Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методу навчання трансформерів для задач створення зображень за текстовими даними
(тема)

Виконав:
студент 2 курсу, групи СШІм-22-1
Поліїт М.Р.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)
Керівник доц. Золотухін О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Поліїту Максиму Руслановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методу навчання трансформерів для задач створення зображень за текстовими даними

затверджена наказом університету від 1 квітня 20 24 р. № 260Ст
2. Термін подання студентом роботи до екзаменаційної комісії 12 червня 20 24 р.
3. Вихідні дані до роботи Розроблена та навчена на трансформерах нейронна мережа для створення зображень за текстовими даними

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Трансформери в задачі «text-to-image»
3) Розробка архітектури
4) Програмна реалізація

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	01.04.2024	виконано
2	Аналіз науково-технічних джерел інформації	02.04.2024 – 10.04.2024	виконано
3	Аналіз проблематики теми кваліфікаційної	11.04.2024 – 14.04.2024	виконано
	роботи		
4	Аналіз предметної галузі кваліфікаційної	15.04.2024 – 21.04.2024	виконано
	роботи		
5	Постановка задачі дослідження	22.04.2024 – 30.04.2024	виконано
6	Аналіз методів для вирішення поставленої задачі	01.05.2024 – 12.05.2024	виконано
7	Програмна реалізація	13.05.2024 – 31.06.2024	виконано
8	Оформлення пояснювальної записки	01.06.2024 – 06.06.2024	виконано
9	Проходження нормоконтролю	07.06.2024 – 09.06.2024	виконано
10	Перевірка кваліфікаційної роботи на плагіат	10.06.2024	виконано
11	Попередній захист	11.06.2024	виконано
12	Захист	12.06.2024	виконано

Дата видачі завдання 1 квітня 20 24 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Золотухін О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 64 с., 27 рис., 1 дод., 20 джерел.

АВТОКОДЕР, ГЕНЕРАТОР, ДЕКОДУВАННЯ, ДИСКРИМІНАТОР, ЕНКОДУВАННЯ, НЕЙРОННІ МЕРЕЖІ, ТЕКСТОВИЙ ОПИС, ТРАНСФОРМЕРИ, GAN, NLP, TEXT-TO-IMAGE.

Об’єкт дослідження – нейронна мережа для генерування зображень за текстовим описом.

Предмет дослідження – аналіз та обробка текстового опису за яким необхідно створити зображення.

Мета роботи – дослідження моделі нейронної мережі, яка генерує зображення за текстовим описом використовуючи трансформери.

Методи дослідження – розробка та навчання моделі нейронної мережі для генерування зображення за текстовим описом.

ABSTRACT

Master’s thesis contains: 64 pp., 27 fig., 1 ann., 20 references.

AUTOENCODER, DECODING, DISCRIMINATOR, ENCODING, GAN, GENERATOR, NEURAL NETWORKS, NLP, TEXT DESCRIPTION, TEXT-TO-IMAGE, TRANSFORMERS.

The object of research is a neural network for generating images from a text description.

The subject of the study is the analysis and processing of a text description that is used to create an image.

Purpose – to study the model of a neural network that generates images from a text description using transformers.

Research methods – development and training of a neural network model for generating images from a text description.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БЛФ – багатозмінна логістична функція;

НМ – нейронна мережа;

BERT – Bidirectional Encoder Representations from Transformers – двонаправлені кодувальні уявлення на основі трансформерів;

GPU – Graphics Processing Unit – графічний процесор;

LSTM – Long Short-Term Memory – довготривала короткочасна пам’ять;

NLP – Natural Language Processing – обробка природньої мови;

RNN – Recurrent Neural Network – рекурентна нейронна мережа;

TPU – Tensor Processing Unit – тензорний процесор.

ВСТУП

За останні десятиліття технології в світі на стільки швидко розвинулися та продовжують розвиватися, що іноді навіть важко повірити у їхні можливості. Діджиталізація світу дуже спрощує життя людям як звичайному користувачу, так і спеціалістам різних направлень. Зазвичай велику складність складає саме швидке створення якісних ілюстрацій, адже це необхідно знайти хорошого спеціаліста, витратити певний час на створення ілюстрації в залежності від її складності. Здебільшого, якщо мова не йде про вузькоспеціалізовані задачі з графічного дизайну, то ми хочемо отримати результат швидко та якісно. Такі ілюстрації можуть бути використані на сайтах, буклетах та іншій друкованій чи електронній продукції. Головну проблему складає: як пояснити комп'ютеру що саме ти хочеш побачити на згенерованій ілюстрації? Для вирішення даної задачі відіграє значну роль NLP (Natural Language Processing), адже саме за допомогою цієї технології можна виконати генерацію зображення за текстом. Моделі глибокого навчання, такі як генеративні згорткові мережі (GAN), можуть використовувати NLP для отримання контексту або інструкцій та генерації відповідних зображень.

Процес генерації зображень за описом тексту може бути розділений на кілька ключових етапів. Першим етапом є представлення текстового опису у векторній або числовій формі. Цей крок включає у себе застосування методів векторної репрезентації тексту, таких як word embeddings або трансформерні моделі, що дозволяють отримати числове представлення для кожного слова або токену у тексті.

Наступним етапом є використання цього числового представлення для генерації зображення. Це може бути досягнуто за допомогою глибоких генеративних моделей, зокрема, генеративних згорткових мереж (GAN) або автокодувальних архітектур. У випадку GAN, генератор може приймати на вхід векторні представлення тексту та генерувати відповідне зображення, в

той час як дискримінатор намагається розрізнити між справжніми та синтезованими зображеннями.

Спроби реалізації генерації зображень за описом тексту також включають в себе використання архітектур, які комбінують обробку природної мови та зображень у спільному просторі представлень. Наприклад, можливим є використання моделей, що включають у себе як генеративні, так і розпізнавальні компоненти, які працюють у спільному просторі зображень та тексту. Це дозволяє моделі здійснювати зв'язану обробку інформації і забезпечує кращу взаємодію між текстовими та візуальними аспектами даних.

Узагальнюючи, генерація зображень за описом тексту є складною задачею, яка потребує поєднання методів обробки природної мови та глибокого навчання. Вона відкриває широкі перспективи для створення нових методів генерації контенту та розвитку мультимедійних технологій. Основною метою цього підходу є створення високоякісних зображень, які точно відображають зміст текстового опису.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

Нині досить актуальна проблема автоматизованого створення зображень на основі описів природньою мовою. Головним питанням також є те, на скільки творчо комп’ютер може підійти до створення таких зображень, як багато варіантів різноманітних він може згенерувати. Враховуючи всі ці питання, автокодери з трансформерами все ще вважаються одним з найпотужніших інструментів для генерації зображень за текстовим описом.

