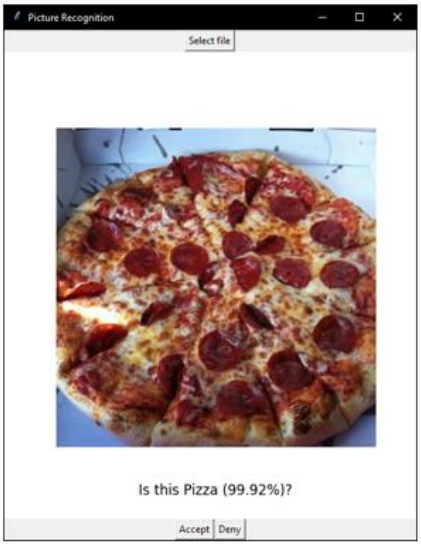
ЕКРАННІ ФОРМИ



Форма спроби розпізнати об’єкти на вибірці зображень

11

ЕКРАННІ ФОРМИ



Форма завантаження, розпізнавання та оцінки передбачення

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Треньов М.Г., Щеголь А.Г. Роль Big Data та аналізу даних у розвитку інформаційних технологій // Всеукраїнська науково-технічна конференція "Застосування програмного забезпечення в інфокомунікаційних технологіях" - Київ: ДУТ, 2023.

2. Треньов М.Г., Щеголь А.Г. Розвиток технологій Інтернету речей в поєднанні з штучним інтелектом // Всеукраїнська науково-технічна конференція "Застосування програмного забезпечення в інфокомунікаційних технологіях" - Київ: ДУТ, 2023.

ВИСНОВКИ

1. Проведено аналіз задач розпізнування зображень.

2. Визначено переваги та можливості сучасних програмних засобів.

3. Досліджено моделі ResNet та виявлено ефективніший з них для вирішення окремих задач розпізнавання зображень.

4. Проведено тестування та оцінка ефективності розробленого програмного забезпечення.