НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Приладобудівний факультет

Кафедра комп'ютерно-інтегрованих оптичних та навігаційних систем

До захисту допущено:

Завідувач кафедри

_____ Надія БУРАУ

«___»_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерно - інтегровані технології та системи навігації і керування»

спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»

на тему: « ШТУЧНИЙ ІНТЕЛЕКТ В СИСТЕМАХ РОЗПІЗНАВАННЯ СИМВОЛІВ »

Виконав :
студент IV курсу, групи ПГ-91
Казьмірук Костянтин Олександрович _____

Керівник:
Доцент, к.т.н., доцент
Мироненко Павло Степанович _____

Рецензент:
Доцент, кандидат технічних наук
Шевченко Вадим Володимирович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Приладобудівний факультет

Кафедра комп'ютерно-інтегрованих оптичних та навігаційних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма – «Комп'ютерно-інтегровані технології та системи навігації і керування»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Надія Бурау

«___»_____2023 р.

ЗАВДАННЯ

на дипломну роботу студенту

Казьміруку Костянтину Олександровичу

1. Тема роботи « **ШТУЧНИЙ ІНТЕЛЕКТ В СИСТЕМАХ РОЗПІЗНАВАННЯ СИМВОЛІВ** », керівник роботи Мироненко Павло Степанович, канд. тех. наук, доцент, затверджений наказом по університету від «___»_____ 20__ р. №_____

2. **Термін подання** студентом роботи: 06.06.2023 р.

3. **Зміст роботи**:

1. Інформаційно-аналітичний огляд сучасного використання штучного інтелекту у різних галузях.

2. Створення веб-інтерфейсу для програми зчитування даних з чеків.

3. Створення BackEnd API для виведення даних у веб-інтерфейс.

4.Розроблення схеми використання веб-інтерфейсу - підключення до API зчитування даних, які використовує модель штучного інтелекту.

5. Оформити проект та задеплоїти на сервер.

6. Дата видачі завдання: 07.03.2023

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Інформаційно-аналітичний огляд сучасного використання штучного інтелекту у різних галузях	10.05.2023	Вик
2.	Створення веб-інтерфейс для програми зчитування даних з чеків.	01.04.2023	Вик
3.	Створити BackEnd API для виведення даних у веб-інтерфейс.	01.05.2023	Вик
4.	Розроблення схеми використання веб-інтерфейсу. Підключення до API зчитування даних, які використовує модель штучного інтелекту.	25.05.2023	Вик
5.	Оформлення проекту та задеплоїти на сервер.	07.06.2023	Вик

Студент Костянтин Казьмірук

Керівник роботи Павло Мироненко

АНОТАЦІЯ

Дипломна робота представлена на 62 аркушах, використані джерела з 15 найменувань; у роботі наведено 25 рисунки.

Дана дипломна робота спрямована на дослідження штучного інтелекту, його використання у різних галузях господарства , огляду відстежування об’єктів та зчитування даних з документів.

У роботі було проведено огляд сучасного стану використання штучного інтелекту на прикладах різних провідних компаній світу, що працюють в технологічних галузях, наведені приклади сучасного використання штучного інтелекту, напрямки подальшого його розвитку та приклад у використанні.

ABSTRACT

The diploma thesis consists of 62 pages and references 15 sources. It includes 25 figures.

This diploma thesis aims to explore artificial intelligence and its applications in various sectors of the economy, particularly object tracking and data extraction from documents.

The thesis provides an overview of the current state of artificial intelligence usage, focusing on case studies from leading global companies in the technology industry. It presents examples of contemporary applications of AI, discusses future directions for its development, and highlights implementation.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ШІ – Штучний Інтелект

LLM (Large Language Model) – велика мовна модель

ChatGPT – Чат-бот на основі LLM

DALL-E AI – Штучний інтелект, який створює картинку за описом

ReactJS – JavaScript бібліотека

NextJS – JavaScript фреймворк на основі ReactJS

TailwindCSS – CSS фреймворк

AWS – Амазон веб сервіси

API (Application Programming Interface) – це набір правил і протоколів, які дозволяють різним програмним компонентам взаємодіяти один з одним. API визначає, які функції та операції можуть бути використані для комунікації між різними програмними компонентами, такими як програми, бібліотеки або сервіси.

SQL (Structured Query Language) – це стандартна мова програмування для керування та роботи з реляційними базами даних. SQL дозволяє створювати, модифікувати та керувати базами даних, виконувати операції зберігання, оновлення, вибірки та видалення даних.

ВСТУП

Тема штучного інтелекту набуває все більшої актуальності в сучасному світі. ШІ впливає на різні сфери життя, включаючи технології, медицину, транспорт, фінанси, освіту та багато інших.

Однією з основних переваг штучного інтелекту є здатність обробляти великі обсяги даних і виділяти у них корисну інформацію. ШІ може виявити складні залежності та тренди, що дозволяє зробити більш точні прогнози та приймати обґрунтовані рішення. Це особливо корисно в галузі біомедицини, де ШІ може допомогти у виявленні нових ліків, діагностиці захворювань та управлінні даними пацієнтів.

Штучний інтелект є галуззю комп'ютерних наук, що зосереджується на створенні імітації інтелекту та когнітивних функцій у комп'ютерних системах. Він включає в себе розробку алгоритмів та моделей, які дозволяють комп'ютерам аналізувати, розуміти та виконувати складні завдання, які зазвичай пов'язані з людським інтелектом. Штучний інтелект знаходить своє застосування в багатьох галузях, включаючи:

1. Медицину: ШІ може бути використаний для діагностики і прогнозування хвороби, аналізу медичних зображень та допомозі в розробці нових лікарських препаратів;
2. Автономні транспортні засоби: ШІ використовується для розробки систем управління транспортними засобами в режимах управління автопілотами автомобілями, дронами та роботами;
3. Фінанси: ШІ використовується для прогнозування ринків, автоматизації фінансових операцій та розробки інтелектуальних фінансових портфелів;
4. Електронна комерція: ШІ допомагає у персоналізації пропозицій, рекомендації покупцям, обробці великих обсягів даних та автоматизації процесів продажу;
5. Розваги: ШІ використовується в іграх, віртуальній реальності, розпізнаванні голосу та образів, що дозволяє створювати більш реалістичний досвід для користувачів.

Майбутні програми навчання систем з ШІ повинні зосередитися на тому, як працювати з штучним інтелектом в різних умовах експлуатації, бути більш спеціалізованими та настроєними на конкретний контекст.

РОЗДІЛ 1. ОГЛЯД ПРО ШТУЧНИЙ ІНТЕЛЕКТ. ЙОГО ПЕРЕВАГИ ТА НЕДОЛІКИ

Штучний інтелект допомагає автоматизувати та оптимізувати рутинні задачі, звільняючи людей від монотонної роботи і відкриваючи простір для творчості та інновацій. Наприклад, автоматизовані системи виробництва можуть покращити ефективність та якість продукції, а автономні автомобілі можуть забезпечити безпеку та зручність у транспорті.

Один із очевидних прикладів використання ШІ - це наші смартфони. Майже кожен сучасний мобільний пристрій має в собі предикативну систему набору тексту, як T9 або сучасніші варіанти, які допомагають нам швидше та ефективніше вводити повідомлення, електронну пошту або робити пошукові запити. Це лише одна з безлічі ситуацій, коли ми використовуємо ШІ, не навіть задумуючись про це.

1.1 Від математики до штучного інтелекту

Знання математики та алгоритмів відіграють ключову роль у розвитку та розширенні штучного інтелекту . ШІ по суті є комп'ютерною програмою або апаратним забезпеченням, яке намагається моделювати та імітувати розум та поведінку людини.

ШІ базується на математичних концепціях та алгоритмах для розв'язання складних завдань. Наприклад, нейронні мережі, одна з ключових технологій у сфері ШІ, використовують математичні моделі, такі як лінійна алгебра та теорія ймовірностей, для моделювання роботи людського мозку. Алгоритми машинного навчання, зокрема методи навчання з учителем та без

учителя, використовуються для тренування моделей ШІ на великих обсягах даних та розв'язання завдань класифікації, розпізнавання образів, прогнозування тощо.

Крім того, алгоритми ШІ також використовуються для оптимізації рішень та прийняття розумних вирішень в реальному часі. Наприклад, генетичні алгоритми можуть бути використані для пошуку оптимальних рішень в складних проблемах, таких як планування маршрутів або оптимізація виробничих процесів

Знання математики та складних алгоритмів допомагає розробникам ШІ створювати моделі та алгоритми, які здатні аналізувати, інтерпретувати та робити швидкі та якісні висновки з великих обсягів даних. Це дає можливість ШІ "навчитися" на основі даних та здійснювати передбачення, класифікацію та прийняття рішень.

Компанія Deep Blue, була першою, яка впровадила математику та алгоритми, що змогли перемогти чемпіона з гри в шахи, це є прикладом значного прориву в галузі штучного інтелекту і комп'ютерної науки. Deep Blue був розроблений командою інженерів та науковців з IBM, зокрема Гаррі Каспаровим, в навчальних матчах у 1996 році та отримав перемогу над чемпіоном світу в першому матчі 1997 року.

Основою алгоритму Deep Blue представляли розширені версії методу мінімаксного дерева та альфа-бета відсічення, що є класичною технікою в галузі шахового програмування. Алгоритм складався з декількох компонентів.

Генерацію можливих ходів. Deep Blue аналізував шахову дошку та генерував всі можливі ходи, доступні для кожного стану гри.

Оцінка позиції. Для кожного стану гри Deep Blue використовував функцію оцінки, яка призначала числову оцінку поточної позиції на дошці. Ця оцінка враховувала різні фактори, такі як позиція фігур, контроль над центральними пунктами, кількість фігур на дошці тощо.

Пошук з використанням методу мінімаксного дерева. Deep Blue використовував метод мінімаксу для пошуку найкращого ходу. Він

рекурсивно аналізував можливі ходи та відповіді на них, шукаючи найоптимальнішу стратегію для кожного гравця.

Альфа-бета відсічення. Ця техніка дозволяла скоротити кількість ходів, обмежуючи дослідження гілок, які мають меншу важливість. Вона базується на використанні верхньої (альфа) та нижньої (бета) меж, які визначаються під час пошуку та допомагають виключити невигідні ходи.

Однак, успіх Deep Blue не обмежується лише алгоритмами. Ключовою складовою його успіху була велика обчислювальна потужність, яку надавали спеціально побудовані обчислювальні системи. Deep Blue був оснащений великою кількістю процесорів, що дозволяло йому швидко розраховувати та аналізувати великі кількості можливих ходів.

Алгоритм Deep Blue використовував розширені версії методів мінімаксного дерева та альфа-бета відсічення, сполучені з потужними обчислювальними ресурсами. Ця комбінація дозволила системі аналізувати широкий спектр можливих ходів та приймати оптимальні рішення на кожному кроці гри.

З іншого боку, нейронні мережі та методи глибинного навчання, такі як згорткові нейронні мережі (Convolutional Neural Networks), стали революційними в сфері комп'ютерного бачення і гри в шахи. Вони змогли значно спростити процес прогнозування шахових позицій та знаходження оптимальних ходів.

Завдяки нейронним мережам, шахові алгоритми можуть вивчати та впроваджувати складні зв'язки і шаблони, які не легко врахувати вручну. Вони можуть ефективно оцінювати стан шахової дошки та прогнозувати ймовірні наступні ходи, опираючись на великий обсяг тренувальних даних та вдосконалений алгоритм навчання.

Під штучною нейронною мережею розуміється математична модель, а також її програмна та апаратна реалізація, побудована за принципу біологічних нейронних мереж - нервових клітин живого організму. Це поняття виникли при спробі змоделювати процеси, що протікають у мозку людини.

Штучна нейронна мережа являє собою систему простих процесорів (штучних нейронів), з'єднаних і взаємодіючих між собою. Кожен з процесорів мережі має справу з сигналами, які періодично надходять або передаються іншим процесорам. Велика мережа здатна вирішувати найскладніші завдання в найкоротші терміни.

З математичної точки зору нейронні мережі являють собою спосіб вирішення нелінійних завдань оптимізації. Кібернетика використовує теорію нейронних мереж у вирішенні завдань адаптивного управління, побудові алгоритмів для робототехніки [1].

1.2 Відкрийте світ нейронних мереж: Покрокове навчання та застосування

Алгоритм корекції помилки є стандартним методом навчання. У цьому методі ваги змінюються тільки тоді, коли реакція є неправильною. У такому випадку вага змінюється на протилежне значення помилки.

Ми розглянемо алгоритм зворотного розповсюдження помилки на прикладі нейронної мережі, що складається з трьох шарів. У даному прикладі мережа має два входи та один вихід (див. Рисунок 1.1).

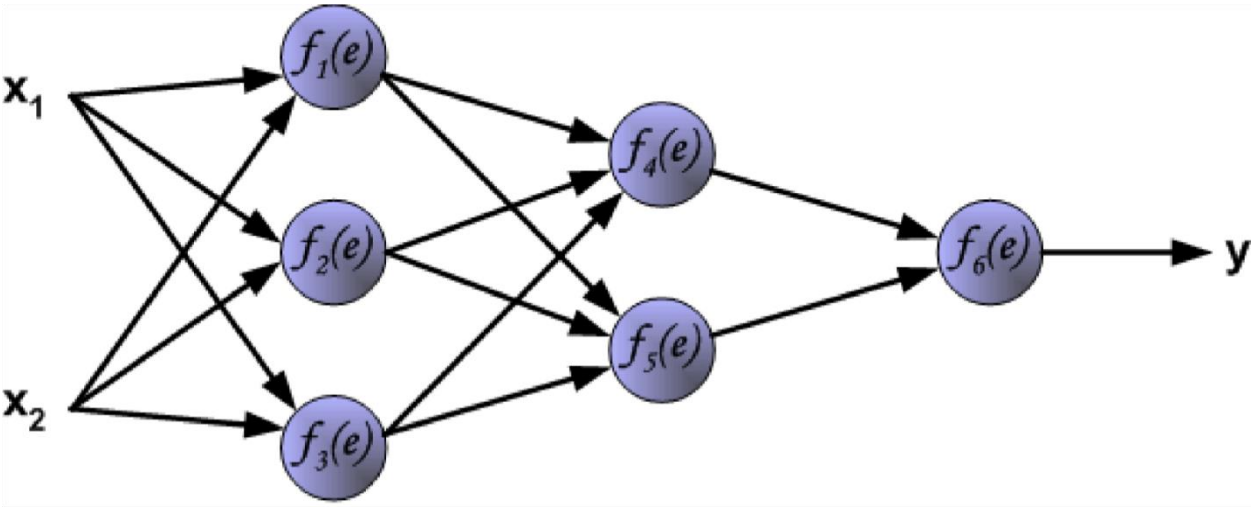

Рисунок 1.1 Нейронна мережа з трьох шарів

Для навчання нейронної мережі необхідно мати дані для тренування. У нашому випадку, ці дані включають сигнали □', □ та очікуваний результат □. Під час кожної ітерації, вагові коефіцієнти адаптуються з використанням нових тренувальних даних. Основна ідея алгоритму полягає у зміні вагових коефіцієнтів.