Спроби реалізації генерації зображень за описом тексту також включають в себе використання архітектур, які комбінують обробку природної мови та зображень у спільному просторі представлень. Наприклад, можливим є використання моделей, що включають у себе як генеративні, так і розпізнавальні компоненти, які працюють у спільному просторі зображень та тексту. Це дозволяє моделі здійснювати зв'язану обробку інформації і забезпечує кращу взаємодію між текстовими та візуальними аспектами даних.

1.1 Аналіз предметної галузі

Обраний підхід генерації зображень за описом використовуючи автокодери з трансформерами дозволяє створювати візуальний контент, який буде відповідати текстовим сценаріям або описам. Такий спосіб створення зображень має дуже багатий перелік областей застосування: веб-сайти, типографія, графічний дизайн та інше.

Автокодери – це тип нейронної мережі, яка навчається відтворювати вхідні данні на виході. Такі нейронні мережі складаються з двох частин: енкодер та декодер. Енкодер приймає вхідні дані і перетворює їх в складний вектор або кодування. Декодер використовує це кодування для відтворення

вхідних даних. У випадку зображень, автокодери зазвичай здатні відтворити те саме зображення, яке було на вході описано текстом.

Трансформери – це архітектура нейронних мереж, що базується на механізмах уваги і розроблена для обробки послідовностей даних, таких як текст. Вони відомі своєю здатністю до моделювання довгих залежностей в послідовностях, що робить їх ефективними для обробки текстових даних.

Процес генерації зображень за текстовим описом можна розділити на два етапи: етап навчання та етап генерації. На етапі навчання автокодер з трансформерами навчається на навчальній вибірці, яка складається з пар «текст-зображення». Текстовий опис подається на вхід енкодеру, який перетворює його в складний вектор. Цей вектор потім передається декодеру, який відтворює відповідне зображення. Під час навчання мережі простежуються зв'язки між текстом і зображенням, щоб вона могла вчитися ефективно генерувати зображення на основі текстового опису. Після завершення навчання починається етап генерації зображення із використанням автокодеру за новими текстовими описами. екстовий опис подається на вхід енкодеру, який створює кодування. Потім декодер використовує це кодування для створення відповідного зображення.

1.2 Постановка задачі

Необхідно дослідити навчання автокодеру на трансформерах із використанням навчальної вибірки «текст-зображення» для виконання задач по створенню зображень на основі текстового опису.

1.3 NLP в задачах «text-to-image»

При вирішенні задач щодо обробки тексту з метою генерації зображень певну роль відіграє Natural Language Processing (NLP), адже саме ця технологія штучного інтелекту використовується для перетворення

текстових описів у відповідні зображення до нього. Моделі, такі як генеративні згорткові мережі (GAN) або варіанти автокодерів, можуть використовувати текстовий опис як вхід для генерації відповідного зображення.

Natural Language Processing (NLP) – це галузь штучного інтелекту, що вивчає обробку та аналіз людської мови за допомогою комп'ютерних алгоритмів та моделей. Вона займається розумінням, генерацією, перекладом та іншими аспектами обробки природної мови.

NLP має декілька напрямків досліджень, а саме: морфологічний аналіз (вивчається будова слова та його форм, включаючи відмінювання та похідні), синтаксичний аналіз (аналіз структури речень та їх компонентів, таких як підмети, присудки, об'єкти тощо), семантичний аналіз (розуміння значення тексту та його відношень), дискорсивний аналіз (вивчення структури та смислу текстів у контексті), машинний переклад (автоматичний перевод тексту з однієї мови на іншу), розпізнавання іменованих сутностей (визначення та класифікація іменованих об'єктів у тексті, таких як імена людей, місця, організації тощо), сентиментний аналіз (визначення та класифікація емоційного тону тексту, наприклад, позитивного, негативного або нейтрального), генерація тексту (створення тексту за допомогою комп'ютерних алгоритмів, що враховують синтаксичні та семантичні правила).

NLP використовує широкий спектр методів та підходів, включаючи правила, статистичні моделі та нейронні мережі. Останнім часом найбільшу популярність отримали моделі на базі нейронних мереж, зокрема трансформери, які показали значні покращення у багатьох завданнях NLP завдяки своїй здатності до моделювання довготривалих залежностей у тексті.

1.4 Автокодери

Автокодери – це тип нейронної мережі, яка навчається відтворювати вхідні данні на виході. Такі нейронні мережі складаються з двох частин: енкодер та декодер.

Енкодер приймає вхідні дані і перетворює їх в складний вектор або кодування. Декодер використовує це кодування для відтворення вхідних даних (рисунок 1.1). У випадку зображень, автокодери зазвичай здатні відтворити те саме зображення, яке було на вході описано текстом.

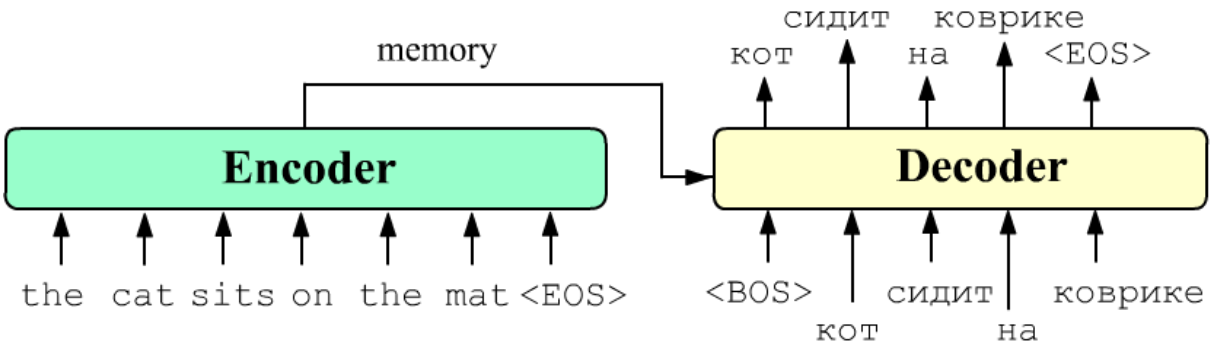

Рисунок 1.1 – Робота енкодеру та декодеру

Основна мета автокодерів – навчити модель відтворювати вхідні дані на виході, при цьому енкодер перетворює вхідні дані у складний код, а декодер відтворює ці дані з цього коду. Процес навчання автокодера зазвичай здійснюється за допомогою методу зворотного поширення помилки.

Математично, енкодер і декодер у автокодері можуть бути представлені як функції $f_{enc}(\mathbf{x})$ і $f\text{dec}(\mathbf{z})$, де $\mathbf{x}$ – вхідні дані, а $\mathbf{z}$ – отриманий код. Процес автокодування може бути описаний наступним чином:

1) енкодування ($f$enc): енкодер приймає на вхід дані $\mathbf{x}$ і перетворює їх у код $\mathbf{z}$, який представляє собою вектор у прихованому просторі: $\mathbf{z}=f\text{enc}(\mathbf{x})$;

2) декодування (*f*dec): декодер використовує отриманий код **z** для відтворення вихідних даних **x′**, які максимально наближені до вхідних даних **x: x′**=*f*dec(**z**).

