КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

ДОСЛІДЖЕННЯ АЛГОРИТМУ ГЕНЕРАЦІЇ ЗОБРАЖЕННЯ НА ОСНОВІ ТЕКСТОВОГО ОПИСУ

Виконав студент 4-го курсу
Олександр ГРАБАР _ (підпис)

Науковий керівник:
доцент, кандидат фізико-математичних наук

Віктор ВОЛОХОВ
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування
"_____" _____202_ р.,

протокол №

Завідувач кафедри
Микола НІКІТЧЕНКО _____
(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 56 сторінок, 8 рисунків, 1 таблиця, 1 додаток, 34 джерела посилань.

ГЕНЕРАЦІЯ ЗОБРАЖЕНЬ, ГЛИБИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, CLIP, DALL-E MINI, VQGAN.

Об'єкт дослідження – генерації зображень на основі текстового опису.

Предмет дослідження – використання нейронних мереж для генерації зображень на основі текстового опису.

Мета роботи полягає в аналізі нейронної мережі для генерації зображень на основі текстового опису.

Методи дослідження – огляд існуючих статей, публікацій та інших літературних джерел, підготовка даних, комп’ютерне моделювання та проєктування.

Проведено аналіз представлених технологій, архітектур та видів нейронних мереж. Було розглянуто існуючі аналоги, а також проведено їх порівняльний аналіз. Було проведено постанову завдання. Було обрано засоби для реалізації та тестування системи, а саме мову програмування Python та середовище розробки Google Colab.

Як результат, було проаналізовано та протестовано алгоритм DALL-E mini, що перетворює текстовий опис на картинки.

ЗМІСТ

С.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

BART – Bidirectional and Auto-Regressive Transformer – двонапрвлений і авторегресивний трансформатор;

BOS – Beginning of sentence – токен, що представляє початок кожного рядка навчальних даних;

CLIP – Contrastive Language-Image Pre-Training – контрастна мовно-образна підготовка;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

DNN – Deep Neural Network – глибока нейронна мережа;

GAN – Generative Adversarial Network – генеративно-змагальна мережа;

GPT – Generative Pre-trained Transformer – генеративний попередньо навчений трансформатор;

GRU – Gated Recurrent Units – керовані рекурентні блоки;

LSTM – Long-Short Term Memory – мережа з довготривалою і короткочасною пам’яттю;

NLP – Natural Language Processing – обробка природної мови;

NN – Neural Network – нейронна мережа;

RNN – Recurrent Neural Network – рекурентна нейронна мережа;

T-NLG – Turing Natural Language Generation – тьюрингова генерація природної мови;

VQGAN – Vector Quantized Generative Adversarial Network – векторна квантована генеративна змагальна мережа.

ВСТУП

У сучасному світі популяризуються тенденції діджиталізації та автоматизації всіх процесів, з’являється великий масив інформації, опрацювати який людині стає все тяжче. На допомогу прийшло використання нейронних мереж, які можуть замінювати людину у певних площинах при прийнятті рішень. Ця практика стає дедалі більш популярною, створюється безліч нових нейронних мереж, кожна з яких має свою власну задачу, наприклад, розпізнавання образів, мови і жестів, порівняння і розпізнавання даних, класифікація даних за параметрами, тощо.

Виходячи з описаної вище актуальності явища нейронних мереж, об’єктом дослідження було обрано використання нейронних мереж у сучасному світі.

У свою чергу, предметом дослідження став аналіз методів і засобів розробки нейронних мереж.

Метою роботи є аналіз нейронних мереж для генерації зображень на основі текстового опису.

Модель «текст-зображення» – це модель машинного навчання, яка приймає на вхід опис природною мовою і створює зображення, що відповідає цьому опису. Такі моделі почали розроблятися в середині 2010-х років, як результат розвитку глибоких нейронних мереж.

Моделі перетворення тексту в зображення, як правило, поєднують мовну модель, яка перетворює вхідний текст у латентне представлення, і генеративну модель зображення, яка створює зображення, обумовлене цим представленням. Найефективніші моделі, як правило, були навчені на величезних обсягах зображень та текстових даних, вилучених з Інтернету.

Методи дослідження – огляд існуючих статей, публікацій та інших літературних джерел, підготовка даних, комп’ютерне моделювання та проєктування.

Для досягнення поставленої мети слід виконати такі завдання: провести аналіз предметної області, зробити постановку завдання, описати обрані засоби розробки, провести експерименти з розглянутою моделлю, що використовує алгоритми, які генерують зображення через текст.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття штучного інтелекту

Штучний інтелект (ШІ) відноситься до створення розумних машин, які можуть виконувати завдання, здебільшого вимагаючи людського інтелекту. Він розроблений з метою навчання з досвіду, виявлення шаблонів та прийняття рішень на основі наданої інформації. Штучний інтелект може вирішувати проблеми, такі як розпізнавання образів, зображень, системи технічного зору, обробка природних мов і тексту та інші форми відтворення вхідних даних.

Словник англійської мови Oxford University Press визначає штучний інтелект як "теорію та створення комп'ютерних систем, здатних виконувати задачі, які здебільшого потребують людського інтелекту, включаючи візуальне сприйняття, розпізнавання мови, прийняття рішень і переклад між мовами" [1].

Програми, засновані на штучному інтелекті, включають у себе розширені вебпошукові системи (наприклад, Google), системи рекомендацій контенту (наприклад, YouTube, Netflix, Amazon), обробку людської мови (наприклад, Siri, Alexa), безпілотні автомобілі (наприклад, Tesla) і стратегічні ігрові системи високого рівня (наприклад, шахи та ґо).

Цікаво, що завдання, які потребують "інтелекту", часто виключаються з визначення штучного інтелекту, коли машини стають достатньо здатними їх виконувати. Така лінія мислення відома як "ефект штучного інтелекту". Наприклад, у роботі [5] зазначено, що оптичне розпізнавання символів, яке колись рахувалось елементом ШІ, тепер виключено з цього визначення і вважається непрогресивною технологією.

У роботах [6], [7] визначено, що термін "штучний інтелект" був введений в 1956 році. З того часу у штучного інтелекту були хвилі підйому, а потім хвилі падіння та повна втрата фінансування (відома як «зима штучного

інтелекту»), а потім нові підходи, успіх і відновлення фінансування. З моменту свого початку, дослідження у галузі штучного інтелекту випробували і відкинули багато різних підходів. Серед цих підходів були спроби симулювати мозок, моделювати процеси вирішення людських проблем, використовувати формальну логіку, створювати великі бази даних знань та імітувати поведінку тварин. У роботах [9], [10], [11] говориться, що у перші десятиліття 21-го століття в галузі домінувало математично-статистичне машинне навчання, і ця техніка виявилася дуже успішною, допомагаючи вирішувати багато складних проблем у промисловості та академічних колах.

Різні галузі штучного інтелекту (ШІ) фокусуються на конкретних цілях та використовують певні інструменти. Класичні області досліджень у сфері ШІ включають розумові процеси, планування, навчання, обробку природної мови, комп’ютерний зір, відчуття і можливість маніпуляції та переміщення об'єктів. Загальний інтелект, або здатність вирішувати будь-яку проблему, є одним з тривалих напрямків у цій площині.

Для розв'язання цих завдань, дослідники в області ШІ використовують та об'єднують різноманітні методи, включаючи алгоритми пошуку та математичну оптимізацію, формальну логіку, штучні нейронні мережі, а також статистичні, ймовірнісні та економічні методики.

Штучний інтелект ґрунтується теж і на знаннях з комп'ютерних наук, психології, медицини, лінгвістики, філософії та з інших наукових галузей.

У дослідженнях, зазначених у джерелах [13] та [14], було відмічено, що ця галузь була заснована на припущенні, що інтелект людини можна "точно описати настільки, що машину можна симулювати". Це породило філософські аргументи щодо розуму та етичних наслідків створення штучних істот, наділених людськими властивостями. Ці питання досліджувалися міфами, фантастикою та філософією з часів античності. Комп’ютерні вчені та філософи з того часу припустили, що штучний інтелект може стати ризиком для існування людства, якщо його раціональні можливості не будуть спрямовані на досягнення корисних цілей.

У роботах [21], [22] визначено, що деякі вчені, такі як Стівен Хокінг і Стюарт Рассел, висловлюють занепокоєння тим, що якщо передовий штучний інтелект одного разу отримає здатність перепроєктовувати себе зі зростаючою швидкістю, нестримний «вибух інтелекту» може призвести до вимирання людства. Маск характеризує штучний інтелект як «найбільшу екзистенційну загрозу для людства». Засновники OpenAI структурували його як некомерційну організацію, щоб вони могли зосередити свої дослідження на створенні позитивного довгострокового впливу на людину.

У роботах [4], [21], [23] визначено, що Маск і Альтман заявили, що вони частково мотивовані занепокоєнням щодо екзистенційного ризику від загального штучного інтелекту. OpenAI стверджує, що «важко зрозуміти, яку користь штучний інтелект людського рівня може принести суспільству», і що так само важко зрозуміти, «наскільки він може завдати шкоди суспільству, якщо його створювати або використовувати неправильно». Дослідження безпеки не можна відкладати: «через дивовижну історію штучного інтелекту важко передбачити, коли штучний інтелект на рівні людини стане доступним». У роботі [4] OpenAI заявляє, що штучний інтелект «має бути продовженням індивідуальної людської волі та духу свободи, якомога ширше і рівномірно поширений...». Співголова Сем Альтман очікує, що цей десятирічний проєкт перевершить людський інтелект.

У роботах [25], [26] Вішал Сікка, колишній генеральний директор Infosys, заявив, що «відкритість», коли зусилля «загалом дадуть результати в інтересах людства», є фундаментальною вимогою для його підтримки, і що OpenAI «відмінно співпадає з нашими давніми принципами» та їхнє «намагання робити цілеспрямовану роботу». Згідно з Кейдом Метцом з журналу Wired, існує припущення, що компанії, подібні до Amazon, можуть мати бажання використовувати програмне забезпечення та дані з відкритим кодом, щоб зрівняти умови гри з такими корпораціями, як Google і Facebook, які володіють величезними запасами приватних даних. Альтман заявляє, що компанії Y Combinator будуть ділитися своїми даними з OpenAI.

У 2019 році OpenAI стала комерційною компанією під назвою OpenAI LP, щоб отримати додаткове фінансування, залишаючись під контролем некомерційної організації під назвою OpenAI Inc у її структурі, яку OpenAI називає «з обмеженим прибутком». У роботах [28], [29] Маск поставив запитання: «Що є найкращим, що ми можемо зробити, щоб забезпечити гарне майбутнє?» Ми можемо сидіти осторонь, або ми можемо заохочувати регулятивний нагляд, або ми можемо приймати участь у належній організаційній рамці з особами, які виявляють глибоку турботу щодо прогресу у розробці штучного інтелекту таким чином, щоб він був безпечним і корисним для людства. Маск визнав, що «завжди є деякі ризики того, що, фактично намагаючись просувати (дружній) ШІ, ми можемо створити те, що нас хвилює»; тим не менш, найкращим захистом є «розширити можливості якомога більшої кількості людей мати ШІ. Якщо всі мають здібності ШІ, то не існує жодної людини чи невеликої групи осіб, які можуть мати суперздібності ШІ».

У роботі [21] протиінтуїтивна стратегія Маска й Альтмана спрямована на зменшення ризику того, що ШІ заподіє загальну шкоду, надаючи ШІ кожному, викликає суперечки серед тих, хто стурбований екзистенціальним ризиком штучного інтелекту. Філософ Нік Бостром скептично ставиться до підходу Маска: «Якщо у вас є кнопка, яка може робити погані речі світові, ви не хочете давати її всім». Під час розмови 2016 року стосовно технологічної сингулярності Альтман повідомив, що «ми не плануємо випускати весь наш вихідний код» і згадав про план «дозволити широким колам світу обирати представників для нового керівного органу». Грег Брокман висловився, що «наша мета зараз... зробити найкраще, що можна зробити. Це трішки невиразно».