Застосовуючи формули для обчислення вихідних сигналів та функції активації, ми можемо отримати значення вихідного нейрона (див. Рисунок 1.2) [2].

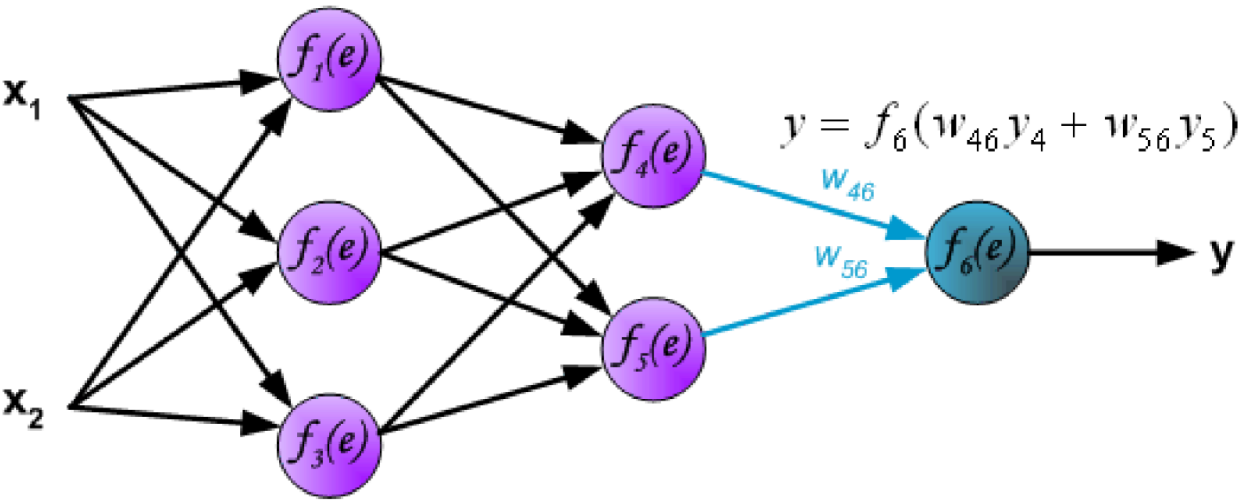

Рисунок 1.2 Обчислення вихідних значень нейрона

У наступному кроці алгоритму, результат (y) порівнюється з очікуваним

значенням (z). Різниця між цими значеннями (□) відома як помилка вихідного шару (див. Рисунок 1.3) [2].

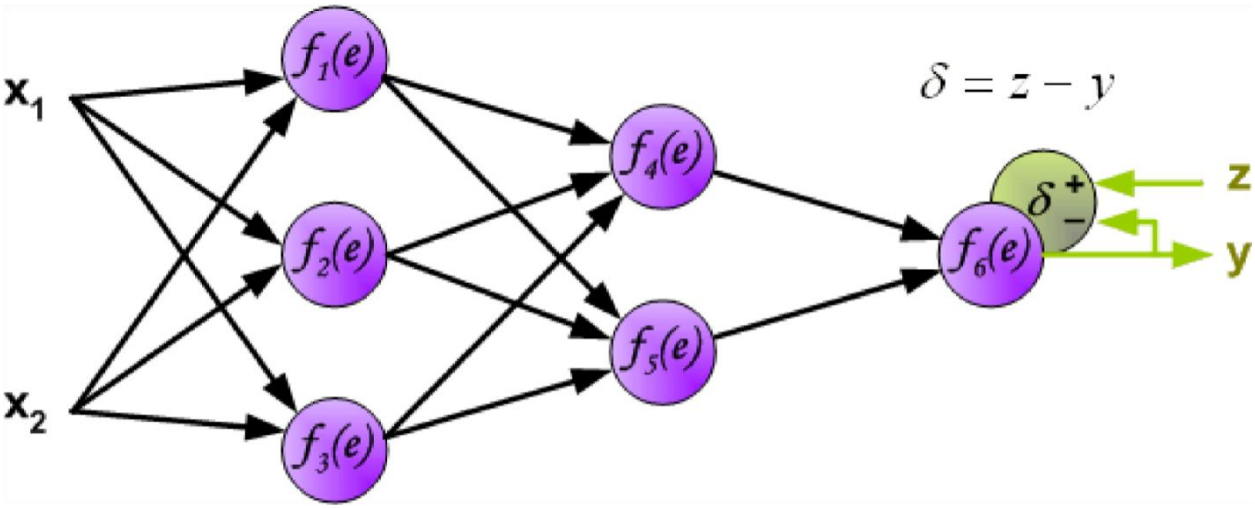

Рисунок 1.3 Помилка вихідного шару

Цей алгоритм ґрунтується на ідей про розповсюдження помилкового сигналу (□) у зворотному напрямку до всіх нейронів, чиї сигнали були вхідними (див. Рисунок 1.4 та Рисунок 1.5) [2].

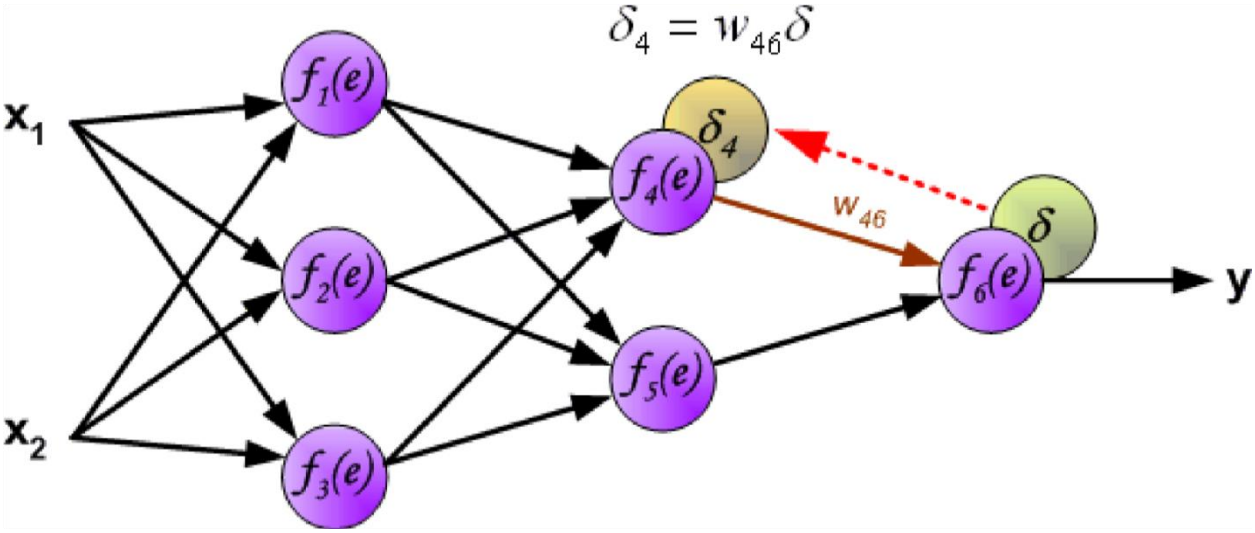

Рисунок 1.4 Помилковий сигнал

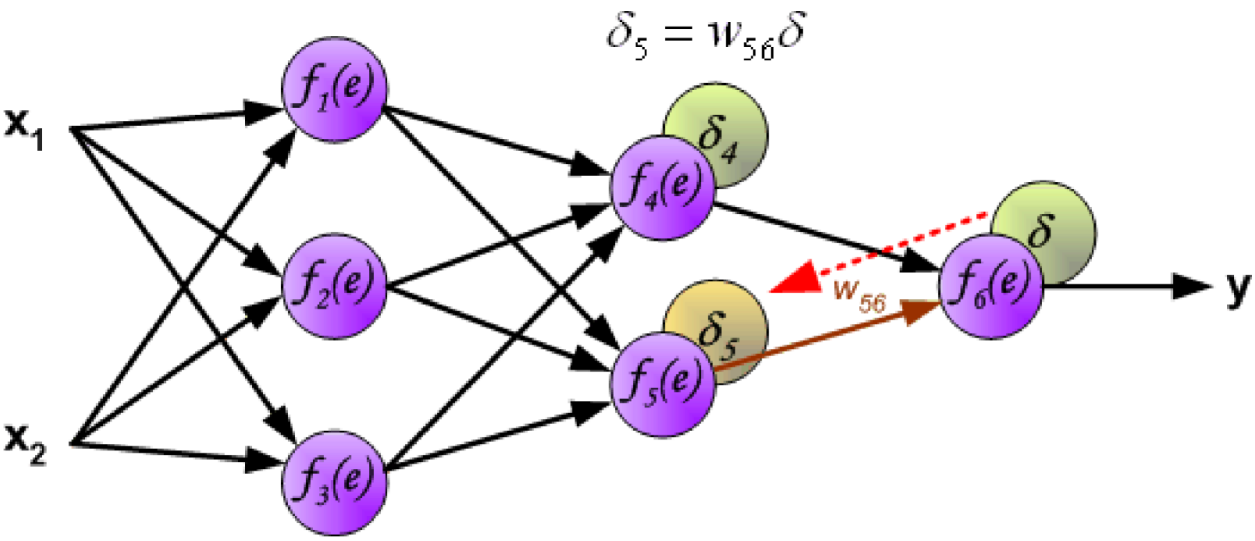

Рисунок 1.5 Помилковий сигнал 2

Вагові коефіцієнти залишаються незмінними. Якщо нейрон був вхідним для декількох нейронів, значення сумуються (див. Рисунок 1.6) [2].

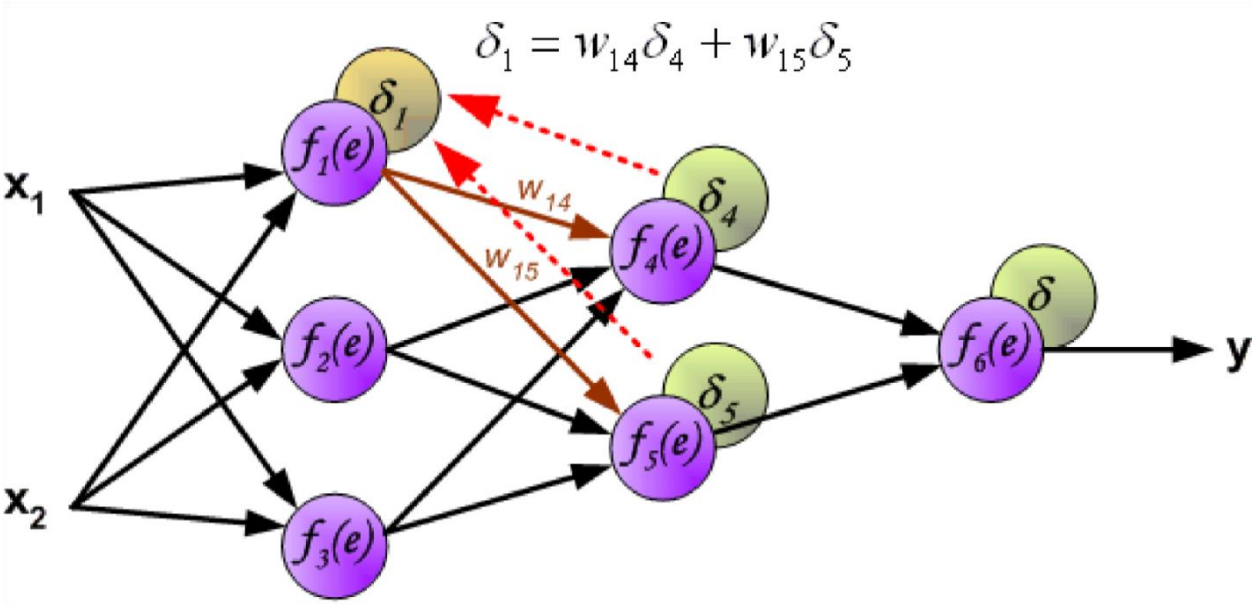

Рисунок 1.6 Сума значень вхідних нейронів, для поточного

Після того, як ми визначили величину помилки для кожного нейрона, можна скоригувати вагові коефіцієнти кожного вузла вводу. Для корегування ваг використовується похідна функції активації, яка обчислюється як:

$$deactivation = activation(x) \bullet (1 - activation(x)) \quad (1.1)$$

Крім того, у нас є коефіцієнт η, який впливає на швидкість навчання (див. Рисунок 1.7) [2].