Під час навчання автокодера метою є мінімізація функції втрат між вхідними та відтвореними даними. Це може бути виконано за допомогою різних функцій втрат, таких як середньоквадратична помилка (MSE) або перехресна ентропія. Оптимізація параметрів моделі зазвичай здійснюється за допомогою алгоритмів градієнтного спуску.

Застосування автокодерів може бути широким, включаючи стиснення даних, вилучення ознак, відновлення даних, анамольне навчання та генерацію контенту. Вони також можуть бути використані як попередній етап для подальших завдань, таких як класифікація або кластеризація даних. У сучасних дослідженнях і застосуваннях автокодерів зустрічаються різноманітні архітектури, включаючи згорткові, рекурентні та трансформерні структури.

1.5 Трансформери

Трансформери – це архітектура нейронних мереж, що базується на механізмах уваги і призначена для обробки послідовностей даних, таких як текст. Їх вперше представлено у роботі «Attention is All You Need» в 2017 році.

Трансформери демонструють вражаючі результати в багатьох завданнях обробки природної мови (NLP), але їх можна також успішно застосовувати в задачах «text-to-image». Головною перевагою трансформерів є здатность до моделювання довготривалих залежностей у тексті.

Розглянемо основні компоненти трансформера (рисунок 1.2):

– механізм уваги (Attention Mechanism): це ключовий елемент трансформера, який дозволяє моделі фокусуватися на різних частинах

вхідної послідовності. У контексті «text-to-image», механізм уваги може допомогти моделі відділити ключові слова або фрази, які необхідно врахувати під час генерації зображення;

– кодувальний та декодувальний шари (Encoder and Decoder Layers): трансформер складається з набору кодувальних та декодувальних шарів, які працюють разом для обробки вхідної інформації та генерації вихідної. У випадку «text-to-image», кодувальні шари можуть перетворювати текстовий опис у векторну форму, яка потім передається декодувальним шарам для генерації зображення;

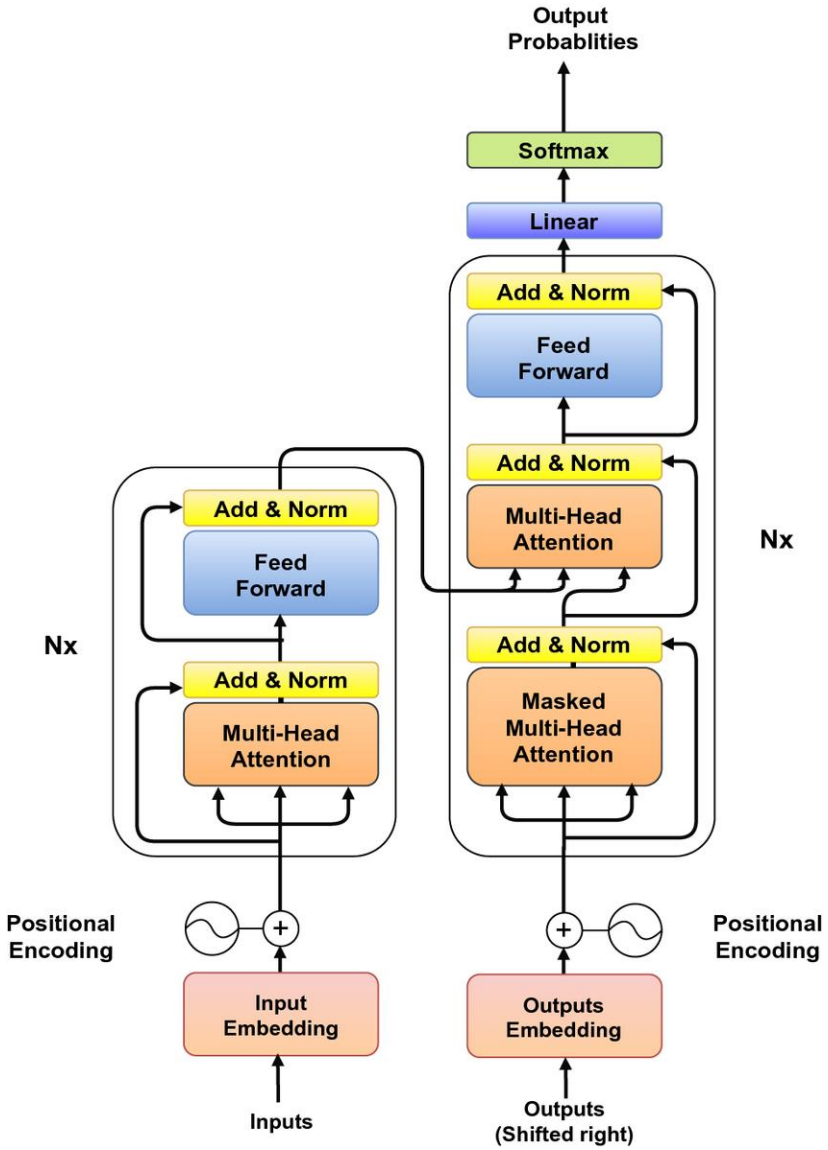

Рисунок 1.2 – Архітектура трансформера

– позиційне кодування (Positional Encoding): так як трансформери не мають внутрішньої структури послідовності, необхідно додатково враховувати позиційну інформацію у вхідних даних. Позиційне кодування дозволяє моделі розрізняти позиції слів у послідовності;

– генеративна модель (Generative Model): декодер трансформера може виступати як генеративна модель, яка приймає закодований текстовий опис та генерує відповідне зображення. Він може використовувати механізм уваги для вибору відповідних частин зображення, щоб надати увагу ключовим елементам текстового опису.

Трансформери складаються з п’яти типів шарів основних, що дозволяють їм ефективно обробляти послідовності даних та може мати додаткові. Основні шари в трансформерах включають:

– шар вбудовування (Embedding Layer): цей шар перетворює індекси вхідних токенів (слів або символів) у векторні представлення. Вбудовування дозволяють моделі представляти слова у просторі з низькорозмірними векторами, де схожі слова розташовані близько одне до одного за семантичним значенням;

– шар позиційного кодування (Positional Encoding Layer): цей шар додає до вбудовувань інформацію про позицію слова в послідовності. Він розв'язує проблему безпосереднього врахування порядку слів у трансформерах, які не мають внутрішньої структури послідовності;

– мультиголовний механізм уваги (Multi-Head Attention Layer): цей шар використовується для моделювання взаємодії між словами у вхідній послідовності. Він обчислює вагований суму значень вкладення для кожного слова, залежно від його відносного значення для інших слів у реченні;

– шар позиційно-залишкового з'єднання (Position-wise Feedforward Layer): цей шар використовується для обробки результатів механізму уваги. Він застосовує повністю зв'язаний шар до кожного позиційного вектора окремо, що дозволяє моделі незалежно обробляти кожне слово в реченні;

– шар нормалізації (Normalization Layer): цей шар застосовується після кожного з перерахованих вище шарів для стабілізації градієнтів та прискорення збіжності моделі.