У роботі [31], навпаки, початкове рішення OpenAI відмовитися від GPT-2 через бажання «помилитися через обережність» за наявності потенційного зловживання було розкритиковане прихильниками відкритості. Деліп Рао, експерт із створення тексту, заявив: «Я не думаю, що OpenAI витратив достатньо часу на те, щоб довести, що GPT-2 насправді небезпечний».

Інші критики стверджували, що відкрита публікація необхідна для повторення дослідження та для того, щоб мати можливість придумати контрзаходи.

У 2017 податковому році OpenAI витратив 7,9 мільйонів доларів США, або чверть своїх функціональних витрат, лише на хмарні обчислення. Для порівняння, загальні витрати DeepMind у 2017 році були набагато більшими і становили 442 мільйони доларів США. Влітку 2018 року для простого навчання ботів OpenAI Dota 2 потрібно було орендувати 128 000 процесорів і256 графічних процесорів у Google на кілька тижнів. Згідно з політикою OpenAI, що була прийнята у березні 2019 року, обмеження прибутку моделі дозволяє OpenAI LP легально залучати інвестиції з венчурних фондів і, крім того, надавати працівникам частки в компанії, щоб вони могли сказати: «Я збираюся відкрити ШІ, але в далекоглядній перспективі це не буде неприбутковим для нас як сім'ї» [33]. Багато провідних дослідників працюють у Google Brain, DeepMindабо Facebook, Inc, які пропонують опціони на акції, які некомерційна організація не зможе. У роботі [34] було зазначено, що у червні 2019 року OpenAI LP залучила грошові кошти в розмірі мільярд доларів від Microsoft. Цю суму OpenAI планує витратити "протягом п'яти років і, можливо, набагато швидше". Альтман заявив, що навіть мільярда доларів може виявитися недостатньо, і що лабораторії може зрештою знадобитися «більше капіталу, ніж будь-яка некомерційна організація коли-небудь збирала» для досягнення загального штучного інтелекту.

Перехід від некомерційної компанії до компанії з обмеженим прибутком скептично оцінив Орен Етціоні з некомерційної організації Allen Institute for AI, який погодився, що залучити провідних дослідників до некомерційної організації важко, але заявив: «Я не згоден з думкою, що некомерційна організація може «не конкурувати» та вказав на успішні малобюджетні проєкти OpenAI та інші. «Якби більший і краще фінансований завжди був кращим, IBM все одно була б номером один». Після переходу публічне оприлюднення винагороди топ-працівників OpenAI LP більше не вимагається законом. Некомерційна організація OpenAI Inc. є єдиним контрольним акціонером OpenAI LP. OpenAI LP, незважаючи на те, що є комерційною компанією, зберігає офіційну фідуціарну відповідальність за

некомерційний статут OpenAI Inc. Переважній частині правління OpenAI Inc не дозволяється мати фінансові частки в OpenAI LP. Крім того, членам меншості, які мають частку в OpenAI LP, заборонено голосувати через конфлікт інтересів. Деякі дослідники стверджують, що перехід OpenAI LP на комерційний статус несумісний із заявами OpenAI про «демократизацію» штучного інтелекту. Журналіст Vice News написав, що «взагалі ми ніколи не могли покластися на венчурних капіталістів для кращого людства».

Розглянемо поняття генеративної моделі. Оригінальна стаття про генеративне попереднє навчання (GPT) мовної моделі була написана Алеком Редфордом і його колегами та опублікована в препринті на вебсайті OpenAI 11 червня 2018 року. Це показало, як генеративна модель мови здатна здобувати знання про світ і обробляти довготривалі залежності шляхом попереднього навчання на різноманітному корпусі з довгими фрагментами безперервного тексту.

GPT-2, що є скороченою назвою Generative Pre-trained Transformer 2, є моделлю мови трансформатора, що працює в режимі неконтрольованого генерування тексту. Ця модель є прямим наступником GPT, і була анонсована на початку 2019 року. Початково були доступні лише обмежені демонстраційні версії GPT-2 для широкої громадськості.

Повна версія GPT-2 була випущена не зразу через хвилювання щодо можливого зловживання, включаючи програми для написання фейкових новин. Деякі експерти висловили скептицизм щодо того, що GPT-2 становить значну загрозу. Інститут штучного інтелекту Аллена відповів на GPT-2 інструментом для виявлення «нейронних фейкових новин». Інші дослідники, такі як Джеремі Ховард, попереджали про «технологію, яка повністю заповнює Твіттер, електронну пошту та мережу прозою, що звучить розумно та відповідає контексту, яка заглушить будь-яке інше мовлення та її неможливо відфільтрувати». У листопаді 2019 року OpenAI випустив повну версію мовної моделі GPT-2. Кілька вебсайтів містять інтерактивні демонстрації різних екземплярів GPT-2 та інших моделей трансформаторів. Автори GPT-2 стверджують, що мовні моделі без

нагляду є загальноприйнятими для навчання, що проілюстровано тим, що GPT-2 досягає найсучаснішої точності та заплутаності в 7 із 8 нульових завдань (тобто модель не була додатково навчена для жодного завдання - конкретні приклади введення-виведення). Корпус, на якому проходило навчання, називається WebText, містить трохи більше 8 мільйонів документів із загальним обсягом 40 ГБ тексту з URL-адрес, опублікованих у поданнях Reddit із принаймні 3 голосами «за». Це дозволяє уникнути певних проблем із кодуванням словника за допомогою лексем слів, використовуючи кодування пар байтів. Також є можливість представити будь-який рядок символів шляхом кодування як окремих символів, так і багатосимвольних токенів.

Generative Pre-trained Transformer 3 (GPT-3; стилізований GPT·3) – це алгоритм, який є рекурентною моделлю мови, що використовує глибинне навчання для генерації тексту, який схожий на той, що створений людиною. Враховуючи початковий текст як підказку, він створить текст, який продовжує підказку.

Архітектура – це мережа, яка має стандартну трансформаторну структуру (з використанням деяких інженерних налаштувань), відрізняється вражаючим розміром контексту, який становить 2048 токенів, і має 175 мільярдів параметрів (що вимагає 800 ГБ пам'яті). Методом навчання є «генеративне попереднє навчання», що означає, що він навчений передбачати, що буде наступним маркером. Модель продемонструвала ефективне навчання за кілька етапів виконання багатьох текстових завдань [7].

GPT-3, створений OpenAI, представляє собою третє покоління моделі передбачення мови в серії GPT-n. OpenAI – це лабораторія дослідження штучного інтелекту, розташована в Сан-Франциско. GPT-3 був анонсований у травні 2020 року і перебував на етапі бета-тестування до липня того ж року. Ця модель належить до загальної тенденції у сфері обробки природної мови (NLP) з попередньо натренованими мовними репрезентаціями. Якість тексту, створеного GPT-3, настільки висока, що може бути важко визначити, чи був він написаний

людиною, що має як переваги, так і ризики. 28 травня 2020 року 31 дослідник та інженер OpenAI представили оригінальну статтю про GPT-3. У своїй статті вони попередили про потенційну небезпеку GPT-3 і закликали до досліджень, щоб зменшити ризик. Австралійський філософ Девід Чалмерс охарактеризував GPT-3 як «одну з найцікавіших і важливих систем штучного інтелекту, коли-небудь створених. «Огляд за квітень 2022 року в The New York Times описав можливості GPT-3 як здатність писати оригінальну прозу з вільною мовою, еквівалентною людській.» [31].

22 вересня 2020 року корпорація Майкрософт повідомила про отримання ліцензії на першочергове використання GPT-3. Це означає, що Майкрософт має право використовувати GPT-3 виключно для своїх потреб. Проте інші особи і компанії все ще можуть отримувати результати за допомогою API.

За даними The Economist, вдосконалені алгоритми, потужні комп'ютери та зростання обсягу цифрових даних сприяли революції в галузі машинного навчання. Упродовж 2010-х років нові методи привели до "швидкого поліпшення розв'язання завдань", включаючи маніпулювання мовою. У роботі [8] визначено, що моделі програмного забезпечення навчаються навчатися, використовуючи тисячі чи мільйони прикладів у «структурі... частково заснованій на нейронній архітектурі мозку». В області обробки природної мови (NLP) одна з використовуваних архітектур – це нейронна мережа глибокого навчання, що заснована на моделі, вперше представленій у 2017 році – Transformer. Моделі GPT-n базуються на цій архітектурі нейронної мережі глибокого навчання на основі Transformer. Існує ряд систем НЛП, здатних обробляти, видобувати, організовувати, з’єднувати та порівнювати текстові дані, а також правильно відповідати на запитання.

У роботі [11] визначено, що 11 червня 2018 року, команда дослідників та інженерів OpenAI оприлюднила свою першу статтю про генеративні мовні моделі. Ці моделі є системами штучного інтелекту, які можна попередньо навчити за допомогою величезного та різноманітного корпусу тексту через набори даних у процесі, який вони назвали генеративним попереднім навчання (GP).

Автори описали, як продуктивність розуміння мови в обробці природної мови (NLP) була покращена в GPT-n за допомогою процесу "генеративного попереднього навчання мовної моделі на широкому спектрі немаркованого тексту, а потім подальшого точного налаштування на конкретні завдання". Це усунуло потребу в нагляді з боку людини та у тривалому ручному маркуванні.

У лютому 2020 року Microsoft представила свою технологію Turing Natural Language Generation (T-NLG), яка вважалася «найбільшою мовною моделлю з коли-небудь опублікованих із 17 мільярдами параметрів», завдання, які включали конспектування текстів і відповіді на запитання.

28 травня 2020 року препринт arXiv, створений групою з 31 інженера та дослідника OpenAI, описав розробку GPT-3, «сучасної мовної моделі» третього покоління. Команда збільшила потужність GPT-3 більш ніж на два порядки порівняно з її попередником, GPT-2, зробивши GPT-3 найбільшою моделлю нерозрідженої мови на сьогоднішній день. (У розрідженій моделі для багатьох її параметрів встановлено постійне значення, тому навіть якщо є більше загальних параметрів, є менш значуща інформація). Оскільки GPT-3 структурно подібний до своїх попередників, його більша точність пояснюється його підвищеною ємністю та більшою кількістю параметрів. Ємність GPT-3 у десять разів більша, ніж у Microsoft Turing NLG, наступної за величиною моделі NLP.

Шістдесят відсотків зваженого набору даних попереднього навчання для GPT-3 надходить із відфільтрованої версії Common Crawl, що складається з 410 мільярдів токенів, закодованих парами байтів. Інші джерела – 19 мільярдів токенів із WebText2, що становить 22% від зваженої загальної суми, 12 мільярдів токенів із Books1, що становить 8%, 55 мільярдів токенів із Books2, що становить 8%, і 3 мільярди токенів із Вікіпедії, що становить 3%. GPT-3 навчався на сотнях мільярдів слів, а також здатний кодувати в CSS, JSX і Python, серед іншого. Огляд 2022 року знову підкреслив, що навчання продовжує включати перегляд Вікіпедії.

1.2 Поняття нейронної мережі

Штучна нейронна мережа (ANN), що часто відома просто як нейронна мережа, є системою обчислень, яка виникла з інспірації біологічними нейронними мережами, що утворюють мозок у тварин.

Ці мережі базуються на наборі взаємопов'язаних одиниць або вузлів, відомих як штучні нейрони, що можуть імітувати нейрони біологічного мозку. Як і синапси в мозку, кожне з'єднання може передавати сигнали іншим нейронам. Штучний нейрон отримує сигнал, обробляє його, а потім поширює сигнали іншим нейронам, які з’єднані з ним.