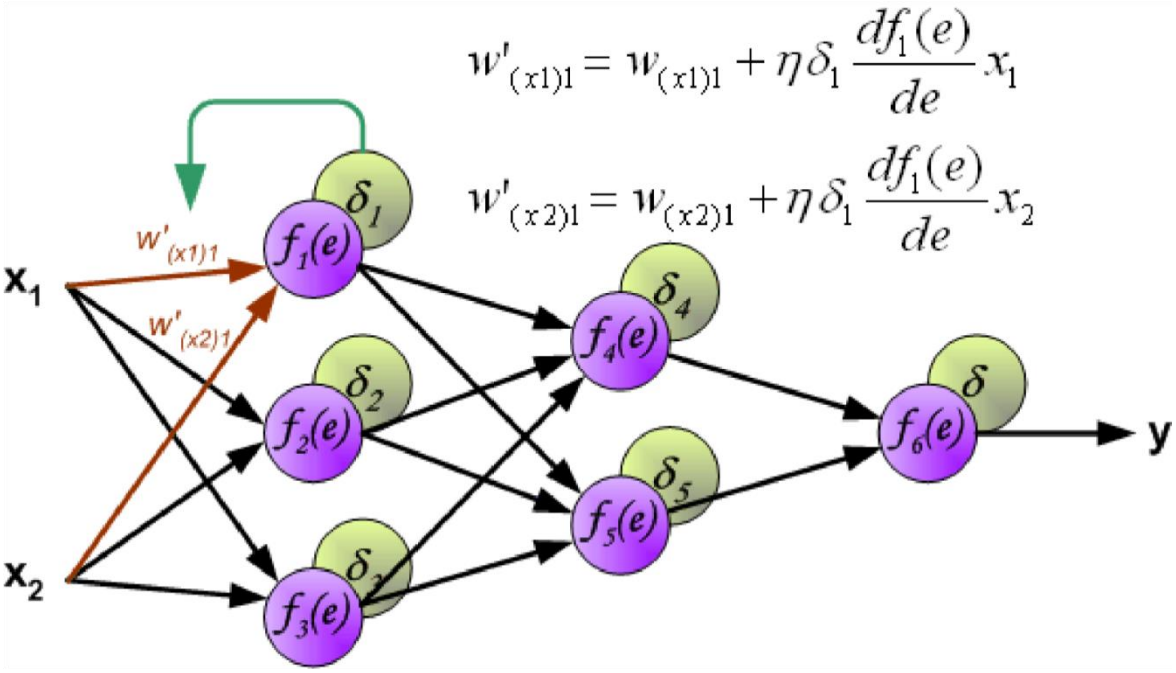

Рисунок 1.7 Вплив коефіцієнта на швидкість навчання

1.3 Штучний інтелект: інновації та автоматизація в сучасних компаніях

Використання штучного інтелекту в компанії Tesla для створення автопілоту є одним з інноваційних рішень, яке має значний потенціал та вплив на автомобільну промисловість та суспільство в цілому.

Автопілот Tesla використовує штучний інтелект, машинне навчання та комп'ютерне зорове сприйняття, щоб забезпечити автоматизоване керування автомобілем. ШІ системи аналізують велику кількість даних, зібраних з різних сенсорів, включаючи камери, радари та лідари, для розпізнавання

дорожньої ситуації, виявлення перешкод, розмітки дороги та інших автомобілів (див. Рисунок 1.8). Це дозволяє автомобілю самостійно реагувати на дорожні умови, уникати зіткнень та забезпечувати безпеку пасажирів та навколишніх.

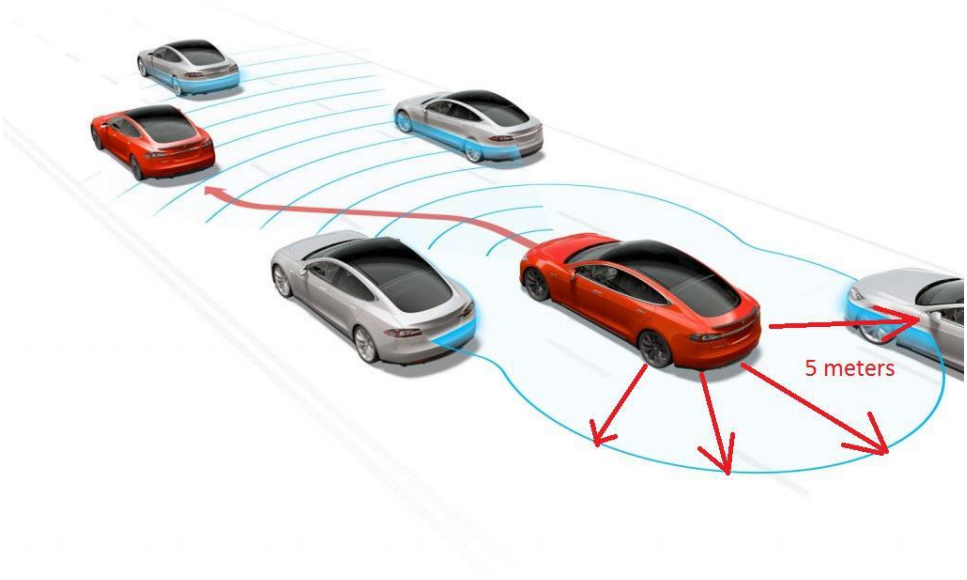

Рисунок 1.8 Робота датчиків на Tesla

Використання штучного інтелекту в автопілоті Tesla має потенціал покращити безпеку на дорозі та зробити її більш ефективною. ШІ системи можуть оперативно реагувати на змінні умови на дорозі та швидше приймати рішення, ніж людина. Вони можуть аналізувати великі обсяги даних, швидко реагувати на небезпечні ситуації та мінімізувати ризики.

У нових версіях автопілоту, компанія Tesla впроваджує додаткові функціональні можливості, які дозволяють автомобілю самостійно запаркуватися (див. Рисунок 1.9) або під'їхати до власника за допомогою простого натискання на кнопку.

Функція самопаркування дозволяє автомобілю знайти відповідне місце для паркування, навіть вузьке або обмежене просторове середовище. Коли водій активує цю функцію, автопілот використовує сенсори, камери та алгоритми ШІ, щоб розпізнати вільне місце для паркування. Потім автомобіль автоматично керує рухом, маневруючи для зайняття місця для

паркування. Це зменшує стрес та зусилля, які пов'язані з паркуванням, і сприяє більш точному та ефективному використанню доступного простору.

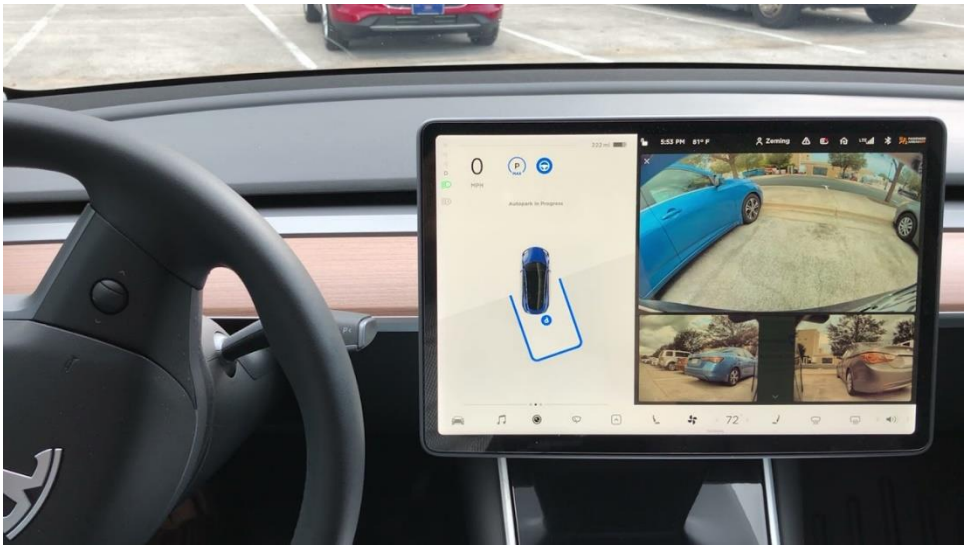

Рисунок 1.9 Самостійне паркування автомобілем Tesla

Крім того, кнопка "під'їхати до власника" є ще однією цікавою функцією в автопілоті Tesla. Коли власник автомобіля натискає на цю кнопку на мобільному додатку або на ключовій картці, автомобіль самостійно активує систему автопілота і рухається до власника, якщо він знаходиться на невеликій відстані. Це може бути особливо зручно, коли власник хоче, щоб автомобіль під'їхав ближче до нього для зручності виходу або завантаження речей.

Ці функції демонструють, як штучний інтелект в автопілоті Tesla надає додатковий рівень зручності та автоматизації. Вони роблять використання автомобіля більш простим та ефективним для власників, зменшуючи їх зусилля та час, пов'язані з паркуванням або пошуком автомобіля. Проте, важливо зазначити, що власники повинні користуватися цими функціями з відповідальністю та в рамках правових вимог та безпечних умов.

Microsoft використовує штучний інтелект не лише у своїй пошуковій системі Bing, але й впроваджує його на рівні операційної системи.

У пошуковій системі Bing (див. Рисунок 1.10), запуск ШІ-чату дозволив залучити нових користувачів та збільшити активність використання. Близько

третини активних користувачів Bing стали новими користувачами завдяки впровадженню ШІ-чату. Кількість запитів зросла, що свідчить про збільшену зацікавленість користувачів. Популярність режиму чату в Bing зростає, з близько 30% відвідувачів, які використовують цей режим під час своїх пошукових сесій. Це призводить до зростання кількості повідомлень, які обмінюються користувачі в рамках цього режиму.

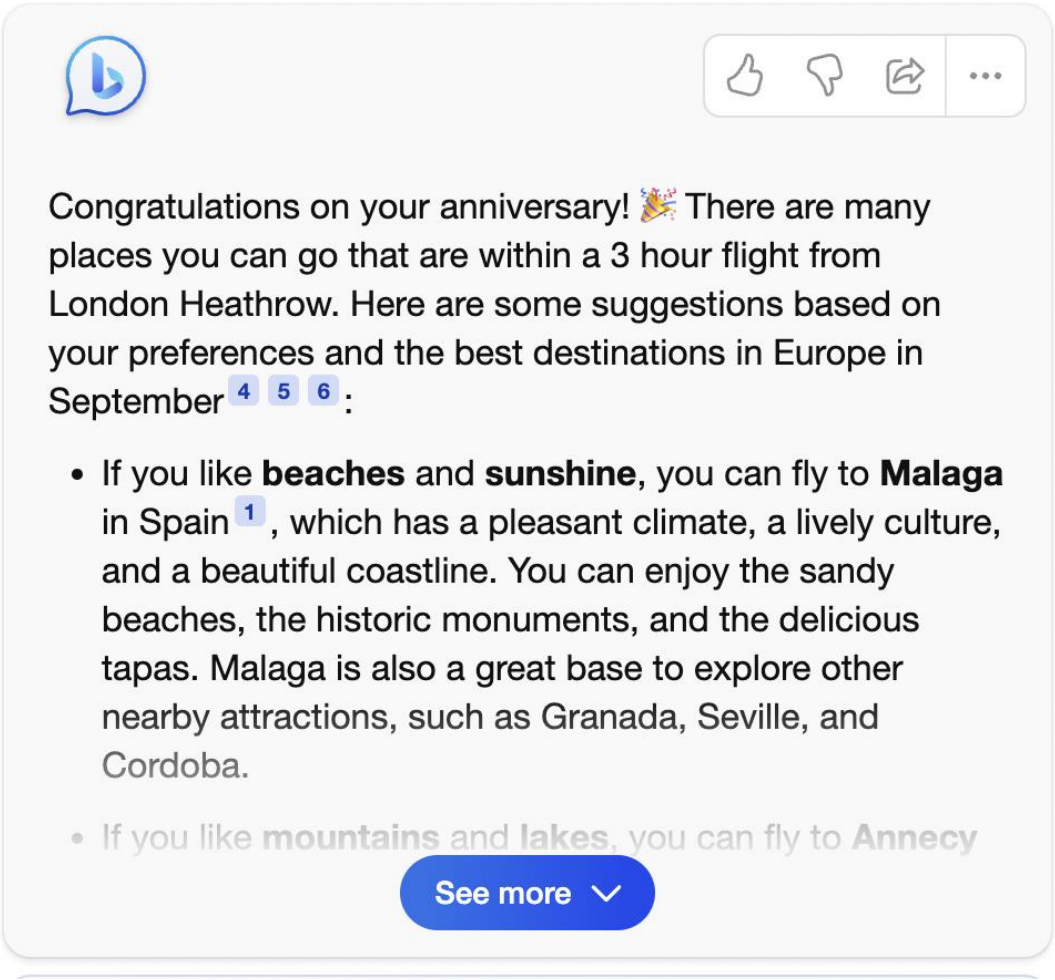

Рисунок 1.10 Пошукова система Bing

Крім пошукової системи, штучний інтелект компанії Microsoft також впроваджується використання на рівні операційної системи. Впровадження LLM в операційні системи дозволяє автоматизувати деякі рутинні завдання, покращує продуктивність та забезпечує користувачам більш інтуїтивний і персоналізований досвід використання комп'ютерів. Це може включати функції, такі як пошук файлів основаних на базі всіх файлів чи базі з окремої

папки, впровадження власної LLM, яка буде допомагати з рутинними задачами з вводу тексту чи пошуку потрібного тексту для власних потреб, це полегшує та покращує взаємодію користувача з комп'ютером.

1.4 Великий вибух: OpenAI (ChatGPT)

Після неймовірного успіху ChatGPT, ідеї про його використання та можливості почали ширитися по всьому інтернету. Люди виявили захоплення та зацікавлення новими можливостями, які надає цей штучний інтелект.

OpenAI змінює світ за допомогою свого ChatGPT, штучного інтелекту, шляхом створення потужного LLM зі здатністю генерувати тексти на різні теми та відповідати на запитання користувачів. Запуск ChatGPT викликав великий інтерес серед мільйонів користувачів упродовж короткого періоду. Цей інструмент з штучним інтелектом може використовуватися для різних цілей, включаючи отримання інформації, структурування текстів, створення резюме або стратегічних планів розвитку.

За використанням ChatGPT спостерігається різноманітність сценаріїв використання. Від задавання загальних запитань до отримання інформації до структурування тексту або навіть створення документів у різних стилях. Він може виступати в ролі вчителя, який надає пояснення та допомогу учням у вивченні різних предметів.

Крім цього, ідеї про використання ChatGPT знаходять застосування в сфері бізнесу. Він може служити як віртуальний асистент для клієнтів, виконувати завдання з обробки даних та аналітики, допомагати в розробці маркетингових стратегій та багато іншого.

Важливо зазначити, що ChatGPT все ще знаходиться в стадії навчання, і, як попереджає компанія, може надавати не завжди коректні відповіді або виявляти упереджену поведінку. Однак OpenAI прагне вдосконалювати

систему, враховуючи відгуки користувачів.

В одному зі своїх проектів, OpenAI зосередився на вивченні та навчанні системи штучного інтелекту в грі Dota 2 (див. Рисунок 1.11).