Крім основних шарів, трансформери також можуть включати додаткові шари, такі як зважене з'єднання (weighted connections), шари уваги на рівні блоків (block-level attention), або шари для задач специфічних завдань, таких як машинний переклад або генерація тексту. Використання цих шарів дозволяє трансформерам ефективно моделювати взаємодію слів у тексті та генерувати високоякісні рішення в різних задачах обробки природньої мови.

Серед вже існуючих моделей, які використовують архітектуру трансформери можна виділити: DeepAI Text to Image API, Artbreeder, Runway ML та можливо DALL-E.

1.6 GAN

Генеративно-суперницькі мережі (Generative Adversarial Network) – це тип нейронних мереж, який складається з двох основних компонентів: генератора і дискримінатора, які навчаються конкурувати між собою у процесі генерації реалістичних даних. Цей підхід до генерації даних був вперше запропонований в 2014 році у роботі «Generative Adversarial Networks» і отримав значний успіх у багатьох областях, включаючи обробку зображень, тексту, звуку та інших видів даних.

Процес навчання GAN включає в себе альтернативне навчання генератора і дискримінатора. На кожній ітерації генератор намагається покращити якість своїх згенерованих даних, тоді як дискримінатор намагається вдосконалити свою здатність розрізняти справжні дані від синтетичних.

Генератор – це нейронна мережа, яка призначена для генерації нових екземплярів даних, що схожі на ті, що ми бачимо в навчальному наборі.

Генератор складається з 5 шарів. Він приймає на вхід випадковий шум або інший векторний представлення і перетворює його в зображення, текст або інші дані (рисунок 1.3). Метою генератора є максимізація ймовірності того, що згенеровані дані будуть відповідати реальним даним.

Дискримінатор – це нейронна мережа, яка використовується для класифікації вхідних даних як справжніх (з навчального набору) або синтетичних (згенерованих генератором). Дискримінатор складається з 5 шарів. Він навчається розрізняти між справжніми та синтетичними даними. Метою дискримінатора є мінімізація ймовірності помилкової класифікації справжніх даних як синтетичних та навпаки.

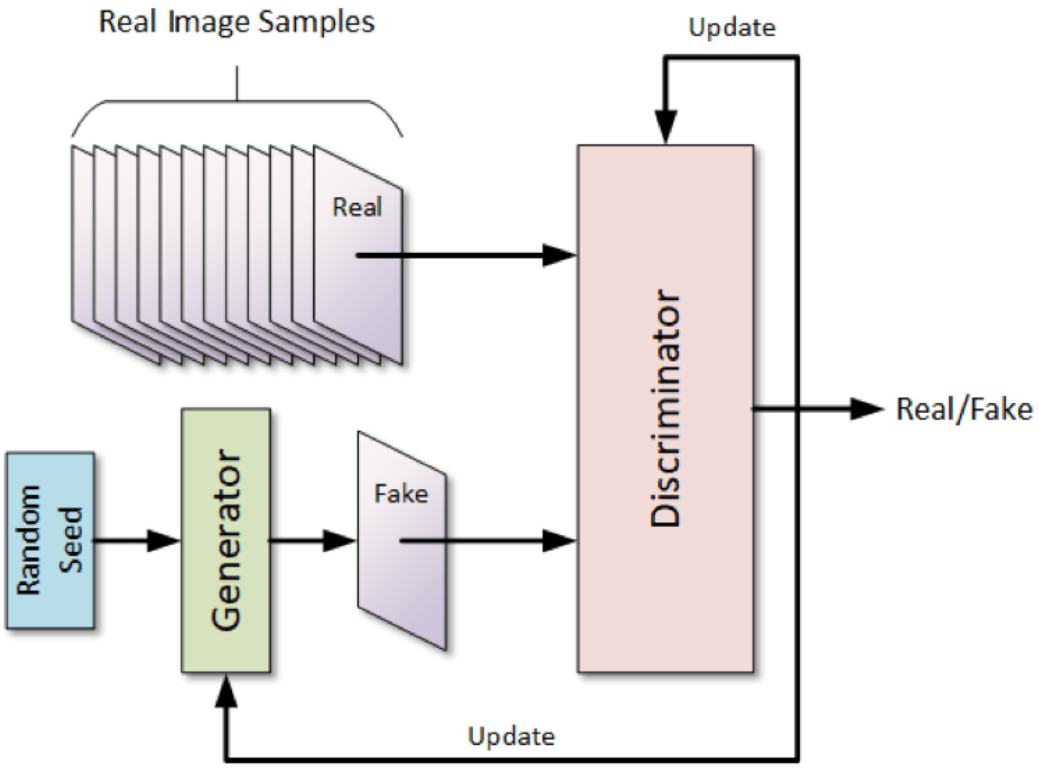

Рисунок 1.3 – Архітектура GAN

Функція втрати в GAN використовується для навчання як генератора, так і дискримінатора. Генератор намагається мінімізувати ймовірність помилкової класифікації його згенерованих даних, тоді як дискримінатор намагається максимізувати ймовірність правильної класифікації.

Серед вже існуючих додатків для генерації зображень за текстовим описом є досить відомий додаток DALL-E. Ця модель спеціалізується на генерації зображень з текстового опису, тобто вона може створювати унікальні та надзвичайно креативні зображення на основі текстових описів.

Основним компонентом архітектури DALL-E є генератор, який перетворює текстові описи у відповідні зображення. Цей генератор може бути базований на трансформерах або інших архітектурах, оптимізованих для генерації зображень.

Що стосується конкретних деталей архітектури DALL-E, OpenAI не розкрила їхні всі аспекти. Однак відомо, що модель DALL-E використовує комбінацію трансформерних блоків і механізмів уваги для взаємодії з текстом і генерації зображень.

Особливість DALL-E полягає в тому, що вона навчається генерувати зображення на основі текстових вводів, які можуть бути досить різноманітними та нестандартними. Наприклад, модель може створювати зображення «кішка з тіста» або «будинок у формі гігантського крокодила» за текстовими описами. Вона відома своєю здатністю до креативного та надзвичайно точного синтезу зображень на основі тексту.

1.7 Порівняння архітектури трансформерів та GAN

Трансформери та генеративно-суперницькі мережі (GAN) – це дві різні архітектури нейронних мереж, кожна з яких має свої переваги і застосування.

Розглянемо переваги трансформерів у порівнянні з GAN. Трансформери виявляються особливо ефективними у завданнях NLP, таких як машинний переклад, генерація тексту та аналіз тексту. Вони дозволяють моделі зосередитися на взаємодії між словами у послідовностях даних, що зазвичай зустрічається в мовленні.