У з’єднанні "сигналом" передається конкретне дійсне числове значення, тоді як вихідний результат кожного нейрона обчислюється за допомогою певної нелінійної функції, яка оперує сумою вхідних даних нейрона. Більшість нейронів та ребер мають ваги, які зазнають змін під час процесу навчання. Вага з'єднання може збільшувати або зменшувати силу сигналу. Нейрон активується тільки тоді, коли сукупний сигнал перевищує порогове значення.

Нейрони зазвичай згруповані у шари, кожен з яких може виконувати різноманітні перетворення на своїх входах. Сигнал просувається від першого (вхідного) шару до останнього (вихідного) шару, можливо, проходячи кілька етапів шарів.

Тема штучних нейронних мереж була розпочата Уорреном Маккаллоком і Уолтером Піттсом у 1943 році. В кінці 1940-х років Д. О. Хебб висунув гіпотезу навчання, яка базувалася на механізмі нейропластичності, відому як навчання Хебба. Вперше комп'ютер для моделювання мережі Хебба було використано Фарлі та Уеслі А. Кларком у 1954 році. Розенблат (1958) створив персептрон. В 1965 році Івахненко та Лапа представили першу багатошарову нейромережу як метод групової обробки даних. Основа безперервного зворотного поширення була отримана в контексті теорії керування Келлі у 1960 році та Брайсона у 1961 році, використовуючи принципи динамічного програмування [12].

У 1970 році Сеппо Ліннайнмаа розробив універсальний метод автоматичного диференціювання (AD) для дискретних з'єднаних мереж диференційованих функцій. У 1973 році Дрейфус використовував цей метод у своїх дослідженнях щодо адаптації контролерів. В 1975 році Вербос розробив алгоритм зворотного розповсюдження, що став важливою основою для навчання багатошарових мереж. У 1982 році Вербос застосував метод автоматичного диференціювання, розроблений Ліннайнмаом, для нейронних мереж. Проте, дослідження в області нейронних мереж було припинено після відкриття Мінським та Папертом у 1969 році того, що базові перцептрони не можуть обробляти XOR-схеми, а обчислювальна потужність комп'ютерів не дозволяла працювати з великими нейронними мережами [26].

З розвитком дуже великих інтегральних схем (VLSI) і технології CMOS у 1980-х роках обчислювальна потужність зросла, що дозволило використовувати більші штучні нейронні мережі.

У 1992 році було введено метод макс-пулінгу для поліпшення розпізнавання об'єктів незалежно від невеликого зміщення та деформацій. Шмідхубер розробив ієрархічну структуру мереж, яка була попередньо натренована неконтрольованим способом і далі доопрацьована методом зворотного поширення помилки. У 2006 році Хінтон висунув пропозицію моделювати кожен шар мережі використовуючи машину Больцмана.

Як зазначається у роботі [27] в 2012 році була створена мережа, яка здатна самостійно навчитися розпізнавати високорівневі концепції, такі як "кішка", спостерігаючи за немаркованими зображеннями. Збільшення обчислювальної потужності графічних процесорів та розподілені обчислення сприяли використанню більших нейронних мереж, зокрема для обробки зображень, що в кінцевому результаті привело до розвитку "глибокого навчання".

У 2010 році Ciresan та його колеги продемонстрували, що графічні процесори можуть ефективно використовувати багатошарові нейронні мережі, незважаючи на проблему "зникаючих градієнтів". У період з 2009 до 2012 років штучні нейронні мережі (ANN) почали отримувати перемоги в конкурсах ANN,

досягаючи результатів на рівні людини в розпізнаванні зображень та машинному навчанні. Наприклад, у 2009 році Грейвс за допомогою двонаправленої LSTM (Long Short-Term Memory) мережі переміг у трьох конкурсах з розпізнавання рукописного тексту без попереднього навчання представлених трьох мов.

Ciresan разом з колегами розробили перший розпізнавач шаблонів, такий як розпізнавання дорожніх знаків, в рамках IJCNN 2012.

Нейронні мережі навчаються через обробку набору прикладів, кожен з яких має відомі "вхідні" дані та "вихідні" дані, формуючи між ними вагові асоціації. Цей процес навчання включає вимірювання помилки, яка визначає різницю між очікуваним результатом і реальним виведенням мережі. Мережа потім коригує свої ваги залежно від величини помилки. Після достатньої кількості коригувань, вихід мережі все більше наближається до цільового результату. Цей процес відомий як контрольоване навчання.

Ці системи "навчаються" виконувати завдання, вивчаючи приклади, і нерідко не потребують конкретних інструкцій для кожного завдання. Наприклад, у випадку розпізнавання зображень вони можуть навчитися ідентифікувати зображення з котами, вивчаючи приклади зображень, які були ручним чином позначені як "з котом" або "без кота", і використовуючи ці дані для розпізнавання котів на інших зображеннях. Вони роблять це без будь-якої попередньої інформації про котів, наприклад, про їхню шерсть, вуха, хвіст або вуса. Замість цього вони автоматично генерують ідентифікаційні характеристики на основі оброблених прикладів.

1.3 Штучні нейрони як основна складова мережі

Архітектура штучних нейронних мереж базується на моделюванні біологічних нейронів і включає в себе штучні нейрони. Кожен штучний нейрон приймає вхід і генерує вихід, який може бути направлений до інших нейронів. Джерелами вхідних даних для нейрона можуть стати властивості зовнішніх даних, наприклад, графічні об'єкти або тексти, або результати роботи інших нейронів. Вихідний сигнал нейрона використовується для виконання різноманітних завдань, включаючи ідентифікацію об'єктів на зображеннях [23].

Для визначення вихідних даних нейрона, спочатку обчислюється зважена сума всіх вхідних значень, при цьому враховуються ваги кожного входу. До цього значення додається зміщення. Потім ця зважена сума, також відома як активаційна функція, проходить через зазвичай нелінійну функцію активації для визначення вихідних даних. Результатом роботи можуть бути зовнішні дані, як-от зображення або текстові документи, і вони використовуються для виконання завдань різного характеру, наприклад, визначення об'єктів на зображеннях.

1.4 Парадигми навчання нейронних мереж

Існують дві парадигми навчання нейронних мереж: навчання з учителем і навчання без учителя. У навчанні з учителем наявний готовий набір вхідних даних та відповідних міток, які модель намагається відтворити. У навчанні без учителя нейронна мережа самостійно навчається без використання заздалегідь заданих відповідей. Кожна з цих парадигм має свої особливості і застосовується для вирішення різних класів задач.

На сьогоднішній день існує велика кількість різних архітектур нейронних мереж та методів їх навчання, багато з яких запатентовано. Однак, основними підходами є навчання з учителем, яке використовує алгоритм зворотного поширення помилки, і навчання без учителя, яке використовує алгоритми, такі як алгоритми Хебба і Кохонена [26]. Ці підходи мають значний перетин з біологічною реальністю, де, наприклад, діти можуть навчатись як з вчителем, так і без нього.

Крім того, в останні роки сформувалася третя парадигма навчання, відома як навчання з підкріпленням.

Ця категорія навчання полягає в зв'язку між двома наборами даних, позначеними як X 'і Y', і має на меті встановити функціональний зв'язок f: X → Y. Тут Y виступає у ролі вчителя, який має певні бажані результати, а X представляє дані, що породжують Y дані. Це аналогічно ситуації, коли вчитель навчає учнів виконувати певне завдання.

Особливістю цієї парадигми навчання є пряме використання помилки, яка визначається як різниця між очікуваним і фактичним результатом. Параметри мережі служать вхідними даними для функції вартості, що оцінює розбіжність між поточним виведенням та цільовим.

Навчання з учителем виявляється дуже ефективним для задач, де відомий шаблон або ціль, до якої треба прагнути. Такі задачі включають класифікацію зображень, розпізнавання голосу, апроксимацію функцій і прогнозування. Важливо підкреслити, що нейронну мережу потрібно навчити за допомогою пар вхідних даних (X) і відповідних виходів (Y). Наявність виходів (Y) є обов'язковою умовою для навчання з учителем. Навчання без учителя, напротивагу, оперує з даними без попередньо визначених міток чи категорій. В такому контексті, нейронна мережа намагається виявити патерни та отримати знання, використовуючи лише вхідні дані X. Це подібно до самоосвіти, коли індивід набуває знання на основі власного досвіду, формуючи власні критерії. У такій ситуації ми не надаємо специфікації результатів для кожного індивідуального спостереження, але нейронна мережа здатна навчатися самостійно.

У цьому контексті вартісна функція відіграє ключову роль, впливаючи на вихідні значення нейронів, а також на взаємозв'язок між вхідними значеннями.

Завдання, в яких використовується навчання без учителя, включають кластеризацію, статистичне моделювання, компресію даних та мовні моделі.

1.5 Модель нейронної мережі

Модель нейронної мережі включає в себе її структурні характеристики та алгоритми навчання, що використовуються. Архітектура мережі обумовлює основні засади її організації (наприклад, мережі з плоскими шарами, повнозв'язні, слабозв'язні, прямого поширення, рекурентні тощо).

Конфігурація нейронної мережі уточнює її структуру в межах обраної архітектури: кількість нейронів, кількість входів та виходів, вибрані активаційні функції.

Розглянемо декілька ключових архітектур нейронних мереж [1], [2].

1. Мережі прямого поширення – це мережі, де всі зв'язки направлені від вхідних до вихідних нейронів, як у персептроні Розенблатта та багатошаровому персептроні.
2. Рекурентні мережі – в них сигнал з вихідних або прихованих нейронів частково відправляється назад на вхідні нейрони мережі.
3. Мережі на основі радіальних базисних функцій – це мережі з одним прихованим шаром, в яких активаційна функція нейронів є радіально-симетричною. Вони використовуються для класифікації та прогнозування.
4. Мережі Кохонена – це тип мереж, що використовують навчання без учителя для розв'язання задач кластеризації. Вони складаються лише з двох шарів: вхідного та вихідного.
5. Карти Кохонена або самоорганізовані мапи – варіант мереж Кохонена, де кількість вихідних нейронів значно перевищує кількість формуємих кластерів. Використовуються для візуалізації результатів кластеризації багатовимірних даних.
6. Повнозв'язні мережі – це мережі, в яких кожен нейрон з'єднаний з усіма іншими нейронами, що забезпечує максимальну щільність зв'язків.
7. Слабозв'язні мережі – мережі, у яких нейрони з'єднані лише з найближчими сусідами.
8. Нейронні мережі з плоскими шарами – це мережі, в яких нейрони формують каскади, названі шарами, причому нейрони кожного шару з'єднані з усіма нейронами наступного та попереднього шарів, а в межах шару зв'язки відсутні.

Кожна архітектура придатна для розв’язання деякого набору задач аналізу даних (класифікація, регресія, кластеризація, прогнозування) і використовує відповідні алгоритми навчання.

1.6 Використання нейронних мереж

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) імітують функціонування зорового кортексу мозку, втілюючи елементи концептуального мислення. Вони ефективно справляються з завданнями розпізнавання образів, при цьому їх обчислювальні операції можуть бути ефективно розподілені на графічні процесори, що дозволяє розробляти економічно обґрунтовані платформи зі штучним інтелектом. CNN застосовуються в системах комп'ютерного зору для самокерованих авто, комерційних безпілотників, робототехніки, а також у системах відеоспостереження [20]. Кожного разу, коли ви розблоковуєте свій смартфон за допомогою технології ідентифікації обличчя, ви використовуєте CNN.

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) мають властивість короткочасної пам'яті, що дозволяє їм аналізувати послідовності даних будь-якої довжини. RNN розбивають потік даних на елементи та встановлюють зв'язки між ними. Ці алгоритми найчастіше використовуються в задачах розпізнавання рукописного тексту та голосу. Коли ви використовуєте Shazam для пошуку пісні, говорите з Siri, Google Now або коли ви залишаєте голосові замітки для Cortana – у всіх цих випадках роботу виконують рекурентні нейронні мережі.