Dota 2 - це складна командна стратегічна гра, в якій гравці керують героями та спільно працюють для досягнення перемоги. OpenAI здійснив багато спроб залучити свої алгоритми до гри Dota 2 і вивчити навички гри на професійному рівні.

Рисунок 1.11. Використання OpenAI в Dota 2

У 2018 році OpenAI представила систему під назвою OpenAI Five, яка була навчена грати в Dota 2 на рівні професійних гравців. Ця система вивчала гру, граючи проти самих себе та шляхом самонавчання. OpenAI Five провела низку матчів проти професійних команд гравців Dota 2 та показала вражаючі результати (див. Рисунок 1.12).

OpenAI Five виявила високий рівень стратегічного мислення, комунікації та координації в грі. Вона проявила здатність до адаптації до змінних умов гри та вміння робити складні стратегічні рішення.

Вивчення і навчання в Dota 2 дозволяють OpenAI вдосконалювати свої алгоритми та розуміння штучного інтелекту. Цей проект викликав значний інтерес у геймінговій спільноті та підкреслив потенціал штучного інтелекту

в сфері вивчення складних стратегічних ігор.

Рисунок 1.12 Результати перемог OpenAI Five над геймерами [3]

У 2018 році, після значного покращення своїх навичок, OpenAI Five змогла здолати професійних гравців у серії матчів. Наприклад, в одному зі своїх виступів проти команди OG, володарів The International 2018, OpenAI Five здобула перемогу. Протягом того ж року OpenAI Five також здобула перемогу проти команди TNC Predator.

Важливо зазначити, що OpenAI Five не завжди була безпереможною. Вона також зазнала поразок у деяких матчах проти досвідчених гравців. Проте, показник вінрейту OpenAI Five, тобто відношення перемог до поразок, залишався високим і продемонстрував вражаючі здібності системи.

Точні числові дані щодо загальної кількості ігор та вінрейту OpenAI Five проти людей можуть бути недоступними або зміненими з часом, оскільки це активна галузь досліджень та розвитку штучного інтелекту. Однак, варто зауважити, що успіхи OpenAI Five в грі Dota 2 підкреслюють прогрес та потенціал штучного інтелекту у вирішенні складних завдань в ігровій сфері.

DALL·E - це модель штучного інтелекту, розроблена компанією OpenAI. DALL·E поєднує в собі два потужних підходи - генеративні моделі

тексту та моделі обробки зображень, що дозволяє їй створювати унікальні та творчі зображення на основі текстового опису.

Одна з найвизначніших особливостей DALL·E полягає в його здатності генерувати реалістичні зображення, які не обмежуються просто повторенням вхідного тексту (див. Рисунок 1.13). Він може виконувати завдання, де потрібно створювати нові, фантазійні зображення на основі опису, навіть якщо такі зображення ніколи не зустрічалися в навчальному наборі даних.

DALL·E базується на архітектурі GPT (Generative Pre-trained Transformer) і використовує сотні мільйонів параметрів для навчання. Він навчається на великому обсязі текстових та зображень даних, що дозволяє йому розуміти зв'язки між текстом та візуальними елементами.

Для генерації зображень DALL·E працює за принципом кодування та декодування. Спочатку текстовий опис перетворюється на числовий вектор, а потім цей вектор декодується в зображення за допомогою глибокої нейронної мережі. Після навчання модель може створювати нові зображення, комбінуючи різні концепти та елементи, що були побачені в навчальних даних.

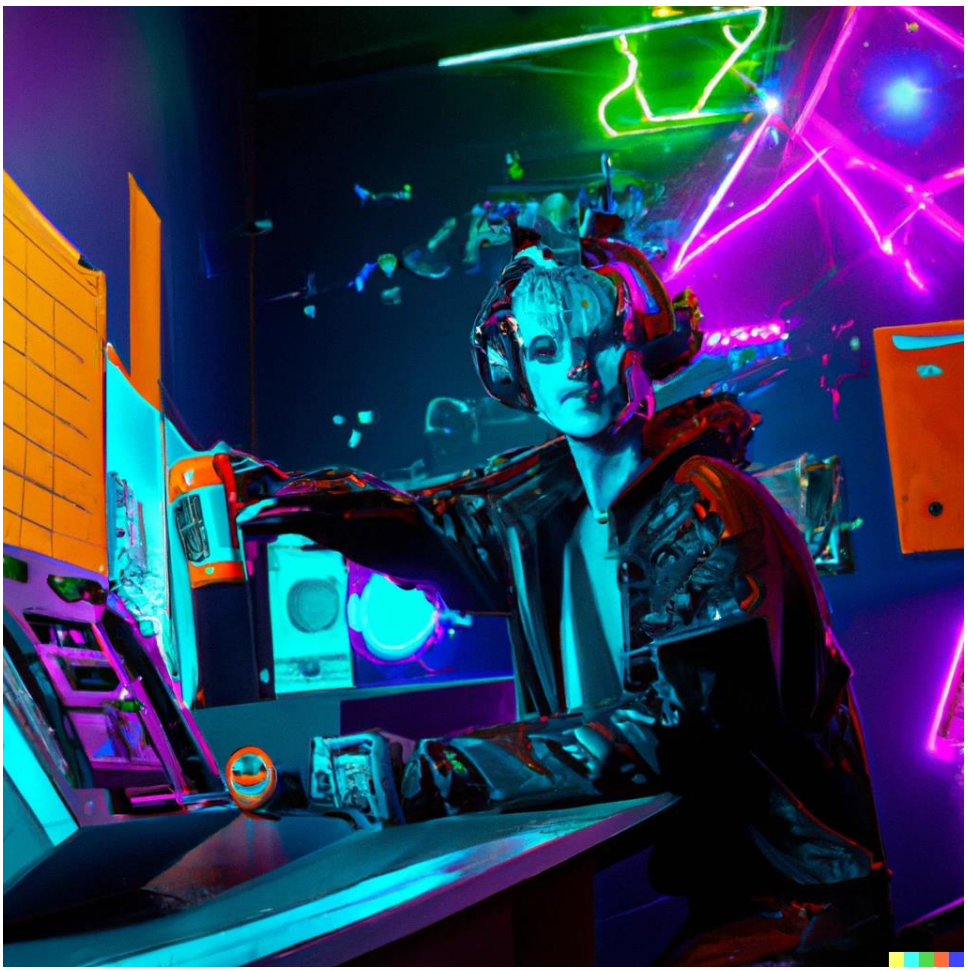

Рисунок 1.13 “Кіберпанк-монстр у диспетчерській” [4]

DALL·E має потенціал знайти застосування у багатьох сферах, включаючи творчість, дизайн, маркетинг, рекламу та інші галузі, де створення унікальних та цікавих зображень може бути важливим. Використовуючи DALL·E, можна створити нові художні твори, ілюстрації, рекламні матеріали та багато іншого, що відображає великий потенціал цієї моделі в креативних процесах.

У галузі дизайну та мистецтва DALL·E дав новий імпульс для творчого виразу. Художники та дизайнери використовують його для створення унікальних ілюстрацій (див. Рисунок 1.14), малюнків та графічних композицій. Завдяки широкому спектру вхідних текстових описів, DALL·E дозволяє створювати надзвичайно різноманітні твори мистецтва, включаючи фантастичні сцени, неіснуючі створіння та абстрактні композиції.

Рисунок 1.14 “Освітлена сонцем крита зона відпочинку з басейном із чистою водою та іншим басейном із напівпрозорою пастельно-рожевою водою, поруч із великим вікном, цифрове мистецтво” [4]

У сфері маркетингу та реклами DALL·E відкриває нові можливості для створення привабливих та оригінальних графічних зображень. Компанії можуть використовувати цю модель для створення унікальних логотипів, банерів, постерів та інших рекламних матеріалів. Інтерактивність DALL·E дозволяє підходити до кожного клієнта індивідуально, створюючи персоналізовані графічні елементи на основі їх вказівок.

Багато компаній використовують DALL·E для розробки нових продуктів та концепцій. Завдяки здатності моделі створювати зображення на основі текстових описів, вона може бути використана для швидкого прототипування та візуалізації ідей. Різноманітність та креативність генерації DALL·E дає змогу швидко експериментувати з різними концепціями та варіаціями продуктів.

1.5 Штучний інтелект: негативні наслідки, які потрібно ураховувати перед його використанням

Зростання штучного інтелекту також поставляє перед нами виклики та проблеми, зокрема з питань етики, безпеки та упередженості. Питання, пов'язані з тим, як забезпечити прозорість та відповідальність штучного інтелекту, а також як запобігти зловживанню та неконтрольованому розвитку ШІ, стають все більш актуальними.

Штучний інтелект має безліч потенційних переваг і може стати великим проривом в багатьох сферах. Проте, перед його використанням або додаванням до програми необхідно усвідомлювати його можливі негативні наслідки. Ось деякі з них:

1. Автономність та відсутність контролю. Штучний інтелект може мати непередбачувану поведінку, особливо якщо він недостатньо контрольований. Це може призвести до небезпечних ситуацій або неправильних висновків.

2. Алгоритмічна несправедливість. Штучний інтелект використовує дані для навчання, і якщо дані містять викривлення або приховану несправедливість, то і результати можуть бути несправедливими. Неправильні рішення, які базуються на упередженості або дискримінації, можуть призвести до негативних наслідків для людей.

3. Приватність і безпека. Використання штучного інтелекту вимагає збирання і обробки великого обсягу даних, що може викликати проблеми з приватністю і безпекою. Некоректне зберігання або використання цих даних може наражати особисті дані на ризик.

4. Вплив на робочі місця. Розвиток штучного інтелекту може мати вплив на робочі місця та зайнятість. Автоматизація завдань, роботизація та заміщення людей машинами можуть призвести до втрати робочих місць і соціально-економічних проблем.

5. Відповідальність та етичність. Штучний інтелект може стати

об'єктом зловживання або неналежного використання. Потрібно визначити відповідальність за дії штучного інтелекту та розробляти етичні стандарти для його використання.

6. Залежність від технології. Штучний інтелект може стати настільки невід'ємною частиною нашого життя, що люди можуть стати залежними від нього. Це може призвести до втрати навичок, зменшення самостійності та втрати контролю над рішеннями.

7. Втрата людського фактору. Штучний інтелект може заміняти людей у багатьох сферах, включаючи медицину, право, фінанси та інші. Це може призвести до втрати людського впливу, емоційного спілкування та індивідуального підходу, який можуть забезпечити лише люди.

8. Економічні нерівності. Розвиток штучного інтелекту може призвести до появи нових економічних нерівностей. Країни або організації з більшими ресурсами можуть мати перевагу в доступі до штучного інтелекту та його використанні, тоді як інші можуть залишатися позаду.

9. Відсутність морального розуміння. Штучний інтелект, незважаючи на свою високу продуктивність, не має морального розуміння та етичного свідомості, що може впливати на його прийняття рішень. Це може викликати конфлікти між моральними нормами та рішеннями, що зроблені штучним інтелектом.

10. Технічні проблеми та помилки. Використання штучного інтелекту супроводжує ризик технічних проблем та помилок. Неналежна настройка або непередбачені ситуації можуть призвести до неправильних результатів та негативних наслідків.

Ці негативні наслідки підкреслюють важливість обережного підходу до використання штучного інтелекту і необхідність розробки етичних принципів та правил для його застосування.

1.6. Формування простору ознак при ідентифікації рухомих об’єктів

1.6.1 Відстеження з використанням фільтра Калмана

Однією з ключових проблем у сфері відеоспостереження є ефективне виявлення та відстеження людей у реальному часі. З метою поліпшення цього процесу, була розроблена система відстежування людей в режимі реального часу з використанням кількох камер (див. Рисунок 1.15). Загалом, ймовірні підходи до відстеження об'єктів намагаються оцінити поточний стан або розташування об'єкта на основі історії попередніх станів. Одним із таких методів є фільтр Калмана (див. Рисунок 1.16).

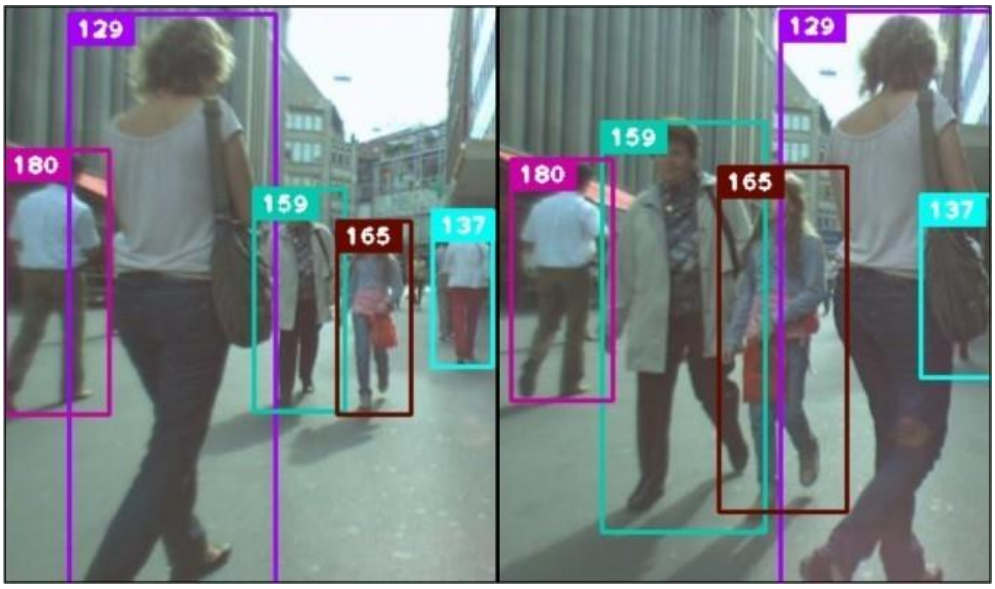

Рисунок 1.15 Приклад роботи відстеження обьектів. Ідентифікатори сходяться на обох фотографіях [5]

Цей алгоритм складається з двох рекурсивних кроків: передбачення та оновлення (див. Рисунок 1.16).

На кроці передбачення фільтр Калмана оцінює новий стан відстежуваного об'єкта, використовуючи інформацію про попередній стан. На кроці оновлення фільтр зчитує нові спостереження шуму для оновлення

ймовірності розподілу станів перед наступним кроком передбачення. Таким чином, фільтр Калмана будує модель системи, яка максимізує ймовірність попередніх вимірювань. В основі алгоритму лежать припущення, що стани об'єкта та шум, якими вони підлягають, розподілені нормально, а система має лінійний характер, тобто кожний стан може бути представлений матрицею, помноженою на попередній стан [5].