Трансформери можуть ефективно обробляти послідовності даних паралельно, що дозволяє їм працювати швидше за GAN, особливо у випадку обробки великих обсягів тексту або інших послідовностей.

Така архітектура може обробляти вхідні дані різного типу і розмірності, включаючи послідовності різної довжини. У порівнянні, GAN зазвичай працюють з фіксованими розмірами вхідних зображень, текстових послідовностей тощо. Універсальність даної архітектури також дозволяє її використовувати для тексту, зображень та інших типів даних.

Трансформери мають відносно просту архітектуру і можуть бути легко навчені на доступних даних. Вони не вимагають складного тренування, як у випадку GAN, де навчання може бути нестабільним і вимагає додаткових технік, таких як стабілізація навчання.

Хоча трансформери мають свої переваги, важливо враховувати, що кожна архітектура підходить для різних типів задач і відомостей. GAN можуть бути корисними у завданнях генерації зображень та інших синтезованих даних, де важлива реалістичність та деталізація, яку можна досягти за допомогою конкуренції між генератором і дискримінатором. У кожному конкретному випадку важливо враховувати характер даних, обсяг навчального набору, потреби задачі та доступні ресурси, щоб визначити, яка архітектура буде найбільш доцільною для використання.

Застосування GAN все ще має певні обмеження такі як важкість у навчанні через складне налаштування гіперпараметрів і модальний колапс та створення досить зашумлених зображень.

1.8 Двонаправлений LSTM в трансформерах

LSTM (Long Short-Term Memory) – це тип рекурентної нейронної мережі (RNN), розроблений для ефективного навчання та роботи з послідовними даними. LSTM був створений для подолання проблеми короткочасної пам'яті, з якою стикаються традиційні RNN, тобто

неможливості ефективно обробляти довгі послідовності через проблему зникання градієнтів.

Основні компоненти LSTM включають:

а) комірка пам'яті (Memory Cell): основний елемент LSTM, який зберігає інформацію протягом тривалого часу. Комірка пам'яті дозволяє мережі зберігати та оновлювати інформацію з часом, що є ключовою особливістю LSTM;

б) затвори (Gates):

– затвор забування (Forget Gate): вирішує, яку інформацію з комірки пам'яті слід забути. Вона приймає вхід з попереднього стану і поточного вхідного значення та виводить число між 0 і 1, де 0 означає «забути все», а 1 означає «зберегти все»;

– затвор входу (Input Gate): визначає, яку нову інформацію додати до комірки пам'яті. Вона приймає поточне вхідне значення і попередній стан і вирішує, які значення оновити;

– затвор виходу (Output Gate): вирішує, яку інформацію вивести з комірки пам'яті. Вона визначає, які частини комірки пам'яті використовувати для обчислення вихідного стану;

в) оновлення стану: комірка пам'яті оновлюється шляхом зваженого додавання нової інформації та збереження важливої старої інформації;

г) оновлення виходу: оновлюється вихідний стан комірки пам'яті на основі значень затворів і стану пам'яті.

Завдяки цим компонентам, LSTM може ефективно обробляти та зберігати інформацію в довгих послідовностях, що робить його дуже корисним для завдань, які вимагають запам'ятовування контексту на тривалий період часу, таких як машинний переклад, розпізнавання мови, аналіз часових рядів та багато інших.

В контексті трансформерів, LSTM не є основним компонентом, оскільки трансформери спираються на механізм самоуваги (self-attention) для обробки послідовностей. Проте, в деяких гібридних моделях або

архітектурах для специфічних задач можуть використовуватися LSTM разом з трансформерами для покращення моделювання послідовних залежностей.

Двонаправлений LSTM (BiLSTM) – це варіант LSTM, який включає два окремих LSTM: один обробляє послідовність зліва направо, а інший справа наліво (рисунок 1.4). Це дозволяє моделі враховувати контекст як з попередніх, так і з наступних елементів послідовності, що покращує здатність моделі розуміти та обробляти текстові дані.

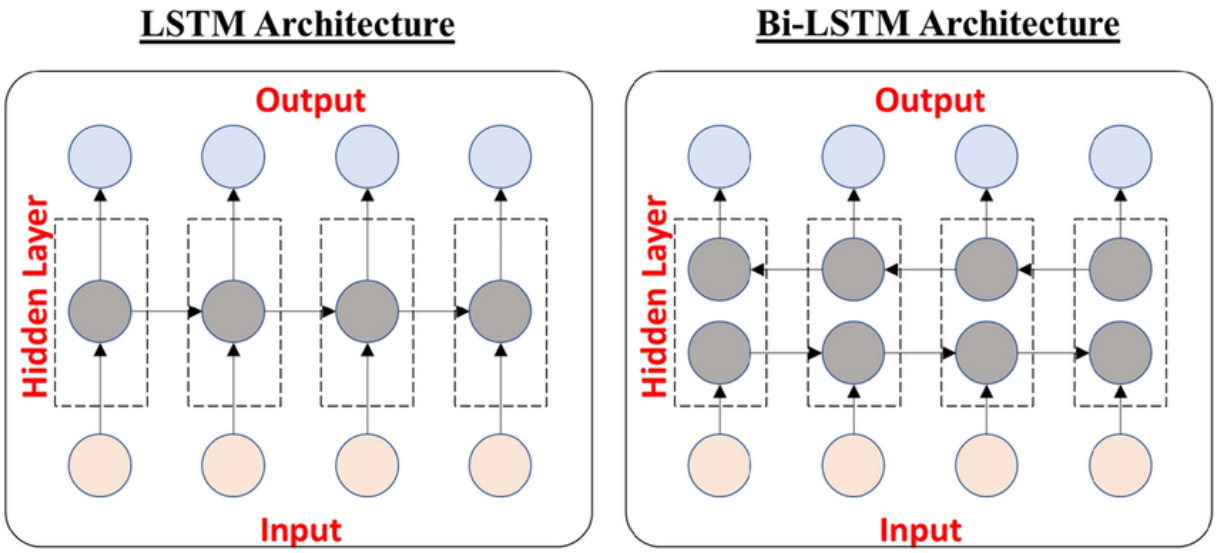

Рисунок 1.4 – Архітектура LSTM та двонаправленої LSTM

Двонаправлений LSTM (Long Short-Term Memory) використовується в трансформерах як один із компонентів кодера для обробки послідовностей даних в обох напрямках. Такий підхід дозволяє збагатити представлення тексту за рахунок інформації з усіх попередніх та наступних слів в реченні. Давайте розглянемо, для чого саме використовується двонаправлений LSTM в трансформерах:

– здатність до контекстуального розуміння: двонаправлений LSTM дозволяє моделі отримати інформацію про контекст як з лівого, так і з правого напрямку послідовності. Це допомагає збагатити представлення

кожного слова або токена із урахуванням всього контексту, включаючи його перед і після;

– підвищення якості представлення: контекстуальне представлення слова або токена, яке отримується за допомогою двонаправленого LSTM, зазвичай більш інформативне і змістовне, оскільки враховується весь контекст, у якому це слово використовується. Це дозволяє моделі краще розуміти смислові зв'язки в тексті;

– зменшення втрати інформації: під час проходження вперед і назад через послідовність двонаправлений LSTM зберігає інформацію про контекст в кожен момент часу. Це допомагає уникнути втрати інформації, що може статися при використанні однонаправленого підходу, де інформація з пізніших слів може втрачатися;

– покращення здатності до моделювання завдань NLP: У задачах, пов'язаних із природною мовою, де важливий контекст та зв'язки між словами, використання двонаправленого LSTM у кодері трансформера може значно покращити результати моделі.