Мережі LSTM (Long Short-Term Memory networks) представляють собою еволюцію RNN. Вони ефективно використовуються для прогнозування коливань різних параметрів (наприклад, фондових індексів або попиту на продукти) через екстраполяцію. Також вони слугують для глибокого розуміння природної мови. До прикладу, Google застосовує LSTM в своєму персональному помічнику та системі машинного перекладу Google Translate. Без LSTM якість перекладу була б порівнянна з програмами 1990-х років, коли кількість помилок була настільки великою, що легше було самому перекласти текст, чим редагувати неточності.

В 2014 році з’явилася нова модифікація RNN що має назву керовані рекурунтні блоки (GRU). За допомогою цих блоків синтезують мову, яка звучить

як натуральна, а також має емоції. Це все дійшло до того, що люди уже не могли відрізнити розмову з ботом від живого співрозмовника, під час тестування Google Duplex і Microsoft XiaoIce. Цікавим є той факт, що Microsoft зміцнився на азіатському ринку, за допомогою компанії XiaoIce, винахід якої дозволив значно зменшити мовні проблеми.

Глибокі нейронні мережі (Deep Neural Networks, DNN) - це структури, що включають більше ніж три рівні. Вони становлять основу глибокого машинного навчання, надаючи змогу розпізнавати складні відносини в різноманітних даних. Одним із визначних застосувань їх можливостей є виявлення кореляцій між прогресуванням хвороб і потенційними ризиковими факторами в великому об’ємі наукових публікацій за допомогою IBM Watson.

Генеративно-змагальні мережі (Generative Adversarial Networks, GAN) – це пара нейронних мереж, де одна генерує варіанти, а інша оцінює наскільки вони правдоподібні. Така схема дозволяє реалізувати навчання без вчителя, що збільшує автономність системи. Наприклад, PixelDTGAN створює окремі зображення одягу, взуття та аксесуарів для каталогів інтернет-магазинів, використовуючи фотографії моделей в цьому одязі як вхідні дані. В найближчому майбутньому нейромережі допоможуть заповнювати каталоги, навіть без залучання фотографа, адже зараз зйомка для каталогів є досить витратною для електронної комерції, не можна оминути й того факту, що обробка фотографій також є часозатратною процедурою.

1.7 Генеративний дизайн

Генеративний дизайн – це ітеративний процес проєктування, який включає програму, яка генеруватиме певну кількість вихідних даних, які відповідають певним обмеженням, і дизайнера, який точно налаштовуватиме можливу область, вибираючи певний вихід або змінюючи вхідні значення, діапазони та розподіл. Дизайнеру не обов’язково бути людиною, це може бути тестова програма в тестовому середовищі або штучний інтелект, наприклад, генеративна змагальна мережа [27]. Дизайнер вчиться вдосконалювати програму (зазвичай залучаючи

алгоритми) з кожною ітерацією, оскільки їхні цілі дизайну з часом стають точнішими.

Результатом можуть бути зображення, звуки, архітектурні моделі, анімація та багато іншого. Таким чином, це швидкий метод дослідження можливостей дизайну, який використовується в різних сферах дизайну, таких як мистецтво, архітектура, комунікаційний дизайн і дизайн продукту.

Цей процес у поєднанні з потужністю цифрових комп'ютерів, які можуть досліджувати дуже велику кількість можливих перестановок рішення, дає змогу дизайнерам генерувати та тестувати абсолютно нові варіанти, що перевершують те, що може зробити людина сама, щоб отримати найбільш ефективний та оптимізований дизайн. Він імітує еволюційний підхід природи до дизайну через генетичні варіації та відбір.

Генеративний дизайн став більш важливим, в основному завдяки новим середовищам програмування або можливостям сценаріїв, які зробили відносно легким навіть для дизайнерів з невеликим досвідом програмування, реалізацію своїх ідей. Крім того, цей процес може створювати рішення для суттєво складних проблем, які в іншому випадку були б вичерпними за допомогою альтернативного підходу, що робить його більш привабливим варіантом для проблем із великим або невідомим набором рішень [20]. Це також полегшується за допомогою інструментів у комерційно доступних пакетах САПР. Стали доступнішими не лише інструменти реалізації, а й інструменти, що використовують генеративний дизайн як основу.

Генеративний дизайн в архітектурі – це ітеративний процес проєктування, який дозволяє архітекторам досліджувати ширший простір рішень із більшою можливістю та креативністю. Архітектурний дизайн довгий час вважався злою проблемою. У порівнянні з традиційним підходом допроєктування «зверху вниз», генеративний дизайн може ефективно вирішувати проблеми проєктування за допомогою парадигми «знизу вгору», яка використовує параметричні визначені правила для створення складних рішень. Саме рішення потім розвивається до гарного, якщо не оптимального рішення. Перевага використання генеративного проєктування як інструменту проєктування полягає в тому, що він не створює

фіксованих геометрій, а приймає набір правил проєктування, які можуть генерувати нескінченний набір можливих проєктних рішень. Створені дизайнерські рішення можуть бути більш чутливими, чуйними та адаптованими до проблеми.

У роботі [10] визначено, що генеративний дизайн передбачає визначення правил і аналіз результатів, які інтегровані в процес проєктування. Визначаючи параметри та правила, генеративний підхід може забезпечити оптимізоване рішення як для структурної стабільності, так і для естетики. Можливі алгоритми проєктування включають клітинні автомати, граматику форм, генетичний алгоритм, просторовий синтаксис і нещодавно штучну нейронну мережу. Через високу складність згенерованого рішення обчислювальні інструменти на основі правил, таких як метод скінченних елементів і оптимізація топології, є кращими для оцінки та оптимізації згенерованого рішення. Ітеративний процес, який забезпечується комп'ютерним програмним забезпеченням, уможливлює підхід під час проєктування методом проб і помилок і передбачає втручання архітекторів у процес оптимізації.

У роботі [12] визначено, що історичні прецеденти включають в себе Собор Святого Сімейства Антоні Гауді, який використовував геометричні форми на основі правил для структур, і Монреальську біосферу Бакмінстера Фуллера, де розроблені правила створення окремих компонентів, а не кінцевий продукт.

Більш пізні випадки генеративного дизайну включають Foster and Partners у Великому дворі королеви Єлизавети II, де мозаїчний скляний дах був розроблений з використанням геометричної схеми для визначення ієрархічних зв'язків, а потім згенероване рішення було оптимізовано на основі геометричних і структурних вимог.

РОЗДІЛ 2 ПОСТАНОВКА ЗАВДАННЯ

2.1 Огляд аналогів

2.1.1 DALL-E

DALL-E і DALL-E 2 – це моделі машинного навчання, розроблені OpenAI для створення цифрових зображень із описів природною мовою, які називаються «підказками» [1]. DALL-E було опубліковано OpenAI у дописі в блозі в січні 2021 року, і він використовує версію GPT-3, модифіковану для створення зображень. У квітні 2022 року OpenAI анонсувала DALL-E 2, наступника, призначеного для створення більш реалістичних зображень із вищою роздільною здатністю, які «можуть поєднувати концепції, атрибути та стилі».

OpenAI не опублікував вихідний код для обох моделей, хоча результати обмеженого вибору зразків підказок доступні на вебсайті OpenAI. 20 липня 2022 року DALL-E 2 увійшов у бета-фазу із запрошеннями, надісланими 1 мільйону осіб зі списку очікування. Раніше доступ був обмежений для попередньо відібраних користувачів для попереднього перегляду дослідження через занепокоєння щодо етики та безпеки. 28 вересня 2022 року DALL-E 2 відкрили для всіх і скасували вимогу щодо списку очікування; користувачі можуть безкоштовно створити певну кількість зображень і придбати додаткові. Кілька імітацій із відкритим кодом, навчених на менших обсягах даних, було випущено іншими.

Назва програмного забезпечення виникла завдяки поєднанню імен відомого художника-сюрреаліста з Іспанії Сальвадора Далі, а також персонажа мультику від Pixar WALL-E

У роботах [1], [14], [15] визначено, що модель Generative Pre-trained Transformer (GPT) спочатку була розроблена OpenAI у 2018 році з використанням архітектури Transformer. Перша ітерація, GPT, була збільшена для створення GPT-2 у 2019 році; у 2020 році вона була знову збільшена для виробництва GPT-3

зі 175 мільярдами параметрів. Модель DALL-E є мультимодальною реалізацією GPT-3 з 12 мільярдами параметрів, яка «замінює текст на пікселі», навчена на парах текст-зображення з Інтернету. DALL-E 2 використовує 3,5 мільярда параметрів, менше, ніж його попередник.

DALL-E було розроблено та оголошено громадськості разом із CLIP (Contrastive Language-Image Pre-training). CLIP – це окрема модель, заснована на нульовому навчанні, яка була навчена на 400 мільйонах пар зображень із текстовими підписами, взятими з Інтернету. Його роль полягає в тому, щоб «розуміти та ранжувати» вихідні дані DALL-E, передбачаючи, який підпис зі списку з 32 768 підписів, випадково вибраних із набору даних (одна з яких була правильною відповіддю), найбільш підходить для зображення [19]. Ця модель використовується для фільтрації більшого початкового списку зображень, згенерованих DALL-E, щоб вибрати найбільш відповідні результати.

DALL-E 2 використовує модель дифузії, засновану на вбудованих зображеннях CLIP, які під час висновку генеруються з текстових вбудованих CLIP попередньою моделлю.

У роботі [24] визначено, що DALL-E може створювати зображення в різних стилях, включаючи фотореалістичні зображення, картини та емодзі. Він може «маніпулювати та переставляти» об’єкти на своїх зображеннях і може правильно розміщувати елементи дизайну в нових композиціях без чітких інструкцій. Том Данн, який писав для BoingBoing, зауважив, що «Наприклад, коли його просять намалювати редьку дайкон, яка сморкається, п’є латте або їде на велосипеді, DALL-E часто малює носову хустку, руки та ноги в правдоподібних місцях». DALL-E продемонстрував здатність «заповнювати прогалини», щоб виводити відповідні деталі без конкретних підказок, таких як додавання різдвяних зображень до підказок, які зазвичай асоціюються зі святкуванням, і належним чином розміщені тіні до зображень, які їх не згадують. Крім того, DALL-E демонструє широке розуміння візуальних тенденцій і тенденцій дизайну.

DALL-E може створювати зображення для різноманітних довільних описів з різних точок зору лише з рідкісними помилками. Марк Рідл, ад'юнкт-професор Технічної школи інтерактивних обчислень Джорджії, виявив, що DALL-E може поєднувати концепції (описані як ключовий елемент людської творчості) [16].

Його здібності до візуального мислення достатньо для розгадування матриць Рейвена (візуальних тестів, які часто проводять людям для вимірювання інтелекту).

Залежність DALL-E 2 від загальнодоступних наборів даних впливає на його результати та призводить до упередженості алгоритму в деяких випадках, наприклад, генерування більшої кількості чоловіків, ніж жінок у запитах, у яких не згадується стать [12]. Навчальні дані DALL-E 2 були відфільтровані, щоб видалити образи насильства та сексуального характеру, але виявилося, що це посилює упередженість у деяких випадках, наприклад, зменшує частоту генерування жінок. OpenAI припускає, що це може бути тому, що жінки більш схильні до сексуалізації в тренувальних даних, через що фільтр впливає на результати. У вересні 2022 року OpenAI підтвердив The Verge, що DALL-E непомітно вставляє фрази в підказки користувача, щоб усунути упередженість результатів; наприклад, «чорношкірий чоловік» і «азіатська жінка»вставляються в підказки, в яких не вказується стать чи раса.