Фільтр Калмана є потужним інструментом для оцінки стану системи. Він забезпечує баланс між точністю та обчислювальною ефективністю, що робить його важливим інструментом у багатьох областях, включаючи автоматичне відстеження об'єктів. Він має свої недоліки, якщо система має нелінійну динаміку або шум не відповідає нормальному розподілу, тоді можуть виникнути неточності в оцінках [5].

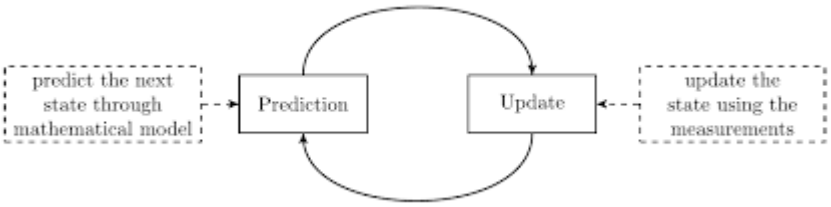

Рисунок 1.16 Фільтр Калмана [5]

1.6.2 Відстеження рухомих об’єктів з використанням алгоритму найближчих сусідів (K-nearest neighbors, K-NN)

Алгоритм найближчих сусідів (K-nearest neighbors, K-NN) - це один з найпростіших та найпоширеніших алгоритмів машинного навчання в області класифікації та регресії. Він використовується для відстеження та класифікації об'єктів на основі їхніх ближніх сусідів у вхідних даних.

Принцип роботи алгоритму полягає в наступному:

1. Збирання навчальних даних: Спочатку необхідно мати набір

навчальних даних, який складається з вхідних об'єктів і відповідних класів або значень, які ми намагаємося передбачити.

2. Вибір параметра K: Параметр K визначає кількість найближчих сусідів, які будуть використовуватися для класифікації нового об'єкта. Це число може бути задане вручну або визначатися шляхом пошуку найкращого значення через перехресну перевірку.

3. Визначення відстаней: Для кожного нового об'єкта обчислюється відстань від нього до всіх навчальних об'єктів у просторі ознак. Відстань може бути обчислена за допомогою різних метрик, таких як Евклідова відстань, Манхеттенська відстань або косинусна відстань.

4. Вибір K найближчих сусідів: З використанням обчислених відстаней вибираються K найближчих сусідів до нового об'єкта. Це можна зробити шляхом сортування відстаней у порядку зростання та вибору K найменших.

5. Класифікація нового об'єкта: Після вибору K найближчих сусідів вирішується, який клас чи значення присвоювати новому об'єкту. Це може бути зроблено за допомогою голосування більшості, де кожен сусід має голос залежно від свого класу або значення, або шляхом зваженого голосування, де кожен сусід має вагу, залежну від його відстані до нового об'єкта.

6. Оцінка точності: Після класифікації нового об'єкта можна оцінити точність алгоритму, порівнявши прогнозований клас або значення з відомими значеннями в навчальних даних. Це допомагає визначити ефективність алгоритму і, за потреби, внести корективи.

Алгоритм найближчих сусідів є простим і інтуїтивно зрозумілим методом класифікації та регресії. Він може бути застосований до широкого спектру задач машинного навчання. Однак, він також має деякі обмеження, такі як чутливість до шуму та великої кількості ознак, а також вимогу до наявності достатньої кількості навчальних даних.

В багатьох прикладних завданнях вимірювання подібності об'єктів є простішим за формування ознакових описів. Наприклад, фотографії осіб, підписи, часові ряди і структури білків є складними об'єктами, які більше

піддаються порівнянню шляхом "накладення з вирівнюванням" без необхідності створення окремих ознакових описів і їх порівняння. Коли міра подібності об'єктів встановлена належним чином, зазвичай виявляється, що схожим об'єктам часто відповідають подібні відповіді. У задачах класифікації це означає, що класи формують компактні локалізовані підмножини, і це припущення називається гіпотезою компактності. Для формалізації поняття "подібності" вводиться функція відстані в просторі об'єктів. Методи навчання, які базуються на аналізі подібності об'єктів, називаються метричними, навіть якщо функція відстані не відповідає всім аксіомам метрики (зокрема, аксіомі трикутника).

Алгоритм найближчого сусіда є одним з перших і найпростіших евристичних методів для вирішення задачі комівояжера. Він відноситься до категорії жадібних алгоритмів і на кожному кроці до знайденої частини маршруту додається нове ребро. Алгоритм припиняє роботу, коли знайдено розв'язок і не намагається його поліпшити.

Алгоритм найближчого сусіда (nearest neighbor, NN) відносить класифікується об'єкт $u \in Xl$ до того класу, якому належить найближчий навчальний об'єкт:

$$w(i,u) = [i = 1]; a(u, Xl) = yu(-1)$$

Алгоритм k найближчих сусідів (k nearest neighbors, kNN) [1]. Щоб згладити вплив викидів, будемо відносити об'єкт u до того класу, елементів якого виявиться більше серед k найближчих сусідів $xu(i), i = 1,...,k$:

$$w(i,u) = [i \le k];$$
$$a(u; Xl, k) = \arg\max \sum_{i=1} [yu(i) = y]k$$

При k = 1 цей алгоритм збігається з попереднім, отже, нестійкий до шуму.

При k =l, навпаки, він надмірно стійкий і вироджується в константу. На практиці оптимальне значення параметра k визначають за критерієм змінного контролю з виключенням об'єктів по одному (leave-one-out, LOO). Для кожного об'єкта $xi \in Xl$ перевіряється, чи правильно він класифікується за

своїми k найближчих сусідів. $LOO(k, Xl) = \sum_{i=1}^{\min k} l[a(xi; Xl\{xi\}, k) \neq yi]$ [6].

Зауважимо, що якщо класифікується об'єкт xi не виключати з навчальної вибірки, то найближчим сусідом xi завжди буде сам xi , і мінімальне (нульове) значення функціоналу LOO (k) буде досягатися при k = 1. Існує і альтернативний варіант методу kNN: в кожному класі вибирається k найближчих до u об'єктів, і об'єкт u відноситься до того класу, для якого середня відстань до k найближчих сусідів мінімальна [6].

1.7 Підсумки

Оглядаючи штучний інтелект та його використання в різних галузях, таких як комунікації, творчість та автономна навігація, можна зробити висновок про його переваги і недоліки.

Великі мовні моделі Large Language Model (LLM), такі як ChatGPT та DALL·E, виявляються корисними інструментами в галузі комунікацій. Вони здатні розуміти та генерувати людську мову з вражаючою точністю і творчістю, що відкриває нові можливості для покращення взаємодії між людьми та машиною.

Однак, необхідно враховувати й недоліки штучного інтелекту. Недосконаліть великих мовних моделей може призводити до появи помилок і неправильностей у виведеннях, що потребує обережного підходу та перевірки з боку людей.

РОЗДІЛ 2. ВИКОРИСТАННЯ ШІ В ОБРОБЦІ ДОКУМЕНТІВ

2.1 ReactJS у глибині: Використання react-router та redux для масштабованих веб-додатків

У React ви можете будувати інтерфейси користувача з окремих частин, які називаються компонентами. Створіть свої власні компоненти React, такі як Thumbnail (мініатюра), LikeButton (кнопка "Подобається") та Video (відео). Потім поєднайте їх в цілі екрани, сторінки та додатки [7]

ReactJS є однією з найпопулярніших бібліотек для розробки інтерфейсів користувача (UI) у веб-додатках. Випущений компанією Facebook у 2013 році, ReactJS дозволяє розробникам будувати складні веб-додатки, які ефективно оновлюються та масштабуються.

Основною концепцією ReactJS є використання компонентів. Компоненти є будівельними блоками веб-додатків і можуть бути створені для різних частин інтерфейсу, таких як кнопки, форми, списки, заголовки тощо. Кожен компонент має внутрішній стан, який визначає його поведінку та вигляд. Компоненти можуть бути вкладені один в одного, що дозволяє побудувати складні ієрархії інтерфейсу.

ReactJS використовує Virtual DOM (віртуальний об'єктний модель) для ефективного оновлення інтерфейсу. Коли стан додатку змінюється, ReactJS оновлює тільки змінені елементи в Virtual DOM, а не всю сторінку. Це забезпечує швидке та ефективне оновлення інтерфейсу та поліпшує продуктивність додатку.

Один з основних принципів ReactJS - "однозв'язковість даних". Це означає, що стан додатку повинен бути однозначно пов'язаним зі станом його компонентів. Коли стан змінюється, ReactJS автоматично оновлює компоненти, які залежать від цього стану. Це спрощує управління станом і

забезпечує послідовність даних.

ReactJS також підтримує використання JSX (розширений синтаксис JavaScript), що дозволяє розробникам писати код з вбудованим HTML-подібним синтаксисом. JSX дозволяє використовувати HTML-подібні теги для опису компонентів та їх структури, що полегшує розуміння та розвиток коду.

ReactJS також має бібліотеки та додатки, які полегшують розробку і покращують продуктивність. Наприклад, Redux (див. Рисунок 2.1) є популярним додатком для керування станом додатків React, а React Router дозволяє визначати маршрутизацію у веб-додатках.

ReactJS є однією з найпопулярніших технологій у сфері веб-розробки, засвідченою великою кількістю проектів, активною спільнотою розробників та широким застосуванням в індустрії. Ось деякі ключові аспекти, які спричинили популярність ReactJS:

Простота використання: ReactJS пропонує простий та зрозумілий API, який дозволяє розробникам швидко оволодіти його концепціями та використовувати його для створення потужних інтерфейсів користувача.

Компонентна архітектура: ReactJS заснований на компонентах, що сприяє модульності та повторному використанню коду. Компоненти можуть бути легко створені, комбіновані та перевикористані, що полегшує розробку та підтримку великих проектів.

Віртуальний DOM: ReactJS використовує віртуальний DOM, що дозволяє оптимізувати процес оновлення інтерфейсу. Замість безпосередньої маніпуляції реальним DOM, ReactJS порівнює віртуальний DOM з попереднім станом, і лише оновлює необхідні елементи, що забезпечує високу продуктивність додатків.

Реактивність: ReactJS надає можливість автоматичного оновлення інтерфейсу при зміні стану компонентів. Це дає розробникам зручність і спрощує роботу з динамічними елементами інтерфейсу.

Велика спільнота: ReactJS має велику та активну спільноту

розробників, що забезпечує підтримку, навчання та поширення знань. Існує безліч ресурсів, документацій, бібліотек та інструментів, які сприяють швидкому розвитку та розширенню екосистеми ReactJS.

Широке застосування: ReactJS може використовуватись для розробки різноманітних типів додатків, включаючи односторінкові додатки, мобільні додатки за допомогою React Native, десктопні додатки за допомогою Electron, а також інтеграцію з різними фреймворками та бібліотеками.

Підтримка від Facebook: ReactJS розроблений командою Facebook, що забезпечує його активний розвиток, підтримку та оновлення. Це надає впевненість розробникам у якості та майбутньому бібліотеки.

Redux є становим контейнером та управлінцем стану для JavaScript-додатків, який використовується в основному у сполученні з бібліотекою ReactJS. Redux надає однозначний та прогнозований спосіб керування станом додатків, що допомагає зберігати, оновлювати та синхронізувати дані у всьому додатку.

Основна концепція Redux базується на одному центральному об'єкті стану, який зберігається у пам'яті. Цей стан представляє собою деревоподібну структуру, яка містить всі дані, необхідні для роботи додатку. Стан Redux є неізменним, тобто його неможливо змінити безпосередньо. Замість цього, будь-які зміни стану повинні бути виконані через спеціальні функції під назвою "дії" (actions).

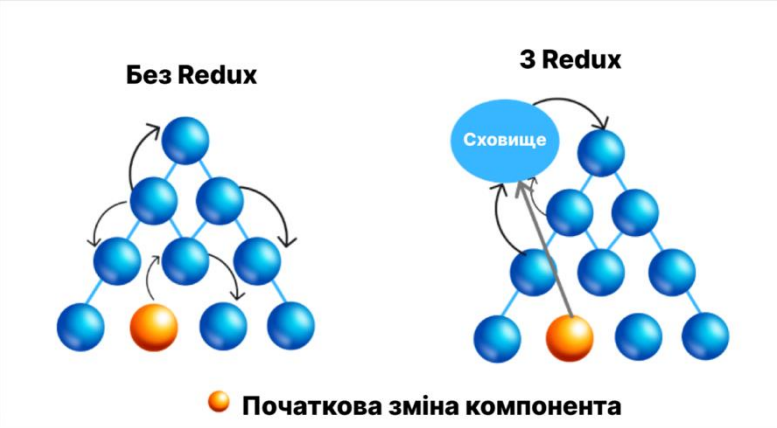

Рисунок 2.1 Порівняння двух типів передачі даних в ReactJS [8]

Дії визначаються як об'єкти з типом (type) та необов'язковими

додатковими даними. Коли відбувається дія, Redux виконує чергування її через "підсумовувач" (reducer). Редуктор - це чиста функція, яка приймає поточний стан та дію, і повертає новий стан. Цей процес зміни стану відбувається виключно за допомогою незмінних операцій, що дозволяє легше слідкувати за змінами та відлагоджувати додаток.

Центральний об'єкт стану Redux доступний для всіх компонентів додатку. Для звернення до стану та виконання змін використовуються "контейнери" (containers). Контейнери є спеціальними компонентами, які зв'язуються зі станом Redux та передають дані та дії до презентаційних компонентів. Це полегшує розподіл стану та логіку додатку між компонентами та сприяє покращенню масштабованості та керованості додатку.

Redux також надає можливість використання "середників" (middlewares) для перехоплення та обробки дій перед тим, як вони досягнуть редуктора. Це дозволяє виконувати асинхронні операції, логування, маршрутизацію та інші операції. Середники можуть бути настроєні та розширені залежно від потреб додатку.

Одна з головних переваг Redux полягає у простоті тестування. Завдяки однозначному та відокремленому стану, діям та редукторам, тести на стан Redux можуть бути легко написані та виконані, забезпечуючи впевненість у стабільності та коректності додатку.