Отже, двонаправлений LSTM у кодері трансформера допомагає забезпечити моделі більш повні і контекстуальні представлення тексту, що веде до кращих результатів у завданнях обробки природної мови.

1.8.1 Механізм самоуваги (self-attention) в трансформерах

Механізм самоуваги (self-attention) є ключовою складовою трансформерів і використовується для обробки послідовностей даних, дозволяючи моделі ефективно фокусуватися на різних частинах вхідної послідовності при обчисленні кожного елемента вихідної послідовності. Самоувага є основою для успіху трансформерів в багатьох завданнях обробки природної мови (NLP) та інших областях, де важливо враховувати контекст елементів даних.

Основні компоненти механізму самоуваги описані далі.

Ключі, запити та значення (Keys, Queries, and Values):

– запити (Queries): вектори, що визначають, на яку інформацію потрібно звернути увагу;

– ключі (Keys): вектори, що допомагають визначити відповідність між запитами та значеннями;

– значення (Values): вектори, що містять фактичну інформацію, яка буде використовуватися в обчисленнях.

Кожен вхідний елемент перетворюється в три вектори (ключ, запит і значення) за допомогою окремих лінійних шарів.

Обчислення ваг. Ваги уваги обчислюються як скалярний добуток запиту (query) та всіх ключів (keys), що нормалізується за допомогою softmax функції. Це дозволяє визначити, наскільки кожен ключ (та відповідне значення) релевантний для поточного запиту.

Формула обчислення ваг уваги:

$$\text{Attention}(Q,K,V)=\text{softmax}(QK^T/\sqrt{d_k})V, \quad (1.1)$$

де Q – матриця запитів;

K – матриця ключів;

V – матриця значень;

d_k – розмірність ключів.

Масштабована точкова продукція (Scaled Dot-Product Attention):

– обчислення включає масштабування скалярного добутку запитів та ключів на $\sqrt{dk}$, щоб запобігти великим значенням градієнтів під час навчання;

– після обчислення ваг застосовується softmax функція для нормалізації ваг до ймовірностей, які визначають важливість кожного елемента в послідовності.

Мультиголовий механізм уваги (Multi-Head Attention). Для покращення здатності моделі фокусуватися на різних частинах вхідної

послідовності, використовується кілька «голів» уваги. Кожна голова обчислює свої ваги уваги і значення паралельно, а потім ці результати об'єднуються. Це дозволяє моделі навчитися різним аспектам взаємодії між елементами послідовності.

Формула мультиголової уваги:

$$\text{MultiHead}(Q,K,V)=\text{Concat}(\text{head}_1,\ldots,\text{head}_h)W^O, \quad (1.2)$$

де $\text{head}_i=\text{Attention}(QW_i^Q,KW_i^K,VWi^V)$;

W^O – лінійний шар для об'єднання результатів.

Переваги механізму самоуваги:

– паралелізація: на відміну від рекурентних нейронних мереж (RNN), механізм самоуваги дозволяє паралельно обробляти всі елементи послідовності, що значно прискорює навчання та виконання моделі;

– гнучкість контексту: самоувага дозволяє моделі фокусуватися на будь-яких частинах послідовності, незалежно від їх відстані, що важливо для завдань, де ключова інформація може знаходитися в різних частинах тексту;

– масштабованість: трансформери, засновані на механізмі самоуваги, добре масштабуются для великих наборів даних та великих моделей, що дозволяє досягати високої продуктивності у складних завданнях.

Механізм самоуваги є критичним компонентом трансформерів, забезпечуючи ефективну та гнучку обробку послідовностей даних, що робить їх надзвичайно потужними для різних задач у галузі обробки природної мови та поза нею.

Загалом використання механізму самоуваги є зараз досить поширеним інструментом. Розглянемо приклади використання. Механізм самоуваги (self-attention) широко використовується в різних задачах обробки природної мови (NLP) та інших областях. Ось кілька прикладів використання цього механізму:

а) машинний переклад:

– Transformer (Vaswani et al., 2017): один з найвідоміших прикладів, де механізм самоуваги використовується для покращення перекладу тексту з однієї мови на іншу. Трансформер замінює рекурентні нейронні мережі (RNN) та довгі короткочасні пам'яті (LSTM), забезпечуючи ефективну паралелізацію обробки та кращу здатність враховувати довготривалі залежності в тексті;

б) генерація тексту:

– GPT (Generative Pre-trained Transformer): модель GPT використовує механізм самоуваги для генерування тексту на основі вхідних даних. Це дозволяє моделі враховувати контекст при створенні зв'язного та граматично правильного тексту;

– GPT-2 та GPT-3: більш потужні версії GPT, що використовують глибші архітектури та більші набори даних для покращення якості та креативності генерованого тексту;

в) аналіз тональності тексту (Sentiment Analysis):

– BERT (Bidirectional Encoder Representations from Transformers): модель BERT використовує двонаправлений механізм самоуваги для отримання контексту з обох сторін кожного слова в реченні. Це дозволяє моделі краще розуміти тональність та інші аспекти тексту, що покращує точність аналізу тональності;

г) резюмування тексту (Text Summarization):

– BERTSUM: модель, яка використовує BERT для резюмування тексту. Вона застосовує механізм самоуваги для вибору найважливіших частин тексту, що повинні бути включені в резюме;

– PEGASUS: модель для резюмування тексту, яка використовує механізм самоуваги для розуміння контексту та виділення ключових речень в тексті;

д) відповіді на питання (Question Answering):

– BERT та RoBERTa: моделі, які використовують механізм самоуваги для розуміння запитання та знаходження відповідної інформації в тексті. Вони можуть точно відповісти на питання, базуючись на контексті наданого тексту;

– T5 (Text-To-Text Transfer Transformer): використовує підхід «текст-в-текст» для різних NLP задач, включаючи відповіді на питання, де механізм самоуваги допомагає враховувати контекст запитання та тексту;

е) обробка зображень (Computer Vision):

– Vision Transformer (ViT): застосовує механізм самоуваги до задач комп'ютерного зору, обробляючи зображення як послідовність патчів (фрагментів) і використовуючи самоувагу для взаємодії між цими патчами. Це дозволяє моделі ефективно аналізувати зображення та виконувати завдання, такі як класифікація зображень та сегментація;

ж) мультимодальні задачі:

– CLIP (Contrastive Language–Image Pre-training): модель, яка об'єднує текстові та візуальні дані, використовуючи механізм самоуваги для навчання спільного представлення тексту та зображень. Це дозволяє моделі виконувати завдання, такі як пошук зображень за текстовим запитом.