Занепокоєння щодо DALL-E 2 та подібних моделей створення зображень полягає в тому, що вони застосовуються для розповсюдження глибоких фейків та інших форм дезінформації [20]. Щоб пом'якшити це, програмне забезпечення відхиляє підказки за участю публічних діячів і завантаження, що містять людські обличчя. Підказки з потенційно неприйнятним вмістом блокуються, а завантажені зображення аналізуються для виявлення образливого матеріалу. Недоліком фільтрації на основі підказок є те, що її легко обійти, використовуючи альтернативні фрази, які призводять до подібного результату. Наприклад, слова «кетчуп» та «червона рідина» не фільтруються, в той час як «кров» фільтрується.

Інше занепокоєння щодо DALLE-2 і подібних моделей полягає в тому, що вони можуть спричинити технологічне безробіття для художників, фотографів і графічних дизайнерів через свою точність і популярність.

Розуміння мови DALL-E 2 має обмеження. Іноді неможливо відрізнити «Жовту книгу та червону вазу» від «Червоної книги та жовтої вази» або «Панда робить латте-арт» від «Латте-арт панди». Він генерує зображення «астронавта верхи на коні», коли йому надається підказка «кінь верхи на астронавті». Він також не може створити правильні зображення за різних обставин. Запит більше ніж на 3 об'єкти, заперечення, числа та зв'язані речення можуть призвести до помилок, а ознаки об'єктів можуть з'явитися на неправильному об'єкті. Додаткові обмеження включають обробку тексту – який, навіть з розбірливими літерами, майже завжди виглядає як тарабарщина в стилі мрії – і його обмежену здатність звертатися до наукової інформації, такої як астрономія або медичні зображення [13].

Більшість охоплення DALL-E зосереджено на невеликій підмножині «сюрреалістичних» або «химерних» результатів. Продукція DALL-E для «ілюстрації дитинчати редиски дайкон у пачці, що вигулює собаку» згадувалася у матеріалах Input NBC Nature та інших публікаціях. Його продукція для «крісла у формі авокадо» також була широко висвітлена.

ExtremeTech стверджує, що «ви можете запитати у DALL-E зображення телефону чи пилососа за певний період часу, і він зрозуміє, як ці об'єкти змінилися». Engadget також зазначив його незвичайну здатність «розуміти, як телефони та інші об'єкти змінюються з часом».

Згідно з MIT Technology Review, однією з цілей OpenAI було «надати мовним моделям кращого розуміння повсякденних концепцій, які люди використовують, щоб зрозуміти речі» [11].

Було кілька спроб створення реалізацій DALL-E з відкритим кодом. Випущений у 2022 році на платформі Hugging Face's Spaces, Craiyon – це модель штучного інтелекту, заснована на оригінальній DALL-E, яка була навчена на нефільтрованих даних з Інтернету. Він привернув значну увагу ЗМІ в середині 2022 року після випуску завдяки своїй здатності створювати гумористичні образи.

Stable Diffusion – це доступна модель із вихідним кодом, подібна до DALL-E, яка була опублікована в серпні 2022 року.

2.1.2 Midjourney

Midjourney – це незалежна дослідницька лабораторія, яка розробляє власну програму штучного інтелекту, яка створює зображення з текстових описів, подібно до DALL-E OpenAI і Stable Diffusion з відкритим кодом. Інструмент наразі перебуває у відкритому бета-тестуванні, яке було запущено 11 липня 2022 року. Команду Midjourney очолює Девід Хольц, який є співзасновником Leap Motion [31]. Midjourney використовує бізнес-модель freemium з обмеженим безкоштовним рівнем і платними рівнями, які пропонують швидший доступ, більшу ємність і додаткові функції. Хольцсказав The Register у серпні 2022 року, що компанія вже була прибутковою. Користувачі створюють ілюстрації за допомогою Midjourney за допомогою команд бота Discord.

Midjourney вперше вийшов у відкриту бета-версію 12 липня 2022 року. Midjourney наразі доступний лише через бота Discord на офіційному Discord сервері, шляхом прямого обміну повідомленнями або запрошенням бота на сторонній сервер. Щоб створити зображення, користувачі використовують команду/imagine і вводять підказку, як і інші інструменти створення мистецтва штучного інтелекту. Потім бот повертає зображення. Midjourney також працює над вебінтерфейсом.

У роботах [5], [9] зазначено, що засновник Девід Хольц бачить художників як клієнтів, а не конкурентів Midjourney; Хольц розповів The Register, що художники використовують Midjourney для швидкого створення прототипів художніх концепцій, які демонструють клієнтам перед тим, як самі починали роботу. Оскільки навчальний набір Midjourney включає роботи художників, захищених авторським правом, деякі митці звинуватили Midjourney у знеціненні оригінальної творчої роботи.

Рекламна індустрія швидко охопила такі інструменти ШІ, як Midjourney, DALL-E та Stable Diffusion тощо. Інструменти, які дозволяють рекламодавцям створювати оригінальний вміст і швидко обдумовувати ідеї, надають нові можливості, такі як «спеціальна реклама, створена для окремих осіб, новий спосіб

створення спеціальних ефектів або навіть підвищення ефективності реклами в електронній комерції», відповідно до Ad Вік.

Програма була використана британським журналом The Economist для створення першої обкладинки випуску в червні 2022 року. В Італії провідна газета Corriere della Sera опублікувала комікс, створений письменником Ванні Сантоні спільно з Midjourney у серпні 2022 року [20]. Чарлі Варцель використав Midjourney, щоб створити два зображення Алекса Джонса для інформаційного бюлетеня Варзеля в The Atlantic. Використання обкладинки, створеної штучним інтелектом, було розкритиковане людьми, які вважали, що це забирає роботу в художників. Варцель назвав свій вчинок «помилкою» у статті про своє рішення використовувати згенеровані зображення.

Зображення Midjourney під назвою Théâtre d'Opéra Spatial зайняло перше місце в конкурсі цифрового мистецтва на ярмарку штату Колорадо у 2022 році. Джейсон Аллен, який написав підказку, яка спонукала Midjourney створити зображення, надрукував зображення на полотні та взяв його на конкурс під назвою «Jason M. Allen via Midjourney». Інші цифрові художники були засмучені цією новиною. Аллен не вибачався, наполягаючи на тому, що він дотримувався правил змагань. Судді двох категорій не знали, що Midjourney використовував штучний інтелект для створення зображень, хоча вони пізніше сказали, що якби вони це знали, вони все одно присудили б Аллену головний приз.

2.2 Визначення вимог до системи

Оскільки оригінальна модель DALL-E 2 є доволі ресурсозатратною для аналізу та експериментів, було вирішено використати для розгляду алгоритм DALL-E mini, який є у 27 разів меншим, ніж оригінальний. Для розуміння поставлених вимог треба розглянути архітектуру розробленого алгоритму.

Під час навчання зображення та описи є доступними і проходять через систему у спосіб, наведений нижче.

Зображення кодуються через кодер VQGAN, який перетворює зображення в послідовність токенів.

Описи кодуються через кодер BART. Вихід BART-кодера і закодовані зображення подаються через BART-декодер, який являє собою авторегресійну модель, метою якої є передбачення наступного токена. Схема ходу тренування моделі зображена на рисунку 2.1

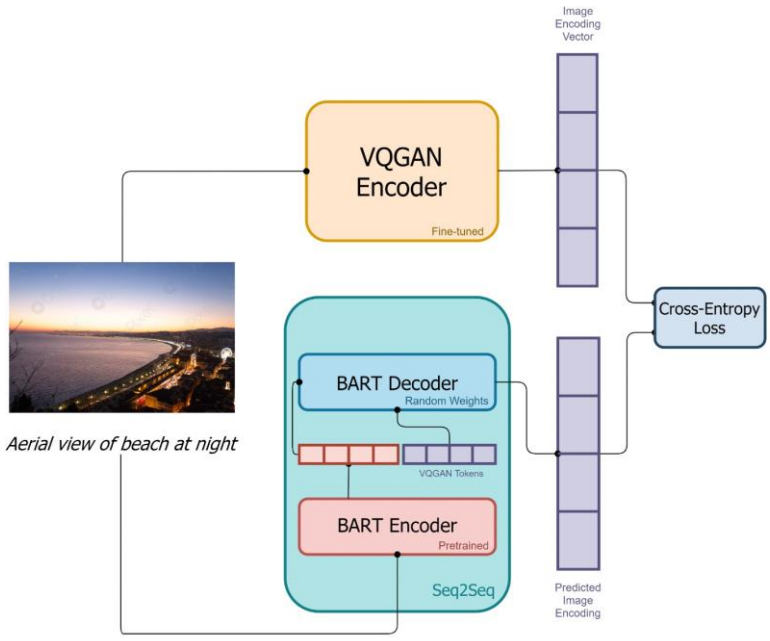

Рисунок 2.1 – Схема навчання моделі DALL-E mini

Для навчання вищезгаданого алгоритму авторами було використано три набори даних: Conceptual Captions Dataset, який містить 3 мільйони пар зображень та підписів; Conceptual 12M, який містить 12 мільйонів пар зображень та підписів; підмножина OpenAI YFCC100M, яка містить близько 15 мільйонів зображень, і яку додатково було зменшено до 2 мільйонів зображень через обмеження місця для зберігання. Приклад вибірки даних наведено на рисунку 2.2.

Рисунок 2.2 – Приклад вибірки даних

2.3 Вибір методів реалізації

2.3.1 Вибір мови програмування

Python - це високорівнева інтерпретована мова програмування зі строгою динамічною типізацією [2]. Одним із важливих аспектів Python є обов'язкове використання відступів, що сприяє зрозумілості коду. Щоб писати чистий та послідовний код для великих проектів у Python розроблені мовні конструкції, а також об’єктно- орієнтований підхід.

Дана мова має багато переваг. Досить сучасний мовний синтаксис і можливість використання різноманітних парадигм програмування (об’єктно-орієнтована, функціональна, реактивна). Python йде в пакеті «з батареями», як говорять пайтоністи, тобто з дуже великою бібліотекою стандартних пакетів. Керована пам’ять – інтерпретатор Python сам звільняє пам’ять, виділену програмі, але яка вже не використовується. Це полегшує продумування програми, скорочує код і унеможливлює допускання помилок.

Python – найпопулярніша мова програмування, яка була створена Гвідо ван Россумом наприкінці 1980-х років у Centrum Wiskunde & Informatica (CWI) у Нідерландах. Python є наступником мови програмування ABC, яка

взаємодіє з операційною системою Amoeba та має можливості обробки винятків. Перші кроки в розробці Python з’явилися в грудні 1989 року. Гвідо ван Россум був провідним розробником проєкту і відігравав роль генерального менеджера до 12 липня 2018 року [1]. В даний час він є одним з п’яти членів Наглядової ради. З січня 2019 року Бретт Кеннон, Нік Коглан, Баррі Варшав, Керол Віллінг та Гвідо ван Россум є членами комітету та активними розробниками ядра Python.

Python – це мова програмування яка застосовується в різних сферах. Вона повністю підтримує об'єктно-орієнтоване та структурне програмування, а також має функціональне та аспектно-орієнтоване програмування (у тому числі метапрограмування та метаоб'єкти). Python також підтримує розширення для дизайну контрактів та логічного програмування.

Першочерговою метою розробки мови Python була підтримка функціонального програмування на мові Lisp. У Python наявні фільтрація, скорочення та виведення, а також він розуміє словники, набори, вирази та генератори. Запозичені з Haskell та Standard ML функціональні інструменти можуть бути реалізовані за допомогою модулів itertools та functools, які знаходяться у стандартній бібліотеці[31].

Розробники Python уникають передчасної оптимізації та не займаються виправленням некритичних частин реалізації CPython. Замість цього вони зосереджуються на простоті та читабельності коду. Розширення написані на низкорівневих мовах таких як C використовуються коли швидкість виконання програми грає велику роль, також можна використовувати компілятори, такі як PyPy або Cython, останній дозволяє перетворити код Python на C та використовувати низькорівневі API.