Redux є популярним вибором для керування станом у ReactJS-додатках, але його можна використовувати і з іншими бібліотеками та фреймворками JavaScript. Ця бібліотека допомагає забезпечити однозначність, прогнозованість та легкість керування станом у великих та складних додатках.

React Router є бібліотекою для маршрутизації веб-додатків на основі ReactJS. Вона надає зручний спосіб управління маршрутами (URL) та відображення відповідних компонентів на основі поточного шляху (route) в браузері. React Router дозволяє створювати односторінкові додатки (SPA) з багатосторінковою навігацією та підтримкою ряду функцій, таких як параметри маршруту, вкладені маршрути, переадресації та багато іншого.

Основні концепції та компоненти, що використовуються в React Router:

1. BrowserRouter: Це компонент, який надає маршрутизацію на основі розділу історії браузера (HTML5 History API). Він дозволяє синхронізувати URL з відображеними компонентами.

2. Route: Цей компонент визначає зв'язок між шляхом (path) та компонентом, який має бути відображений, коли URL відповідає цьому шляху. Можна використовувати параметри в шляху для передачі динамічних даних до компонента.

3. Switch: Компонент Switch використовується для забезпечення відображення лише одного маршруту одночасно. Він переглядає всі дочірні Route-компоненти та відображає перший, чий шлях відповідає поточному URL.

4. Link: Це компонент, який надає зручний спосіб створення посилань, які автоматично оновлюють URL і сприяють навігації в додатку без перезавантаження сторінки.

5. Redirect: Цей компонент використовується для переадресації з одного шляху на інший. Він корисний, коли потрібно перенаправити користувача після певних дій або за умови.

React Router також підтримує багато інших функцій, таких як вкладені маршрути, захист маршрутів, передача параметрів через запити тощо. Він є потужним інструментом для створення складних навігаційних структур у веб-додатках на ReactJS.

За допомогою React Router можна створювати динамічну навігацію, розміщати компоненти на різних шляхах та керувати станом додатка на основі поточного URL. Він забезпечує зручну і покращену взаємодію користувача з додатком, полегшуючи навігацію, пошук і сприяючи кращому досвіду користувача.

Узагалі, ReactJS є потужним інструментом для розробки сучасних веб-

додатків. Він надає широкі можливості для побудови ефективного та легко розширюваного інтерфейсу користувача. За допомогою ReactJS ви можете розробляти великі проекти, де керування станом і оновленням інтерфейсу є простими та ефективними.

Код react-router:

```
import React from 'react';
import {BrowserRouter, Routes, Route } from "react-router-dom"
import Dashboard from "./pages/dashboard.jsx"
import Profile from "./pages/profile.jsx"
import Card from "./pages/card.jsx"

const App = () => (
 return {
 <BrowserRouter>
        <Routes>
            <Route index element={<Dashboard/>}/>
            <Route path="profile" element={<Profile/>}/>
            <Route path="card/:id" element={<Card/>}/>
        </Routes>
 </BrowserRouter>
 }
);
```

2.2 Стилізація ReactJS за допомогою TailwindCSS

Tailwind CSS є одним з найпопулярніших CSS-фреймворків, призначених для швидкої та ефективної розробки веб-інтерфейсів. Він пропонує інноваційний підхід до написання CSS, використовуючи концепцію "utility classes" (класів-утиліт), яка надає гнучкість та швидкість у створенні стилів (див. Рисунок 2.2).

Рисунок 2.2 Зручність використання TailwindCSS [9]

Основні принципи та особливості Tailwind CSS:

1. Utility-first: Tailwind CSS побудований на концепції utility-класів. Замість написання окремих CSS-правил для кожного елементу, ви використовуєте класи-утиліти, які надають маленькі та повторно використовувані стилі. Наприклад, клас `text-center` вирівнює текст по центру, а `bg-blue-500` встановлює фоновий колір блоку на синій.
2. Модульність: Tailwind CSS має широкий спектр utility-класів, що охоплюють всі аспекти стилізації, включаючи розміри, кольори, падіння (margin) та відступи (padding), шрифти, границі, фони та багато іншого.

Ви можете комбінувати ці класи, щоб швидко створити потрібний дизайн.

3. Настроювання: Tailwind CSS надає широкі можливості для настроювання фреймворку. Ви можете змінювати кольорову палітру, розміри, шрифти та інші значення, використовуючи конфігураційний файл. Це дозволяє створювати унікальні стилі, які відповідають вимогам вашого проекту.
4. Респонсивність: Tailwind CSS має вбудовану підтримку для респонсивного дизайну. Ви можете використовувати префікси, такі як `sm:`, `md:`, `lg:`, щоб задавати різні стилі для різних розмірів екрану. Це дозволяє легко адаптувати ваші стилі до різних пристроїв і забезпечувати гарний вигляд на будь-якому пристрої.
5. Швидкість та продуктивність: Tailwind CSS допомагає збільшити продуктивність розробки, оскільки ви можете швидко застосовувати стилі, використовуючи готові класи-утиліти. Він також має оптимізовану і легку базу стилів, що дозволяє зменшити розмір вихідного коду та прискорити завантаження сторінок.
6. Сумісність з React та іншими фреймворками: Tailwind CSS можна використовувати в поєднанні з будь-яким фреймворком або бібліотекою, включаючи React, Vue.js [10], Angular [11] та інші. Він не накладає обмежень на вибір технології, що дозволяє розробникам використовувати його в будь-якому проекті.

Розробники обирають Tailwind CSS замість інших інструментів, таких як Sass, Less, CSS або Material UI, з кількох причин:

1. Простота використання: Tailwind CSS має простий синтаксис та невеликий набір правил. Він не має складної структури або вбудованих компонентів, що полегшує розуміння та використання фреймворку. Розробники швидко освоюють його та можуть швидко створювати стилі для своїх проектів.

2. Флексибільність: Tailwind CSS дає вам повний контроль над стилями, оскільки ви можете налаштувати кольорову палітру, розміри, шрифти та багато іншого в конфігураційному файлі. Ви також можете легко додавати власні стилі та класи до проекту, що дозволяє створювати унікальний вигляд для вашого додатку.
3. Швидкість та продуктивність: Tailwind CSS має маленький розмір файлу та оптимізовану базу стилів. Це дозволяє зменшити розмір вихідного коду та прискорити завантаження сторінок. Крім того, використання utility-класів дозволяє швидко застосовувати стилі без необхідності писати додатковий CSS-код.

Tailwind CSS - це потужний CSS-фреймворк, який надає швидкість, гнучкість та простоту у створенні стильних веб-інтерфейсів. Використовуючи utility-класи, ви можете швидко створювати та змінювати стилі, забезпечуючи ефективну розробку і гарний зовнішній вигляд вашого додатку.

2.3 NextJS: Інноваційність та розширення можливостей ReactJS

Next.js [12] є повноцінним фреймворком реактивного веб-розробки, який можна вважати "fullstack" фреймворком. Це означає, що він надає засоби та зручності для розробки як клієнтської, так і серверної частин вашого додатка.

На клієнтській стороні, Next.js використовує React.js для побудови інтерактивних користувацьких інтерфейсів. Ви можете створювати компоненти, керувати станом додатка, реагувати на події та взаємодіяти з API. React.js забезпечує швидку та ефективну роботу з DOM і забезпечує декларативний підхід до побудови інтерфейсів.

На серверній стороні, Next.js надає можливість виконувати серверний

рендеринг веб-сторінок. Це означає, що ви можете генерувати HTML на сервері перед надсиланням його клієнту. Це дозволяє покращити швидкість завантаження сторінок, поліпшити індексацію пошуковими системами та забезпечити кращий досвід для користувачів. Крім того, Next.js підтримує статичну генерацію, що дозволяє попередньо генерувати статичні HTML-файли для сторінок з відомим наперед вмістом.

Vercel, хоч і є незалежною хмарною платформою, базується на AWS (Amazon Web Services) архітектурі. AWS є однією з провідних хмарних платформ у світі, яка надає широкий спектр послуг для хмарного обчислення та інфраструктури.

Vercel використовує інфраструктуру AWS для розгортання та керування додатками та статичними сайтами. Це означає, що весь обробка та зберігання даних, масштабування, безпека і моніторинг здійснюються на базі AWS. Vercel спирається на надійність та масштабованість інфраструктури AWS для забезпечення стабільності та високої продуктивності своїх сервісів.

Використання AWS архітектури дозволяє Vercel забезпечити глобальну наявність своїх сервісів. AWS має дата-центри в різних країнах і регіонах по всьому світу, що дозволяє Vercel розгортати додатки ближче до користувачів, забезпечуючи низьку латентність і швидкий доступ до контенту.

Також варто відзначити, що співпраця з AWS дозволяє Vercel використовувати широкий набір послуг, які пропонує AWS. Це включає в себе такі послуги, як обчислення, зберігання даних, мережі, безпеку, аналітику та багато інших. Використання цих послуг дозволяє Vercel побудувати потужну та розширювану інфраструктуру для своїх клієнтів.

Основні особливості та переваги Next.js:

1. Серверний рендеринг та статичний генерація: Next.js надає можливість виконувати серверний рендеринг веб-сторінок, що дозволяє генерувати HTML на сервері перед надсиланням його клієнту. Це покращує швидкість завантаження сторінок та покращує індексацію сторінок пошуковими системами. Крім того, Next.js підтримує статичну

генерацію, що дозволяє попередньо генерувати статичні HTML-файли для сторінок з відомим наперед вмістом, що є корисним для сторінок зі статичним контентом.

2. Роутинг: Next.js надає простий та потужний механізм роутингу, що дозволяє легко організовувати навігацію у вашому додатку. Ви можете створювати динамічні маршрути, передавати параметри та керувати станом додатка на основі URL.
3. Hot Module Replacement: Next.js підтримує гарячу заміну модулів (Hot Module Replacement), що дозволяє автоматично оновлювати змінені модулі без перезавантаження сторінки. Це спрощує розробку та прискорює процес ітерації.
4. Широкий екосистема та плагіни: Next.js має широку екосистему плагінів та розширень, які дозволяють розширити його функціональність. Ви можете використовувати плагіни для додавання функціональності, такої як аналітика, оптимізація зображень, обробка форм і багато іншого.
5. Підтримка TypeScript: Next.js має вбудовану підтримку TypeScript, що дозволяє розробникам писати типізований код для покращення безпеки та підвищення продуктивності.
6. Статичне та динамічне рендеринг компонентів: Next.js дозволяє вам використовувати як статичний, так і динамічний рендеринг компонентів. Ви можете визначити, які частини вашого додатка повинні бути побудовані на сервері, а які - на клієнті. Це дає вам гнучкість у виборі найкращого підходу для вашого проекту.
7. Підтримка CSS-in-JS: Next.js має підтримку CSS-in-JS, включаючи популярні бібліотеки, такі як Styled Components і Emotion. Це дозволяє вам створювати компоненти, які включають у себе стилі, що полегшує організацію та управління стилями вашого додатка.
8. API маршрутизація: Next.js має вбудовану підтримку API маршрутизації, що дозволяє легко створювати серверні маршрути для обробки запитів API. Ви можете створювати власні кінцеві точки API та

обробляти запити з клієнтського коду або з інших серверних частин вашого додатка.

9. Розширені можливості маршрутизації: Next.js надає розширені можливості маршрутизації, такі як динамічні маршрути, параметри шляху, запити до сервера під час побудови, обмеження маршрутів та багато іншого. Це дозволяє легко налаштовувати поведінку маршрутів вашого додатка залежно від потреб вашого проекту.
10. Модульність і компонентний підхід: Next.js сприяє розділенню вашого додатка на компоненти, що полегшує організацію, тестування та повторне використання коду. Ви можете створювати повторно використовувані компоненти, що полегшує розширення та підтримку вашого додатка з часом.
11. Статичний експорт і развертання: Завдяки можливості статичного експорту, Next.js дозволяє згенерувати статичні HTML-файли для всього вашого додатка. Це полегшує розгортання додатка на будь-якому хостингу, в тому числі на статичних хостинг-провайдерах. Крім того, Next.js підтримує інтеграцію з популярними платформами розгортання, такими як Vercel, що дозволяє легко розгортати ваш додаток на хмарних сервісах.
12. Підтримка гарячої модифікації: Next.js надає можливість гарячої модифікації, що дозволяє вам змінювати код вашого додатка без перезавантаження сервера або клієнта. Це допомагає прискорити процес розробки та полегшує швидку зміну та тестування вашого коду.
13. Розширена підтримка головок HTTP: Next.js дозволяє налаштовувати головки HTTP, що надсилаються для кожного запиту. Це дає вам контроль над кешуванням, безпекою, автентифікацією та багатьма іншими аспектами вашого додатка.
14. Широке співтовариство та документація: Next.js має велике та активне співтовариство розробників, яке надає підтримку, документацію, навчальні ресурси та приклади коду. Це полегшує початок роботи з

фреймворком та надає доступ до великої кількості знань та ресурсів для розробки вашого додатка.

Однак важливо зауважити, що хоча Next.js надає засоби для розробки повноцінних серверних додатків, він не є заміною для традиційних серверних фреймворків, таких як Express або Koa. Використання Next.js для серверного коду зазвичай обмежується веб-сторінками, API маршрутами та логікою, пов'язаною з рендерингом. Для складніших серверних сценаріїв можна поєднувати Next.js з іншими фреймворками або використовувати його як частину ширшого стеку розробки.

2.4 Синергія між NextJS та Vercel: Максимізація зручності розробки та розгортання веб-додатків

Vercel - це хмарна платформа, яка спеціалізується на розгортанні веб-додатків та статичних сайтів. Вона надає розробникам зручність і простоту управління інфраструктурою, дозволяючи їм швидко розгортати свої проекти та зосередитися на розробці програмного забезпечення.

Одна з важливих переваг Vercel полягає в його простоті використання. Користувачі можуть швидко налаштувати та розгорнути свої проекти, використовуючи лише кілька кроків. Підключення Git-репозиторію та конфігурування параметрів здійснюється інтуїтивно зрозумілим інтерфейсом.

Vercel також відомий своєю швидкістю та масштабованістю. Завдяки розподіленій мережі серверів, розташованих по всьому світу, додатки, розгорнуті на Vercel, можуть працювати ближче до користувача, забезпечуючи швидкий час відгуку та завантаження сторінок. Платформа автоматично масштабує додатки, щоб впоратися зі зростаючим навантаженням, забезпечуючи їх ефективну роботу.