Механізм самоуваги є надзвичайно універсальним і потужним інструментом, який знаходить застосування в багатьох різних областях завдяки своїй здатності ефективно обробляти та інтегрувати інформацію з послідовних даних.

2 ТРАНСФОРМЕРИ В ЗАДАЧІ «TEXT-TO-IMAGE»

В наші дні робота багатьох сфер праці дуже полегшана завдяки комп’ютерам та їхнім можливостям, а також безпосередньо завдяки штучному інтелекту. Якщо взяти наприклад роботу дизайнерів графічних, то малювання ілюстрацій, логотипів, іконок та багато чого іншого потребує завжди креативу, натхнення інакше б у всіх все було однаково. В таких випадках на поміч приходить нейронна мережа по генерації зображень за допомогою трансформерів.

Трансформери грають важливу роль у задачах «text-to-image», забезпечуючи потужні засоби для інтеграції текстової інформації та генерації зображень, відповідно до наданого текстового опису. Розглянемо детальніше, як трансформери використовуються в цих задачах. Трансформери складаються з блоків самоуваги та пропускної лінійної мережі (FFN), які дозволяють моделі ефективно обробляти послідовні дані, враховуючи контекст кожного елемента. У випадку задач «text-to-image», це означає обробку текстових описів для генерації відповідних зображень.

Основні компоненти трансформера:

– енкодинг тексту. Трансформери використовуються для перетворення текстового опису у векторні представлення, які зберігають семантичну інформацію. Це дозволяє моделі зрозуміти контекст і ключові деталі тексту;

– інтеграція механізму уваги. Механізм уваги в трансформерах дозволяє моделі зосередитися на важливих частинах текстового опису під час генерації зображення. Це критично важливо для врахування деталей, таких як об'єкти, кольори, розташування та інші характеристики, згадані в тексті;

– генерація зображення. Використовуючи векторні представлення тексту, генератор (часто базується на архітектурі трансформера) створює зображення. Генератор може бути розширений за допомогою

спеціалізованих блоків, які навчаються перетворювати текстові вектори у візуальні представлення;

– мультиголовий механізм уваг. Цей механізм дозволяє моделі враховувати різні аспекти текстового опису одночасно, що допомагає створити більш точне і детальне зображення;

– поєднання тексту і зображень. У більш складних моделях, таких як DALL-E, використовується архітектура, яка інтегрує як текстові, так і візуальні дані, використовуючи трансформери для кодування обох типів даних у спільний простір. Це дозволяє моделі генерувати зображення, які відповідають заданому текстовому опису, а також розуміти зворотний зв'язок між текстом і зображенням.

Як приклад таких моделей можна привести VQ-VAE-2 (рисунок 2.1) та DALL-E.

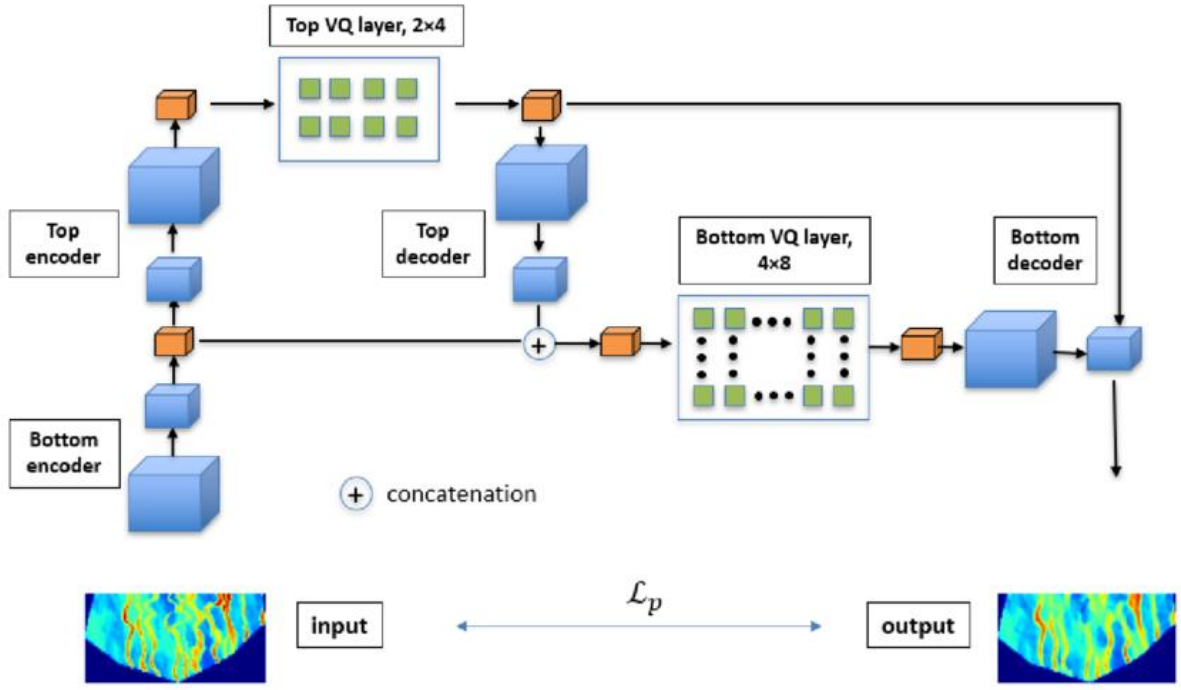

Рисунок 2.1 – Архітектура VQ-VAE-2

Розглянемо кожну модель детальніше.

Звичайно повністю архітектуру DALL-E моделі розробники не розкривають, але основні моменти деякі все ж таки відомо. Модель, розроблена OpenAI, використовує трансформери для перетворення текстового опису у векторні представлення, які потім використовуються для генерації зображень. DALL-E демонструє здатність створювати високоякісні зображення, враховуючи складні текстові запити. Модель VQ-VAE-2, яка поєднує варіаційні автокодери з трансформерами. Вона кодує зображення у векторні представлення і використовує трансформери для обробки тексту, забезпечуючи синтез нових зображень на основі текстового опису.