Pythonists зазвичай називають шанувальників та користувачів мови Python, особливо тих хто більш досвідчений та має глибокі знання. Python було створено таким чином, щоб бути читабельним. Його синтаксис не перевантажений візуально, а також замість розділових знаків, які популярні в більшості мов, він використовує ключові англійські слова. У Python блоки коду відокремлюються за допомогою відступів та без використання фігурних дужок, також в ньому

дозволено використовувати крапку з комою, але це роблять доволі рідко. Ті місця де кількість відступів збільшується це початок нового блоку, це відбувається після деяких операторів, а коли відступи зменшується це показує кінець поточного блоку. Саме через це кажуть, що візуальна структура програми відповідає її семантичній структурі. Цей аспект іноді називають "зовнішнім" правилом, і хоча деякі інші мови також мають цю властивість, в більшості відступи не несуть семантичного навантаження. Загально прийнятим правилом для відступів є чотири пробіли.

Згідно з Гвідо ван Россумом, створювачем Python, дана мова не підтримує та ніколи не буду підтримувати хвостові виклики або першокласні розширення. Проте, розширення генератора Python відбулося починаючи з версії 2.5, де була додана краща підтримка функціональності. В Python 3.3 можна передавати інформацію на кілька рівнів стеку, тоді як з версії 2.5 можна було повертати дані до функції генератора [18].

Python набув широкого користування бібліотеками, такими як TensorFlow, Keras, Pytorch і Scikit-learn, що використовуються для машинного навчання, а також штучного інтелекту. Python почали активно використовувати для обробки природніх мов, завдяки його модульній архітектурі, простому синтаксису та вдосконаленим інструментам обробки тексту. Python успішно інтегровано в багато програмних продуктів, включаючи додатки до програмного забезпечення, такі як Abaqus, 3D параметричне моделювання, як FreeCAD, 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, Nuke visual effects Composer, двовимірні графічні програми, такі як GIMP, Inkscape, Scribus і Paint Shop Pro, а також програми для створення нотаток, такі як Author and Group [28]. GNU Debugger використовує Python як ефективний інструмент для розгортання складних структур, таких як контейнери C++, що дозволяє зручно відображати їх в форматі, зрозумілому для розробників.

Python є широко використовуваною мовою програмування, і його можна знайти як стандартний компонент у багатьох операційних системах. Він постачається з більшістю дистрибутивів Linux, таких як Ubuntu, Fedora, і Gentoo.

Python також використовується як компонент в інсталяторах деяких операційних систем, наприклад, Ubiquity для Ubuntu та Anaconda для Red Hat Linux і Fedora.

Завдяки широкому діапазону можливостей, які надає мова Python, він вважається універсальним і гнучким інструментом. Його потужність і простота використання роблять його вибором багатьох розробників для виконання різноманітних завдань. Тому мова Python була обрана для виконання даного завдання.

2.3.2 Вибір середовища розробки

Для експериментів з моделлю було вирішено використовувати програмне середовище Google Colab, оскільки розробники оригінальної моделі надають інструменти для тестування і аналізу саме через цей ресурс.

Google Colaboratory (Colab) - це виріб від Google Research, що надає користувачам можливість створювати та запускати код Python безпосередньо в інтернет-браузері. Він дуже зручний для застосування в машинному навчанні, обробці даних та освітніх цілях. Технічно Colab є сервісом хостингу ноутбуків Jupyter, який можна використовувати безпосередньо, без необхідності в налаштування. Важливою перевагою Colab є безкоштовний доступ до обчислювальних ресурсів, включаючи графічні процесори. Це середовище розробки надає зручність та широкі можливості для виконання проектів, а також сприяє спільноті і обміну знаннями [11].

Можливість працювати декільком користувачам одразу є однією з найбільших переваг блокноту Google Colab. Завдяки Google Colab кожен, хто має обліковий запис Google, може просто скопіювати блокнот на свій обліковий запис Google Drive. Не потрібно встановлювати жодних модулів для запуску коду, модулі попередньо встановлені в Google Colab.

Інша вагома перевага – це продуктивність. При виконанні програмного коду використовуються обчислювальні потужності серверів Google замість комп'ютера користувача. Запуск скриптів на Python часто вимагає великих обчислювальних

потужностей і може зайняти багато часу. Запускаючи скрипти в хмарному середовищі, не потрібно турбуватися про потужності власного серверу. Продуктивність локальної машини не знизиться під час виконання скриптів на Python [27].

Беручи до уваги вищеописані переваги та можливість напряму взаємодіяти з оригінальною моделлю алгоритму DALL-E через дане середовище розробки, було вирішено використовувати Google Collaboratory.

РОЗДІЛ 3 ПРОЄКТУВАННЯ ТА АНАЛІЗ АЛГОРИТМУ

3.1 Огляд архітектури алгоритму

Для реалізації алгоритму перетворення текстового опису на картинку використовується взаємодія трьох моделей, а саме DALL·E mini для кодування тексту у зображення, VQGAN для декодування зображень, а також CLIP, що збирає передбачення. Розглянемо моделі DALL-E mini, VQGAN та CLIP більш детально. VQGAN – генеративна змагальна нейронна мережа, яка добре генерує зображення, схожі на інші (але не з підказки), а CLIP – ще одна нейронна мережа, яка здатна визначати, наскільки підпис (або підказка) відповідає зображенню.

VQGAN-CLIP – це взаємодія між двома нейромережевими архітектурами, які працюють разом, щоб генерувати нові зображення з текстових підказок. Кожна з них працює разом, щоб генерувати і кваліфікувати піксельні зображення для PixRay, CLIP оцінює, наскільки добре зображення згенероване за допомогою VQGAN відповідає введеному тексту. На рисунку 3.1 схематично показано роботу алгоритму.

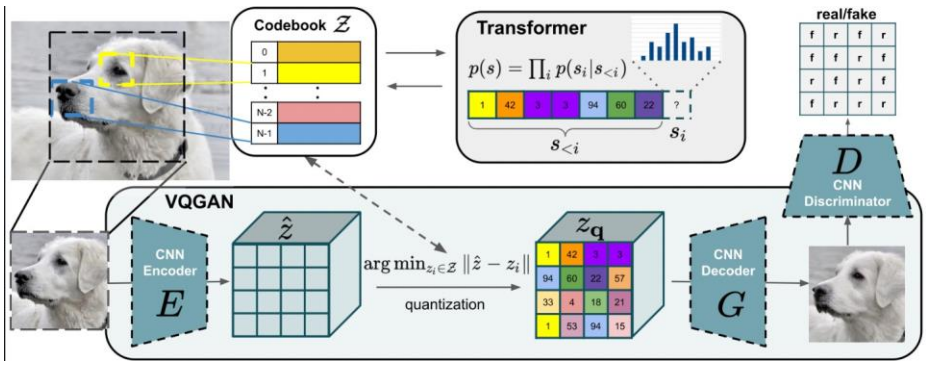

Рисунок 3.1 – Схема обробки зображень VQGAN за допомогою дискримінатора CNN

VQGAN (Vector Quantized Generative Adversarial Network) – це архітектура GAN, яка може бути використана для навчання та генерації нових зображень на основі раніше бачених даних. Вперше він був представлений у роботі «Приборкання трансформерів» Ессера, Ромбаха і Оммера. Він працює за рахунок того, що спочатку дані зображення безпосередньо вводяться в GANдля кодування карти особливостей візуальних частин зображень. Потім ці дані зображення піддаються векторному квантуванню: форма обробки сигналу, яка кодує групи векторів у кластери, доступні за допомогою репрезентативного вектора, що позначає центроїд, який називається «кодове слово» [12]. Після кодування векторно-квантовані дані записуються у вигляді словника кодових слів, також відомого як кодова книга. Вона являє собою проміжне представлення даних зображення, які потім вводяться як послідовність до трансформатора. Потім трансформатор навчається моделювати склад цих закодованих послідовностей у вигляді зображень з високою роздільною здатністю як генератор.

Основна інновація VQGAN – це можливість вводити дані зображення безпосередньо в трансформатор через послідовність кодування кодової книги авторегресійним способом [10]. На практиці трансформатор навчається на послідовності квантованих токенів, що вводяться з кодової книги в авторегресійному режимі: він вчиться передбачати розподіл наступного токена на основі послідовності попередніх токенів. Цей крок значно знижує потенційну вартість генерації таких зображень і дозволяє швидше обробляти дані зображення. На рисунку 3.2 показано схему обробки зображення алгоритмом.

Трансформатор також обмежує контекст генерації зображення «клаптиками» пікселів за допомогою ковзного вікна. Це дозволяє трансформатору враховувати лише локальний контекст патча: лише «заглядаючи» в сусідні патчі

для отримання інформації при формуванні зображення та підвищуючи ефективність використання ресурсів.

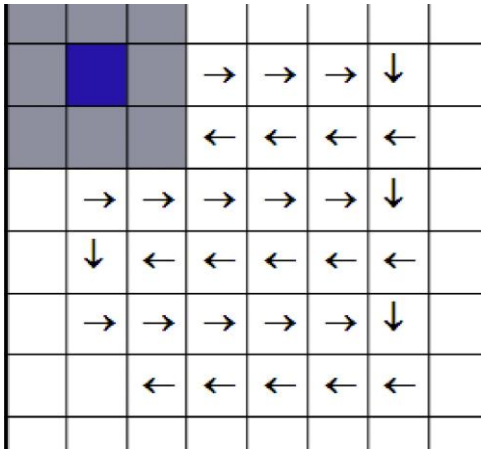

Рисунок 3.2 – Схема роботи з даними зображень у ковзному вікні

В результаті трансформатор видає зображення з високою роздільною здатністю, що генерується або безумовно, або, в міру проходження ітерацій навчання в чомусь на зразок VQGAN-CLIP, умовно, виходячи з контексту події генерації.

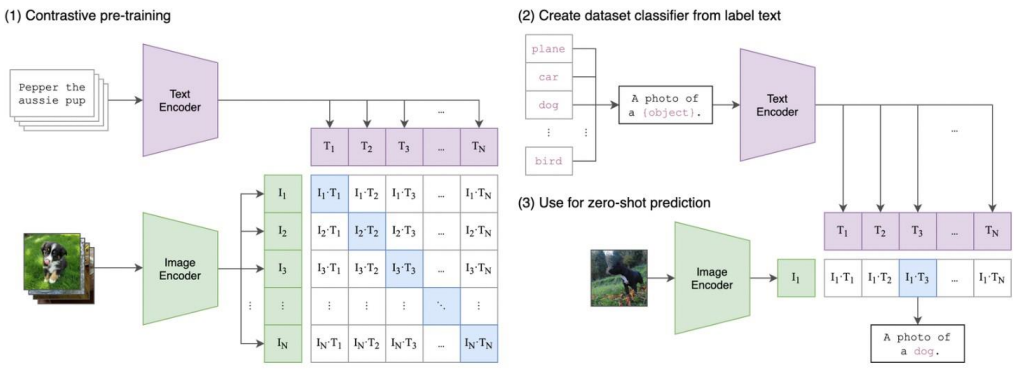

Рисунок 3.3 – Схема роботи алгоритму CLIP

CLIP (Contrastive Language-Image Pre-training) – це модель, навчена оцінювати відповідність підпису, порівняно з іншими підписами в наборі, зображенню [19]. CLIP здатна навчатися з нуля, що дозволяє їй добре працювати навіть на невідомих даних. При застосуванні в VQGAN-CLIP, CLIP може оцінювати якість згенерованих зображень у порівнянні з введеним користувачем підписом, а виведені оцінки можуть бути використані як ваги для «керівництва» навчанням VQGAN для більш точної відповідності об'єкту за допомогою рекурсивної ітерації.