Vercel підтримує різні методи рендерингу, зокрема статичний рендеринг

та серверний рендеринг. Це дозволяє розробникам генерувати статичні HTML-сторінки або виконувати рендеринг на сервері для кожного запиту. Такий підхід полегшує оптимізацію сторінок для швидкості завантаження та поліпшує індексацію пошуковими системами.

Платформа Vercel має вбудовану підтримку безперервної поставки (CI/CD), що дозволяє автоматично розгортати додатки при змінах у вихідному коді. Ви можете підключити свої Git-репозиторії, такі як GitHub або GitLab, і налаштувати автоматичну розгортку при коміті або пул-реквесті.

Крім того, Vercel має інтеграцію з різними сервісами, що спрощує взаємодію розробників з іншими улюбленими інструментами розробки та управління контентом, такими як Contentful, Sanity та інші.

Останнім аспектом є аналітика та моніторинг. Vercel надає засоби для відстеження трафіку, часу завантаження сторінок, помилок та інших метрик, що дозволяє розробникам отримувати інформацію про ефективність своїх додатків і вносити необхідні покращення.

Узагальнюючи, Vercel - це хмарна платформа, яка надає розробникам простоту використання, швидкість, підтримку різних методів рендерингу, безперервну поставку, інтеграцію з іншими сервісами та засоби аналітики та моніторингу. Вона допомагає розробникам ефективно розгортати та керувати своїми проектами, забезпечуючи надійну та швидку роботу їх веб-додатків та статичних сайтів.

2.5 Оптимальне зберігання даних в AWS S3 та PostgreSQL: управління файлами та структуровані дані

PostgreSQL є потужною та розширеною системою керування базами даних (СКБД), яка надає надійне та гнучке зберігання та управління даними. Ось кілька ключових аспектів, про які варто знати:

Відкритість: PostgreSQL є відкритою та безкоштовною базою даних, доступною під ліцензією PostgreSQL. Це означає, що ви можете використовувати, змінювати та розповсюджувати її без обмежень.

Розширені можливості: PostgreSQL підтримує багато просунутих функцій, таких як транзакції, зовнішні ключі, операції JOIN, підзапити, реплікацію, відновлення після збоїв, тригери, операції пов'язані з геоданими та багато інших. Вона також надає можливість розробляти власні розширення та функції.

Підтримка SQL: PostgreSQL повністю підтримує мову SQL, включаючи стандарти SQL-92, SQL-99, SQL:2003 та багато інших. Вона також надає можливість використовувати розширення SQL для роботи з даними.

Розширені типи даних: Крім стандартних типів даних, таких як цілі числа, рядки та дата/час, PostgreSQL підтримує розширені типи, такі як географічні дані, JSON, XML, масиви та бінарні дані.

Висока продуктивність: PostgreSQL має потужну оптимізацію запитів та підтримує індекси, які допомагають прискорити виконання запитів до бази даних. Вона також надає можливість горизонтального та вертикального масштабування для розширення продуктивності системи.

Реплікація та відновлення: PostgreSQL підтримує реплікацію даних для створення резервних копій та забезпечення високої доступності. Вона також має можливості відновлення після збоїв, що дозволяє відновлювати базу даних до попереднього стану у разі необхідності.

Безпека: PostgreSQL надає різні механізми безпеки, такі як

автентифікація, авторизація, шифрування даних та аудит. Вона також підтримує ролі та права доступу для контролю доступу до бази даних.

Сумісність: PostgreSQL є сумісним з іншими базами даних та має можливість імпорту та експорту даних з інших форматів.

PostgreSQL має можливість зберігати файли безпосередньо у базі даних. Ця функціональність називається "Large Object" або "LOB" і дозволяє зберігати та керувати великими об'єктами, такими як файли, у базі даних.

LOB у PostgreSQL працює на основі спеціального типу даних LOB, який дозволяє зберігати бінарні дані різних розмірів. Ви можете зберігати файли будь-якого типу, включаючи зображення, відео, аудіо, документи тощо.
Основні переваги зберігання файлів у PostgreSQL включають:
Централізоване управління: Файли зберігаються в базі даних разом з іншими даними проекту, що спрощує їх управління та забезпечує централізований доступ до них.

Транзакційна безпека: PostgreSQL забезпечує атомарні операції зберігання та отримання файлів, що означає, що зміни в базі даних, включаючи файли, можуть бути збережені або скасовані за допомогою транзакцій.
Забезпечення доступу: Ви можете встановити рівень доступу до збережених файлів, включаючи права доступу до конкретних користувачів або груп користувачів.

Однак, варто враховувати, що зберігання файлів у базі даних може призвести до зростання обсягу бази даних та зниження продуктивності під час роботи з великими файлами. У деяких випадках кращим варіантом може бути зберігання файлів у спеціалізованих файлових системах, таких як AWS S3, а зв'язок з цими файлами можна зберігати у PostgreSQL у вигляді посилань або метаданих.

Amazon S3 (Simple Storage Service) є однією з найпопулярніших послуг хмарного сховища, яку надає Amazon Web Services (AWS). Вона пропонує кілька зручностей та переваг, які сприяють ефективному зберіганню та керуванню об'єктами у хмарному середовищі.

Зручності використання AWS S3:

1. **Надійність та масштабованість:** AWS S3 забезпечує високий рівень надійності та масштабованості. Ваші дані реплікуються автоматично на різних серверах та місцезнаходженнях, що забезпечує високу доступність та захист від втрати даних. Ви можете зберігати великі об'єми даних без необхідності стежити за фізичними ресурсами.
2. **Простота використання:** AWS S3 має простий інтерфейс, що спрощує роботу зі сховищем об'єктів. Ви можете легко створювати, оновлювати та видаляти файли, організовувати їх введенням відповідних віджетів та використовувати команди API для взаємодії з S3.
3. **Гнучкість та налаштування:** AWS S3 надає багато налаштувань, які дозволяють вам керувати доступом до об'єктів, налаштувати політику життєвого циклу, здійснювати реплікацію даних, використовувати шифрування та багато іншого. Ви можете налаштувати права доступу до об'єктів на основі ролей, груп користувачів або індивідуальних користувачів.
4. **Швидкодія:** AWS S3 [14] має високу швидкодію передачі даних, що дозволяє ефективно завантажувати та викачувати файли. Вона також підтримує кешування та розподілення об'єктів для поліпшення продуктивності.
5. **Інтеграція з іншими сервісами AWS:** AWS S3 має широкі можливості інтеграції з іншими сервісами AWS, такими як Amazon EC2, Lambda, CloudFront та іншими. Це дозволяє легко інтегрувати збережені об'єкти з іншими сервісами та розширювати функціональність вашої інфраструктури.

AWS S3 є потужним та надійним сервісом зберігання об'єктів, який пропонує багато переваг для зберігання, керування та розповсюдження вашого контенту. Він є популярним вибором для багатьох підприємств та розробників, які працюють у хмарному середовищі AWS.

2.6 Використання ШІ в обробці документів

Використання штучного інтелекту (ШІ) в обробці документів є однією з найбільш актуальних та перспективних областей досліджень у сучасному інформаційному суспільстві. ШІ відіграє важливу роль у поліпшенні ефективності та точності обробки документів, що сприяє автоматизації багатьох бізнес-процесів та покращує робочий потік інформації.

Однією з ключових переваг використання ШІ в обробці документів є здатність автоматичного розпізнавання та аналізу текстової інформації. Системи ШІ можуть розпізнавати рукописний текст, оптичний текст зі сканів, електронний текст з різних форматів файлів та навіть текст, витягнутий зі зображень. Це дозволяє швидко та точно отримувати необхідні дані з різних джерел без необхідності ручного введення.

Іншою важливою функцією ШІ в обробці документів є автоматичне класифікування та індексування документів. За допомогою алгоритмів машинного навчання, системи ШІ можуть автоматично визначати типи документів, їх категорії та важливість. Це дозволяє швидко знаходити необхідну інформацію, організовувати документи за різними параметрами та сприяє ефективній роботі з великим обсягом документації.

Додатково, ШІ може виконувати автоматичну обробку та аналіз вмісту документів. Використовуючи природну мову обробки (NLP) та інші техніки, системи ШІ можуть автоматично виділяти ключові фрази, розуміти смислові зв'язки між різними елементами тексту, виявляти емоції та настрої, виконувати автоматичну переклад, аналізувати синтаксичні та семантичні залежності та багато іншого. Це відкриває широкі можливості для автоматизації різних завдань, які пов'язані з обробкою текстової інформації, таких як автоматична генерація звітів, класифікація електронних листів, аналіз соціальних мереж та багато іншого.

В моїй роботі доступна функціональність логіну та реєстрації (див.

Рисунок 2.3 та Рисунок 2.4), що дозволяє користувачам створювати облікові записи та увійти на сайт зі своїми обліковими даними. Це надає користувачам персоналізований доступ до функцій та можливостей вашого сайту.

Функціонал логіну та реєстрації може бути реалізований за допомогою різних підходів та технологій, залежно від ваших потреб та вибраного стеку технологій.

```
export const authOptions: NextAuthOptions = {
 callbacks: {
 session: ({ session, user }) => ({
 ...session,
 user: {
 ...session.user,
 id: user.id,
 },
 }),
 },
 adapter: PrismaAdapter(prisma),
 providers: [
 GoogleProvider({
 clientId: process.env.GOOGLE_CLIENT_ID,
 clientSecret: process.env.GOOGLE_CLIENT_SECRET,
 }),
 ],
};
```

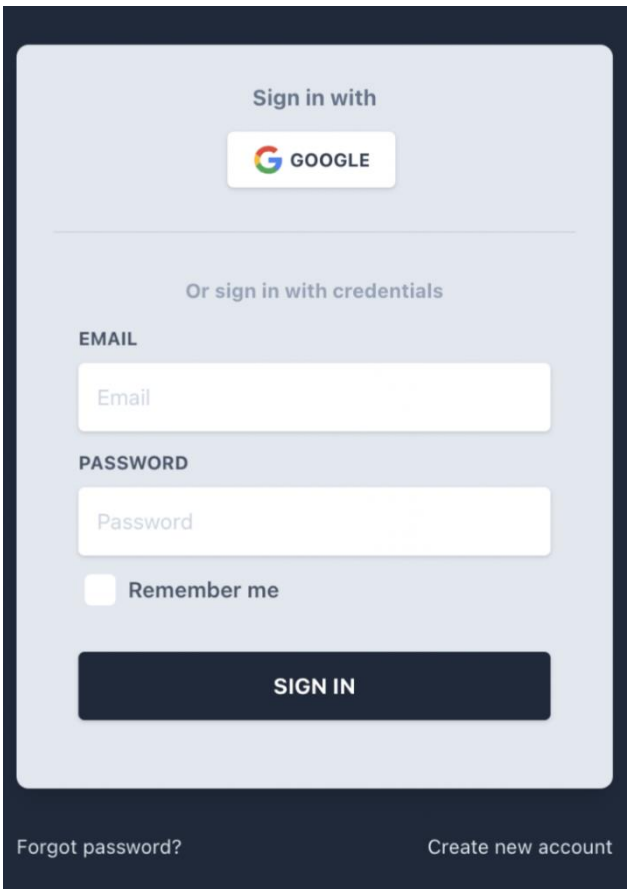

Рисунок 2.3 Логін на сайт

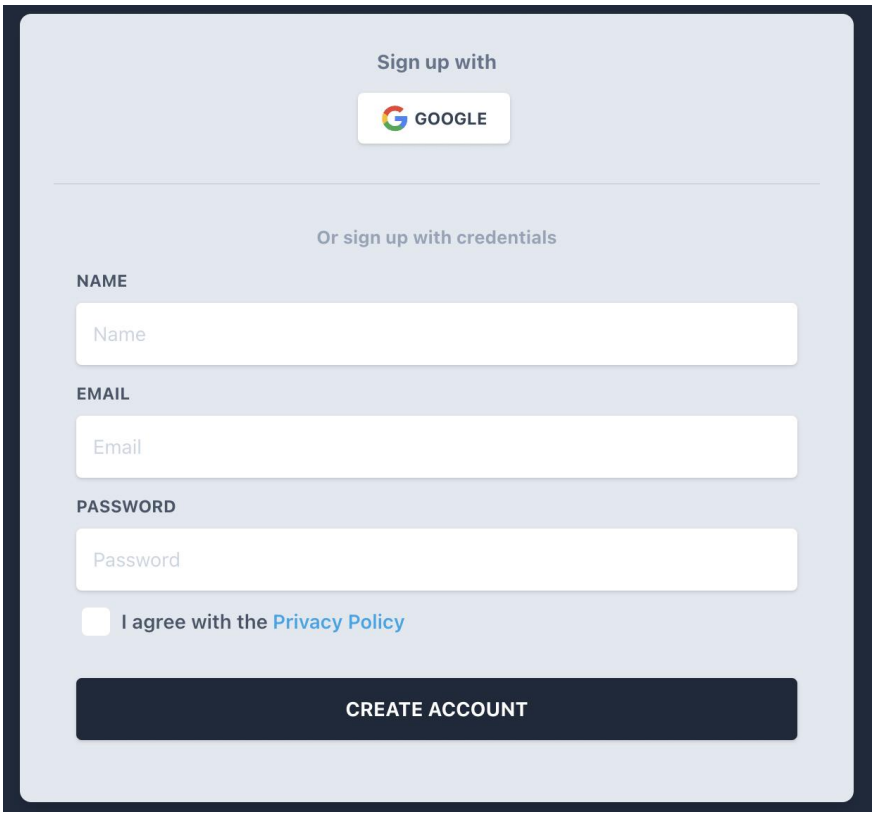

Рисунок 2.4 Реєстрація на веб-інтерфейс

Головна сторінка сайту (див. Рисунок 2.5) може створювати новий захід, додавати список друзів, між якими будуть розподілятися кошти, а також назначити статусу виконання цієї події. Ключових функціоналів:

1. **Форма створення нового заходу:** На головній сторінці можна створити нову подію, в якій користувач може ввести назву заходу, список друзів, та інші деталі. Ця форма дозволить створити новий захід.
2. **Додавання списку друзів:** Після створення заходу користувач може додати список друзів, між якими в подальшому будуть розподілятися кошти. Це може бути реалізовано шляхом введення імен друзів.
3. **Призначення статусу виконання:** Після створення заходу та розподілу коштів, користувач може назначити статус виконання для контролю та відстеження прогресу. Наприклад, ви можете мати різні статуси, такі як "планується", "в процесі", "виконано". Це дозволить учасникам знати, на якому етапі знаходиться захід.