CLIP навчається на нефільтрованих, дуже різноманітних і дуже зашумлених даних, і призначена для використання в режимі «нульового пострілу». З GPT-2 і 3 відомо, що моделі, навчені на таких даних, можуть досягти переконливої продуктивності без помилок; однак, такі моделі вимагають значних обчислень для навчання. Щоб зменшити необхідні обчислення, ми зосередилися на алгоритмічних способах підвищення ефективності навчання нашого підходу.

Існує два алгоритмічні рішення, які призвели до значної економії обчислень. Перший вибір – це прийняття контрастної мети для з'єднання тексту з зображеннями. Спочатку було досліджено підхід «зображення-текст», подібний до VirTex, але виникала проблема з труднощами при масштабуванні для досягнення найсучаснішої продуктивності. У малих і середніх експериментах було виявлено, що контрастний об'єктив, який використовується CLIP, в 4-10 разів ефективніший при класифікації ImageNet з нульовим знімком. Другим вибором стало застосування трансформатора зору, який дав додатковий 3-кратний приріст обчислювальної ефективності порівняно зі стандартним ResNet. Зрештою, CLIP тренувалася на 256 графічних процесорах протягом 2 тижнів, що аналогічно існуючим великомасштабним моделям зображень.

При спільному використанні VQGAN-CLIP створює серію моделей, які можуть бути використані для генерації зображень з рядка тексту. Ці зображення створюються шляхом того, що VQGAN спочатку генерує випадкове шумове зображення, яке квантується вектором і кодується в кодову книгу, кодова книга потім використовується як вхід для трансформатора, який генерує нове зображення з закодованих сигналів, цей вихід потім використовується для оцінки на скільки точним є зображення до введеного запиту через CLIP, після цього даний результат надсилається назад до VQGAN для оновлення моделі генерації зображень для більш точного відображення підказки.

Взаємодію між моделями та сам алгоритм кодування можна описати такою послідовністю кроків:

1) підпис кодується через кодер BART;
2) токен <BOS> (спеціальний токен, що ідентифікує «Початок послідовності») подається через декодер BART;
3) маркери зображень відбираються послідовно на основі прогнозованого декодером розподілу на наступному маркері;
4) послідовності маркерів зображень декодуються через декодер VQGAN;
5) CLIP використовується для відбору найкращих згенерованих зображень.

3.2 Аналіз програмної реалізації

Розробники алгоритму DALL-E mini надали доступ до коду пайплайну роботи алгоритму. Розглянемо покроково взаємодію користувача з алгоритмом. Усі роботи виконуються у програмному забезпеченні Google Collab. Спочатку

завантажуються та встановлюються необхідні бібліотеки, потім завантажуються моделі DALL·E mini, VQGAN та CLIP, а також токенайзер, що буде розбивати строки запитів. Приклад коду, що виконує даний функціонал показано у лістингу 3.1.

Лістинг 3.1 – Код завантаження моделі

```
from dalle_mini import DalleBart, DalleBartProcessor

from vqgan_jax.modeling_flax_vqgan import
VQModel from transformers import CLIPProcessor,
FlaxCLIPModelmodel, params =
DalleBart.from_pretrained(
    DALLE_MODEL, revision=DALLE_COMMIT_ID,
dtype=jnp.float16, _do_init=False
)

vqgan, vqgan_params = VQModel.from_pretrained(
    VQGAN_REPO, revision=VQGAN_COMMIT_ID,
    _do_init=False
)
```

Оскільки обчислення ведуться у хмарному середовищі, для прискорення виконання коду, функції моделі компілюються та розпаралелюються для використання переваг декількох пристроїв. Параметри моделі реплікуються на кожному пристрої для більш швидкого отримання висновків. У лістингу 3.2 зображена функція виведення результатів роботи моделі та декодування зображення.

Лістинг 3.2 – Код декодування зображення

```
@partial(jax.pmap,
axis_name="batch",
static_broadcasted_argnums=(3, 4,
5, 6))def p_generate(
    tokenized_prompt, key, params, top_k,
top_p,temperature, condition_scale
):
    return model.generate(
        **tokenized_pr
        ompt,
        prng_key=key,
        params=params,
        top_k=top_k,
```

Продовження лістингу 3.2

```
            top_p=top_p,
            temperature=temperature,
            condition_scale=condition_s
            cale,
        )
    # decode image
    @partial(jax.pmap,
    axis_name="batch")def
    p_decode(indices, params):
        return vqgan.decode_code(indices, params=params)
```

Для того, щоб приймати запити у вигляді текстових фрагментів, моделі потрібні функції обробки для завантаження даних. Для цієї мети було використано наданий DalleBartProcessor. Він може бути створений безпосередньо за допомогою встановлених пакетів, або шляхом завантаження всього пакету моделі з Weights & Biases. На рисунку 3.4 показано код завантаження такої функції.

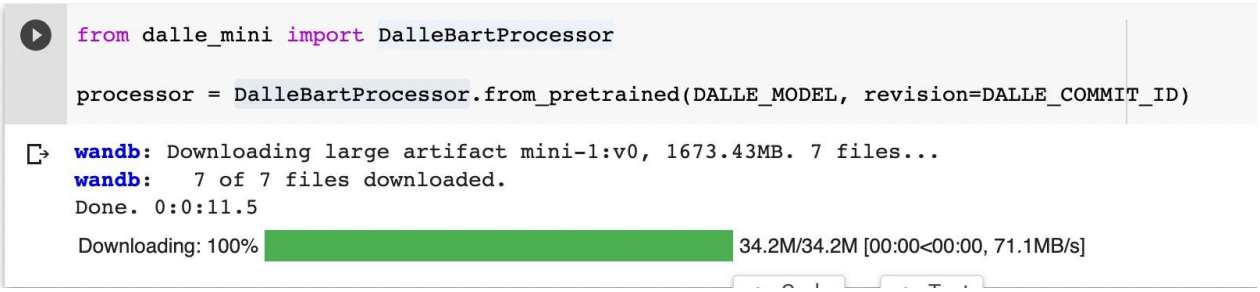

Рисунок 3.4 – Код завантаження DalleBartProcessor

Визначимо рядки запитів, що будуть використовуватися як основагенерації зображень. Оскільки кодувальник сприймає тільки англійську мову, для даного експерименту було використано наступні словосполучення:

– «sunset over a lake in the mountains»;

– «cats looking at flowers».

Потім строки дублюються на кожний пристрій у хмарі, щоб зменшити час обрахунків. Код додавання строк та їхнього розподілення наведено на рисунку 3.5.

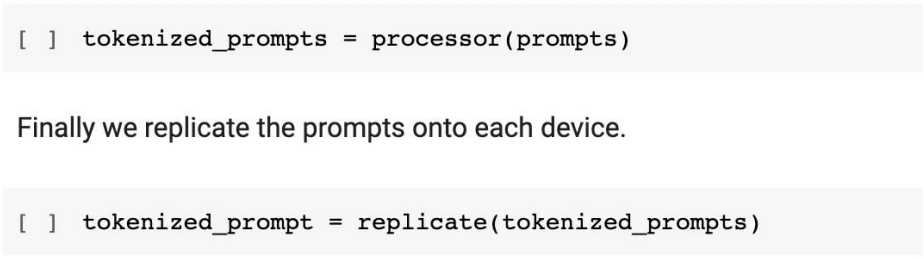

Рисунок 3.5 – Код додавання та дуплікації запитів

Наступним кроком буде генерація зображення за допомогою моделі DALLE-mini та декодування їх за допомогою VQGAN, як було описано у попередньому розділі.

Спочатку задаємо кількість передбачень – кількості картинок, що мають найбільшу вірогідність відповідності до запиту та задаємо параметри для генерування зображення. Дані параметри потрібні для визначення порогових значень, за допомогою яких визначається чи підходить згенероване зображення під заданий опис чи ні. Ініціалізація параметрів наведена на лістингу 3.3.

Лістинг 3.3 – Ініціалізація порогових значень

```
n_predictions
= 8 gen_top_k
= None
gen_top_p =
None
temperature =
None
cond_scale =
10.0
```

Наступний крок – генерація зображень. Спочатку підключаємо необхідні модулі та бібліотеки, потім створюємо список для майбутніх зображень. Наступний крок – надання унікального ключа для кожного з процесів, що проходять на кожній з віртуальних машин. Потім закодовуємо строки запиту за допомогою BOS токену, що представляє початок кожного рядка навчальних даних. Після цього прибираємо його та декодуємо зображення. Застосовуємо CLIP для визначення найбільш влучних зображень та виводимо результат. Код роботи моделі зображено у лістингу 3.4.

Лістинг 3.4 – Код виконання перетворення тексту на картинку

```
      print(f"Prompts: {prompts}\n")
      images = []
      for i in trange(max(n_predictions // jax.device_count(),
1)):
          key, subkey = jax.random.split(key)
          encoded_images = p_generate(tokenized_prompt,
   shard_prng_key(subkey),params, gen_top_k,
                gen_top_p,temperature,cond_scale,
            )
            encoded_images = encoded_images.sequences[..., 1:]
            decoded_images = p_decode(encoded_images, vqgan_params)
            decoded_images = decoded_images.clip(0.0,
   1.0).reshape((-1, 256, 256, 3))
            for decoded_img in decoded_images:
                img = Image.fromarray(np.asarray(decoded_img * 255,
   dtype=np.uint8))
                images.append
                (img)
                display(img)
                print()
```

Як можна побачити, було виведено вісім зображень (цей параметр було задано як кількість передбачень), що найбільш точно підходять під опис, заданий раніше.

Наступним кроком розглянемо результат роботи алгоритму. Варто зазначити, що оскільки алгоритм DALL-E mini натренований на меншій кількості даних, ніж алгоритм DALL-E, якість картинок буде значно нижчою

та менш натуралістичною, ніж при такому самому запиті у повномаштабному алгоритмі.

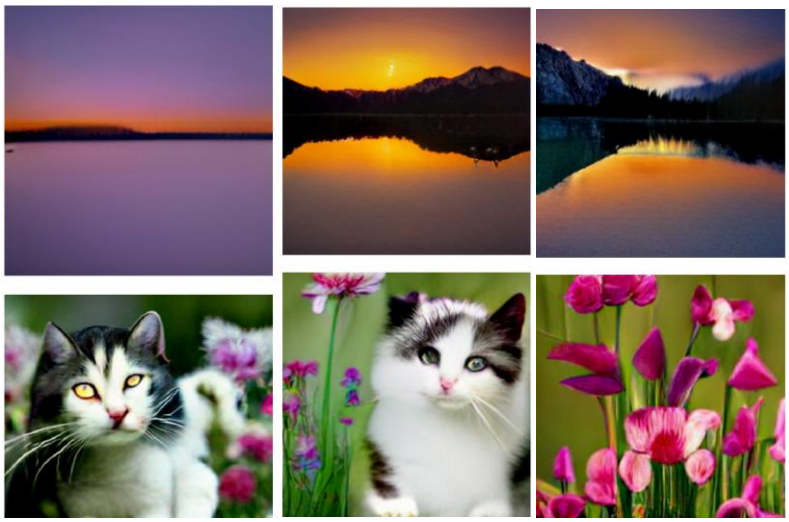

Рисунок 3.10 – Результат виконання алгоритму

Як результат, було отримано зображення, згенеровані на основі текстового опису, що був заданий раніше.

3.3 Тестування системи

Тестування програмного забезпечення є процесом перевірки та оцінювання якості програмного продукту. Це дозволяє об'єктивно і незалежно оцінювати програмне забезпечення, надаючи важливу інформацію, яка допомагає бізнесу визначити і зрозуміти потенційні ризики при впровадженні програмного забезпечення.

Тестування програмного забезпечення включає в себе різні методи, такі як [34]:

– Аналіз вимог до продукту з метою переконання, що вони є повними і правильними в контексті різних аспектів, таких як галузева перспектива, бізнес-потреби, доцільність і здатність до впровадження,

зручність для використання, продуктивність, безпека, інфраструктурні вимоги та інші.