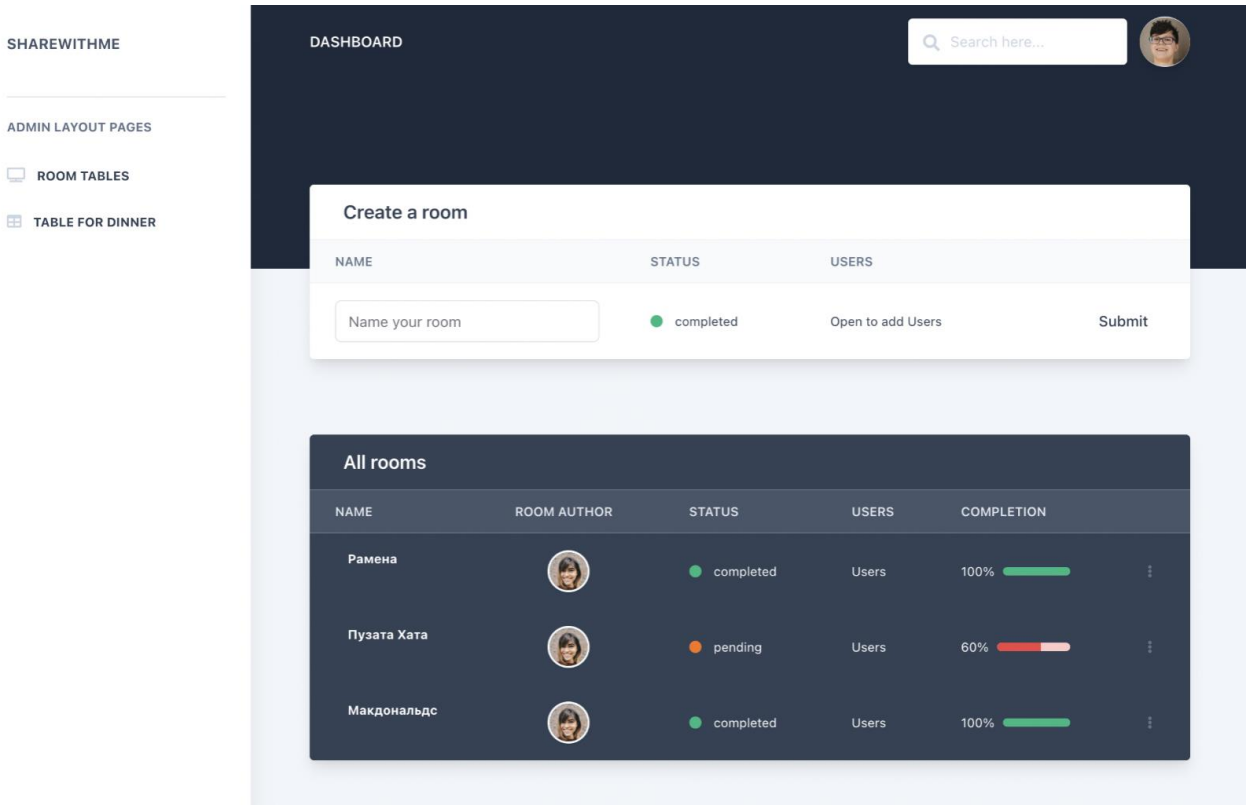

Рисунок 2.5 Головна сторінка сайту

Після додавання заходу та списку друзів, користувач може відкрити окремий захід (див. Рисунок 2.6), який він створив, щоб зробити фото (див. Рисунок 2.7), та відправити його в обробку ШІ, щоб потім розподілити кошти між учасниками.

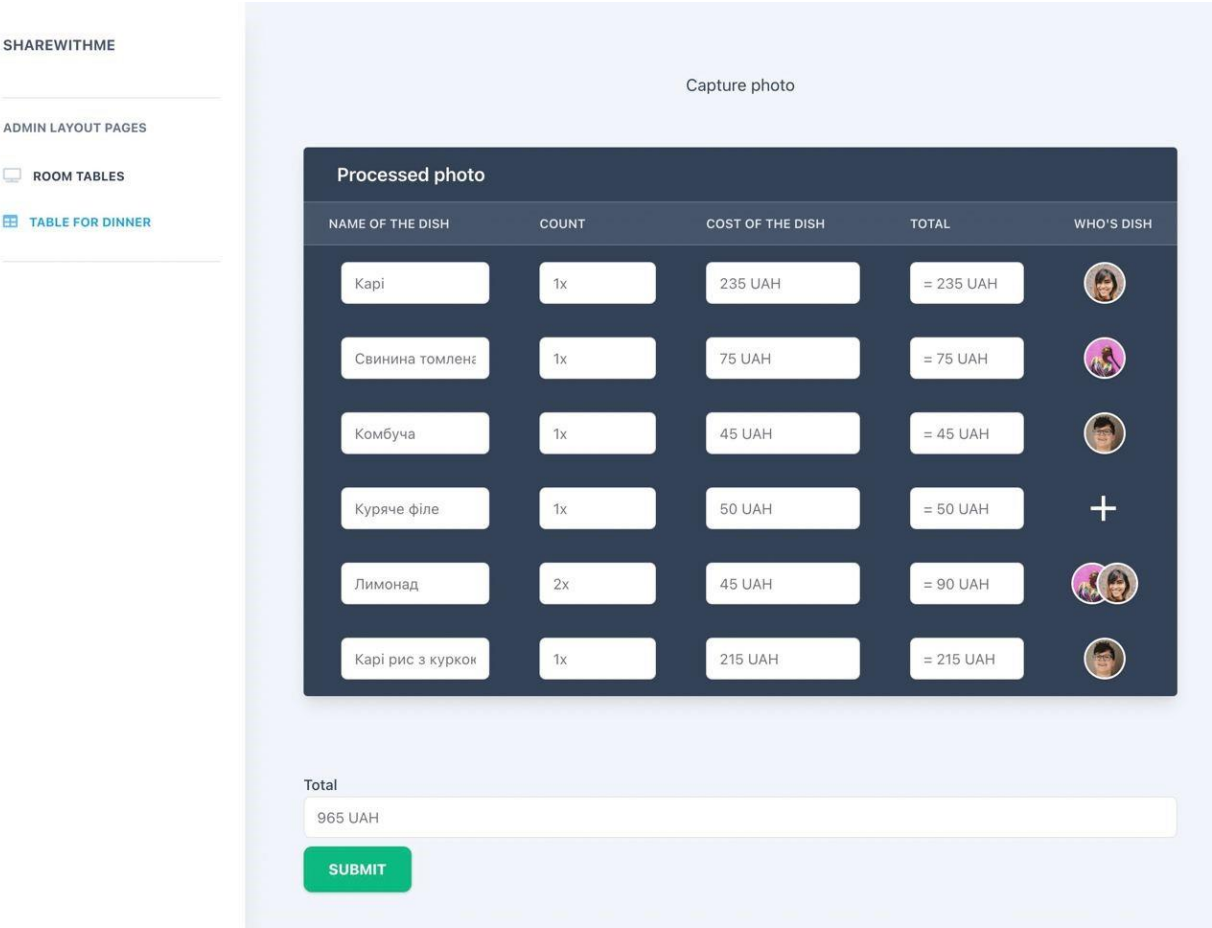

Рисунок 2.6 Сторінка окремого заходу

Рисунок 2.7 Створення фотографії чеку за допомогою камери телефону

Після створення фотографії за допомогою мобільного телефону, фото може бути відправлене на спеціальний API для обробки за допомогою штучного інтелекту.

Ми будемо використовувати онлайн-сервіс FormExtractorAI.com , який використовує штучний інтелект для автоматичного видобування даних з PDF-форм та зображень. Цей сервіс дозволяє швидко та ефективно отримувати структуровані дані з різних типів форматів, таких як рахунки, квитанції, договори, анкети та багато інших.

```
var fs = require("fs");
var request = require("request");
var options = {
method: "POST",
url: "https://worker.formextractorai.com/extract",
headers: {
"X-WORKER-TOKEN": "ACCESS_TOKEN",
"X-WORKER-FORM-ID": "FORM_ID",
"Content-Type": "image/jpeg",
"X-WORKER-ENCODING": "base64", },
 body: base64_encode("FILE_PATH_TO_IMAGE"), };
request(options, function (error, response) { if (error) throw new Error(error);
console.log(response.body); }); // function to encode file data to base64 encoded
string function base64_encode(file) { // read binary data var bitmap =
fs.readFileSync(file); // convert binary data to base64 encoded string return new
Buffer(bitmap).toString("base64"); }
```

Після цього, модель штучний інтелекту за допомогою технології OCR витягне текст для розпізнавання символів та перетворить його у зручний вигляд, для подальшого використання та обробки даних.

Приклад інформації, яку зчитує модель з чеку:

```
{
  "amount": 75,
  "discount": "",
  "name": "Свинина томлена",
  "quantity": 1,
  "sku": "",
  "unit_price": 75
```

}
{
"amount": 45,
"discount": "",
"name": "Комбуча",
"quantity": 1,
"sku": "",
"unit_price": 45
}

На остаточному етапі, після отримання результатів від штучного інтелекту, відповідальна людина може перевірити правильність зчитування даних та фотографії. Це може включати перегляд отриманого тексту, переконання, що всі об'єкти на зображенні правильно розпізнані.

Одним з можливих підходів може бути ручна перевірка та коригування результатів, де відповідальна людина проглядає отримані дані та зображення та виправляє будь-які помилки чи неправильні співпадіння. Наприклад, вона може виправляти помилково розпізнані символи або неправильно визначені об'єкти на фотографії.

Після перевірки даних, відповідальна людина може відзначити кожну людину на фотографії з відповідною стравою, яку вона замовляла.

Рисунок 2.8 Приклад іншого чека, з якого зчитується інформація

Рисунок 2.9 Приклад чека, який не працює і потребує до навчання моделі

ВИСНОВОК

Штучний інтелект - це потужний інструмент, який відкриває безліч можливостей у різних сферах життя. Проте, перед його використанням слід ретельно зважити наслідки і проблеми, які він може викликати. Потрібно бути усвідомленим та відповідальним щодо наслідків, які штучний інтелект може мати на суспільство, економіку та етику. Важливо уникати зловживання та недбалості у використанні штучного інтелекту, а замість цього працювати на створення етичних інтелектуальних систем, які допомагатимуть нам у покращенні якості життя та розвитку суспільства.

У рамках даного дипломного проекту було розроблено веб-інтерфейс з використанням технологій NextJS та TailwindCSS. Основна мета проекту полягала у використанні штучного інтелекту для автоматичного зчитування даних з чеків та розподілу витрачених коштів між людьми.

Для забезпечення функціональності зчитування даних з чеків була використана готова модель штучного інтелекту з веб-сайту https://formextractorai.com/. Ця модель дозволяє автоматично визначати різні елементи чеку, такі як сума, дата, назви товарів тощо. Використання цієї моделі спрощує процес введення даних і забезпечує більш швидку та точну обробку інформації з чеків.

Крім того, було реалізовано функціонал розподілу витрачених коштів між людьми. Цей функціонал використовує отримані дані з моделі зчитування чеків і дозволяє автоматично розподілити витрати між заданими користувачами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алгоритми дослідження нейронної мережі [Електронний ресурс] — Режим доступу: https://core.ac.uk/download/pdf/372700751.pdf
2. “Розробка нейронної мережі для розпізнавання рукописних графічних об’єктів” Білокін Данил 2020 [Електронний ресурс] — Режим доступу: https://ekmair.ukma.edu.ua/items/268971fb-88f7-4852-b60b-397f3a8b84d1
3. Статистика OpenAI Five [Електронний ресурс] — Режим доступу: https://twitter.com/OpenAI/status/1120421259274334209?ref_src=twsrc%5Etfw%7Ctwcamp%5Etweetembed%7Ctwterm%5E1120421259274334209%7Ctwgr%5E3ac45d1d8bc41ce803b4877882c3b45621ae1192%7Ctwcon%5Es1_c10&ref_url=https%3A%2F%2Fwww.engadget.com%2F2019-04-23-openai-five-dota-2-arena-results.html
4. DALL-E AI [Електронний ресурс] — Режим доступу: https://labs.openai.com/
5. ШТУЧНИЙ ІНТЕЛЕКТ В СИСТЕМАХ РОЗПІЗНАВАННЯ СИМВОЛІВ / К.О.Казьмірук / Збірник праць XVI Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених. “Погляд у майбутнє приладобудування”, 16-17 травня 2023 року, КПІ ім. Ігоря Сікорського, м.Київ, Україна. - С 15-18.
6. «Інформаційна система для розпізнавання людини на основі відеоспостереження та аналізу процесу ходи» Нагернюк О.І. 2020
7. Документація ReactJS [Електронний ресурс] — Режим доступу: https://react.dev/
8. Фотографія роботи Redux в ReactJS екосистемі [Електронний ресурс] — Режим доступу: https://www.jobsity.com/blog/why-and-when-you-should-use-redux
9. TailwindCSS автодоповнення [Електронний ресурс] — Режим доступу: https://stackoverflow.com/questions/73168537/how-to-get-the-same-font-family-as-in-this-photo-of-tailwindcss-and-how-to-get-t
10. Документація по VueJS [Електронний ресурс] — Режим доступу:

https://vuejs.org/guide/introduction.html

11. Документація по AngularJS [Електронний ресурс] — Режим доступу: https://angular.io/docs
12. Документація по NextJS [Електронний ресурс] — Режим доступу: https://nextjs.org/docs
13. Amazon Web Services [Електронний ресурс] — Режим доступу: https://aws.amazon.com/?nc2=h_lg
14. Amazon S3 Зберігання файлів [Електронний ресурс] — Режим доступу: https://aws.amazon.com/ru/s3/
15. Онлайн-сервіс для використання моделі штучного інтелекту [Електронний ресурс] — Режим доступу: https://formextractorai.com/