- Запуск програми для перевірки її поведінки.
- Участь у виробничій діяльності за допомогою методів моніторингу і спостереження.
- Перевірка інфраструктури розгортання та зв'язаних автоматизованих скриптів.
- Огляд архітектури і загального дизайну продукту.
- Співпраця з розробниками для поліпшення технік кодування, проєктування, і написання тестів, що базуються на різних методах, наприклад, граничних значеннях.

Програмне забезпечення може містити помилки, які походять від помилок розробників. Ці помилки, або дефекти, присутні в вихідному коді програмного забезпечення. Якщо дефект зустрічається при виконанні програми в певних умовах, система може виконати некоректні дії, що призведе до збою.

Не всякий дефект у програмному забезпеченні неодмінно призведе до збою. Наприклад, помилки в "мертвому коді", який не виконується, ніколи не спричинять збоїв. Втім, дефект, який не викликав збій у поточному середовищі, може стати причиною проблеми при зміні обставин, як-то при запуску програми на новому обладнанні, при зміні даних на вході або при взаємодії з іншими програмами. Одна й та ж помилка може викликати широкий спектр проблем.

Не всі проблеми у програмному забезпеченні спричинені помилками в кодуванні. Через те що іноді замовник не врахував усіх деталей, то виникають нерозпізнані вимоги, які розробник може пропустити і це спричинить дорогі дефекти. Це можуть бути не тільки функціональні вимоги, але й такі нефункціональні вимоги, як можливість тестування, масштабування, відновлення після збоїв, продуктивність, безпека та інше.

Існує багато підходів до тестування програмного забезпечення. Одні методи, такі як огляди, проходження за покроковими інструкціями або перевірки,

відносять до статичного тестування. Це тестування здійснюється без виконання коду програми. Інші методи, які включають виконання коду з заданими вхідними даними, відносять до динамічного тестування.

Статичне тестування може бути неформальним, наприклад, при коректурі тексту, або формальним, коли інструменти для програмування або редактори тексту перевіряють структуру вихідного коду, а компілятори перевіряють синтаксис і потік даних. Динамічне тестування здійснюється під час виконання програми. Воно може бути проведене навіть до того, як програма буде повністю готова, щоб перевірити окремі частини коду або окремі функції чи модулі. Це може бути зроблено за допомогою спеціальних "заглушок" або "драйверів", або у середовищі налагодження.

Дослідницьке тестування є підходом до тестування програмного забезпечення, в якому тестувальник одночасно навчається, розробляє тест та виконує його. Цей підхід, який був введений Джемом Канером у 1984 році, підкреслює особисту свободу та відповідальність тестувальника за постійне удосконалення якості своєї роботи. Дослідницьке тестування розглядає навчання, дизайн, виконання та інтерпретацію результатів тесту як взаємопов'язані дії, що відбуваються паралельно протягом всього проекту.

Статичне тестування зазвичай означає перевірку коду, тоді як динамічне тестування включає його виконання.

Пасивне тестування відноситься до перевірки поведінки системи без активної взаємодії з програмним продуктом. Тестувальники аналізують системні журнали та трасування, шукають моделі та специфічну поведінку, аби приймати рішення, не надаючи власні тестові дані. Пасивне тестування також включає перевірку часу виконання в автономному режимі та аналіз журналу.

Зазвичай методики перевірки програмного обладнання поділяються на тестування білого скриньки та тестування чорного скриньки. Ці два підходи відображають перспективу тестувальника при розробці тестових сценаріїв. Можна також застосувати комбінований метод, який називається тестуванням

сірого скриньки. Тестування сірого скриньки ґрунтується на певних аспектах дизайну та об'єднує елементи тестування білого та чорного скриньок. Останніми роками тестування сірого скриньки набрало обертів і відмовилося від строгого відмежування між тестуванням білого та чорного скриньок.

Таблиця 3.1 – Тестування додатку

№	Тест-кейс	Очікуваний результат	Отриманий результат
1	Запуск без опису	Система сповіщає про відсутність опису	Система сповіщає про відсутність опису
2	Збереження без файлу	Система сповіщає про помилку читання зображення	Система сповіщає про помилку читання зображення

За результатами тестування можна зрозуміти що система готова до використання та має весь необхідний функціонал.

ВИСНОВКИ

Метою даної роботи був аналіз роботи нейронної мережі для генерації зображень на основі текстового опису.

Для досягнення визначеної цілі було виконано розгляд відповідної області знань. Було досліджено поняття нейромережі, штучного нейрона та моделі нейромережі. Проведено огляд моделей навчання нейромереж і сфер застосування нейромереж. Було також розглянуто визначення генеративного дизайну і зроблено огляд наявних аналогів.

Були оглянуті та описані обрані засоби аналізу та тестування: мову програмування Python та програмне середовище розробки Google Collaboratory.

Для аналізу було використано алгоритм з відкритим кодом DALL-e mini, який складається з кодувальника VQGAN, що перетворює зображення на послідовність токенів, з кодувальника BART, який являє собою авторегресійну модель, метою якої є передбачення наступного токена та моделі CLIP, що використовується для відбору найкраще згенерованих зображень. Дану модель було проаналізовано та проведено експерименти з її роботи.

Як результат виконання даної роботи та за допомогою чітко поставлених завдань на початку, отримано повноцінну систему, яка готова до використання в реальних умовах та має весь необхідний функціонал.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hopfield J. J. Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. U.S.A.: 1982. 800 p.

2. Neural Net or Neural Network – Gartner IT Glossary. URL: www.gartner.com .

3. Arbib B. Mind and Body: The Theories of Their Relation. New York: D. Appleton and Company: 1973. 666 p.

4. James J. The Principles of Psychology. New York: H. Holt and Company, 1990. 700 p.

5. Cuntz H. PLoS Computational Biology Issue Image | Vol. 6(8) August 2010». PLOS Computational Biology. 6 (8), 2010. 450 p.

6. Sherrington C.S. Experiments in Examination of the Peripheral Distribution of the Fibers of the Posterior Roots of Some Spinal Nerves. Proceedings of the Royal Society of London, 1998. 500 p.

7. McCulloch W. A Logical Calculus of Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics. 5 (4), 1943. 350 p.

8. Hebb, Donald. The Organization of Behavior. New York: Wiley, 1978. 250 p.

9. Farley B. Simulation of Self-Organizing Systems by Digital Computer. IRE Transactions on Information Theory, 1954. 280 p.

10. Rochester N. Tests on a cell assembly theory of the action of the brain, using a large digital computer. IRE Transactions on Information Theory, 1958. 321 p.

11. Rosenblatt F. The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain. Psychological Review, 1959, 145 p.

12. Werbos P.J. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, 1975. 200 p.

13. Minsky, M. An Introduction to Computational Geometry. MIT Press, 1969. 543 p.

14. Peter F. Cooperativity and the Transition Behavior of Large Neural Nets». Master of Science Thesis. Burlington: University of Vermont, 1981. 118 p.

15. Krizan J.E. Exact Phase Transitions for the Ising Model on the Closed Cayley Tree. Physica. North-Holland Publishing Co, 1983. 351 p.

16. Glasser M.L. The Ising model on a closed Cayley tree, Physica, 1983. 672 p.

17. Rumelhart D.E. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Cambridge: MIT Press, 1986. 128 p.

18. Russell I. Neural Networks Module. Archived from the original on May 29, 2014.

19. McCulloc W. A Logical Calculus of Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, 1943. 133 p.

20. Copeland B. The Essential Turing. Oxford University Press, 2004 p. 403.

21. Billings S. A. Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains. Wiley, 2013. 213 p.

22. Wieczorek S. (2018). Semantic Image-Based Profiling of Users' Interests with Neural Networks. Studies on the Semantic Web, 2018. p. 36.

23. Neuroscientists demonstrate how to improve communication between different regions of the brain. URL: medicalxpress.com.

24. Facilitating the propagation of spiking activity in feedforward networks by including feedback. URL: https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31.

25. Dryden Flight Research Center – News Room: News Releases: NASA NEURAL NETWORK PROJECT PASSES MILESTONE.

URL: https://medium.com/discovery-news/has-nasa-s-kepler-mission-discovered- an-alien-megastructure-a38e917ec41c.

26. Roger Bridgman's defence of neural networks. URL: https://medium.com/p/678c51b4b463.

27. Scaling Learning Algorithms towards {AI} – LISA – Publications – Aigaion 2.0. URL: www.iro.umontreal.ca .

28. Memristive switching mechanism for metal/oxide/metal nanodevices. Nat. Nanotechnol. URL: www.umontreal.ca .

29. Strukov D. B. «The missing memristor found», London, 2008, 456 p.

30. 2012 Kurzweil AI Interview with Jürgen Schmidhuber on the eight competitions won by his Deep Learning team 2009–2012. URL: https://medium.com/p/678c51b4b463.

31. Graves A. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks., 2008. p. 552.

32. Graves A.; Liwicki, M.; Fernandez, S.; Bertolami, R.; Bunke, H.; Schmidhuber, J. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009. 863 p.

33. A fast learning algorithm for deep belief nets. URL: https://medium.com/p/678c51b4b463.

34. A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position URL: https://medium.com/p/6sgy78c51b4b463.

ДОДАТОК А

Код програми «DALL·E mini – Inference pipeline»

```
# Model references

# dalle-mega
DALLE_MODEL = "dalle-mini/dalle-mini/mega-1-fp16:latest"
DALLE_COMMIT_ID = None

# DALLE_MODEL = "dalle-mini/dalle-mini/mini-1:v0"

# VQGAN model
VQGAN_REPO = "dalle-mini/vqgan_imagenet_f16_16384"
VQGAN_COMMIT_ID = "e93a26e7707683d349bf5d5c41c5b0ef69b677a9"
import jax
import jax.numpy as jnp

# check how many devices are available
jax.local_device_count()
# Load models & tokenizer
from dalle_mini import DalleBart, DalleBartProcessor
from vqgan_jax.modeling_flax_vqgan import VQModel
from transformers import CLIPProcessor, FlaxCLIPModel

# Load dalle-mini
model, params = DalleBart.from_pretrained(
    DALLE_MODEL, revision=DALLE_COMMIT_ID, dtype=jnp.float16,
_do_init=False
)

# Load VQGAN
vqgan, vqgan_params = VQModel.from_pretrained(
    VQGAN_REPO, revision=VQGAN_COMMIT_ID, _do_init=False
)
from functools import partial

# model inference
@partial(jax.pmap, axis_name="batch",
static_broadcasted_argnums=(3, 4, 5, 6))
def p_generate(
    tokenized_prompt, key, params, top_k, top_p, temperature,
condition_scale
):
    return model.generate(
        **tokenized_prompt,
        prng_key=key,
        params=params,
```

```
        top_k=top_k,
        top_p=top_p,
        temperature=temperature,
        condition_scale=condition_scale,
    )

# decode image
@partial(jax.pmap, axis_name="batch")
def p_decode(indices, params):
    return vqgan.decode_code(indices, params=params)
print(f"Prompts: {prompts}\n")
# generate images
images = []
for i in trange(max(n_predictions // jax.device_count(), 1)):
    # get a new key
    key, subkey = jax.random.split(key)
    # generate images
    encoded_images = p_generate(
        tokenized_prompt,
        shard_prng_key(subkey),
        params,
        gen_top_k,
        gen_top_p,
        temperature,
        cond_scale,
    )
    # remove BOS
    encoded_images = encoded_images.sequences[..., 1:]
    # decode images
    decoded_images = p_decode(encoded_images, vqgan_params)
    decoded_images = decoded_images.clip(0.0, 1.0).reshape((-1,
256, 256, 3))
    for decoded_img in decoded_images:
        img = Image.fromarray(np.asarray(decoded_img * 255,
dtype=np.uint8))
        images.append(img)
        display(img)
        print()
```