Архітектура DALL-E поєднує ідеї трансформерів та векторних квантованих автокодерів. Вона складається з наступних основних компонентів:

– енкодер тексту. Текстовий опис подається на вхід трансформерному енкодеру, який перетворює його на послідовність векторних представлень. Це схоже на обробку тексту в моделях GPT (Generative Pre-trained Transformer);

– квантований автокодер (VQ-VAE-2). Використовується варіаційний автокодер з векторною квантовкою (VQ-VAE-2) для кодування зображень у векторні представлення (латентні коди). Цей автокодер навчений окремо і перетворює зображення у компактні латентні вектори, які потім можуть бути декодовані назад у зображення;

– трансформер для генерації. Основна частина архітектури DALL-E – це великий трансформер, який навчений передбачати латентні коди зображення, виходячи з текстового опису. Він генерує послідовність латентних кодів, які потім можуть бути перетворені на зображення за допомогою VQ-VAE-2 декодера.

Латентні коди – це внутрішні представлення даних, які отримуються на проміжних етапах роботи нейронної мережі, зокрема у варіаційних автокодерах (VAE) та інших моделях генеративного навчання, таких як

DALL-E. Вони є компактними векторами, які містять стислу, але суттєву інформацію про вхідні дані.

Основні характеристики латентних кодів:

– стиснення інформації: латентні коди являють собою стислий варіант вхідних даних. Наприклад, зображення високої роздільної здатності може бути закодовано у вектор з набагато меншою кількістю елементів;

– суттєві ознаки: латентні коди зберігають ключові ознаки даних, які дозволяють реконструювати оригінальні дані з високою точністю. У випадку зображень це можуть бути форми, кольори та інші візуальні характеристики;

– простір латентних представлень: латентний простір – це багатовимірний простір, у якому знаходяться всі можливі латентні коди. Навчені моделі можуть інтерполювати в цьому просторі для генерації нових, раніше не бачених прикладів.

Розглянемо приклад роботи з латентними кодами у VAE:

– енкодер: енкодер приймає вхідні дані (наприклад, зображення) і перетворює їх на латентний код. Цей код є вектором у латентному просторі;

– декодер: декодер приймає латентний код і відновлює з нього вхідні дані. В ідеалі, відновлені дані мають бути якомога ближче до оригіналу;

– варіаційний підхід: у варіаційних автокодерах (VAE) енкодер не просто створює один латентний код, а генерує параметри розподілу (середнє значення і дисперсію). З цього розподілу потім вибираються конкретні коди для декодування, що забезпечує варіативність у генерації нових даних.

Розглянемо варіанти використання латентних кодів у DALL-E.

По-перше, кодування зображень. DALL-E використовує векторно-квантований автокодер (VQ-VAE-2) для кодування зображень у латентні коди. Ці коди містять стиснуту інформацію про зображення і можуть бути ефективно використані для генерації.

По-друге, генерація зображень на основі тексту: текстовий опис перетворюється на векторне представлення за допомогою трансформера.

Потім модель трансформера генерує латентні коди зображення на основі цього векторного представлення тексту.

По-третє, відновлення зображень: латентні коди, створені трансформером, подаються на вхід декодеру VQ-VAE-2, який відновлює зображення з цих кодів. В результаті, модель DALL-E створює нові зображення, відповідні текстовим описам.

Латентні коди є критично важливими для генеративних моделей, оскільки вони дозволяють: стискати дані (латентні коди зменшують розмір даних, що полегшує їх обробку), генерувати нові приклади (латентні коди можна використовувати для створення нових зображень або інших даних, які схожі на оригінальні, але мають деякі нові властивості), забезпечувати варіативність (моделі можуть створювати різні варіанти зображень або інших даних, змінюючи латентні коди).

Латентні коди відіграють ключову роль у багатьох сучасних генеративних моделях, забезпечуючи ефективне кодування та декодування складних даних, таких як зображення.

Процес роботи DALL-E можна розділити на кілька етапів.

Перший етап. Навчання автокодера: автокодер VQ-VAE-2 навчається на великому наборі зображень. Він вчиться кодувати зображення у латентні вектори (кванти) та відновлювати зображення з цих векторів. Це забезпечує компактне представлення зображень, яке можна використовувати для генерації.

Другий етап. Навчання трансформера: трансформер навчений на парі текстових описів і відповідних латентних кодів зображень. Використовуючи механізм уваги, трансформер навчається передбачати послідовність латентних кодів, які відповідають заданому текстовому опису.

Третій етап. Генерація зображень: під час генерації зображення текстовий опис подається на вхід трансформеру, який генерує відповідні

латентні коди. Ці коди потім подаються на вхід декодеру VQ-VAE-2, який відновлює зображення з цих кодів.

Модель DALL-E була однією з перших моделей по генерації зображень по текстовому опису, а отже саме вона і внесла в розвиток цього напрямку деякі інновації, розглянемо переваги та інновації DALL-E:

– гнучкість у генерації зображень: DALL-E може генерувати зображення з широкого спектру текстових описів, включаючи складні сцени, абстрактні концепції та конкретні деталі;

– висока якість зображень: використання VQ-VAE-2 забезпечує високу якість відновлених зображень, оскільки автокодер здатен зберігати важливі деталі зображення у своїх латентних представленнях;

– інтеграція тексту та зображень: трансформери добре справляються з обробкою послідовностей і можуть ефективно інтегрувати текстову інформацію, враховуючи контекст та деталі опису.

Якою б не була модель продуманою, інноваційною, все одно є певні недоліки. Розглянемо недоліки моделі DALL-E більш детально.

Перший недолік – обчислювальні ресурси. Навчання таких моделей вимагає значних обчислювальних ресурсів, що може бути викликом для менших організацій.

Другий недолік – великі обсяги даних. Для навчання DALL-E потрібні великі набори даних пар «текст-зображення», що також може бути обмеженням.

Третій недолік – контроль якості. Генеровані зображення можуть іноді не повністю відповідати текстовому опису або містити артефакти, що вимагає подальшого вдосконалення моделей.

3 РОЗРОБКА АРХІТЕКТУРИ

Трансформери (Transformers) – це сучасна архітектура нейронних мереж, яка значно покращила продуктивність моделей машинного навчання, особливо у завданнях обробки природної мови (NLP) та розпізнавання зображень. Архітектура трансформерів була представлена в 2017 році дослідниками Google в статті «Attention is All You Need». Найкращим прикладом роботи трансформерів є речення, адже воно складається з впорядкованого набору слів.

Трансформери створюють цифрове уявлення кожного елемента послідовності, інкапсулюють важливу інформацію про нього і навколишній контекст. Уявленя, які було отримано надалі можна передавати в інші нейронні мережі. Це зроблено для подальшої можливості цих нейронних мереж скористаються цією інформацією задля розв'язання певних задач, зокрема для синтезу і класифікації. Створені за допомогою трансформерів такі уявленя щодо певної інформації, вони допомагають якнайкраще виявити та зрозуміти приховані паттерни та взаємозв’язки вхідних даних іншим нейронним мережам. А отже, трансформери добре працюють в задачах синтезу послідовностей та взаємопов’язаних результатів.

Головна перевага трансформерів полягає в їх здатності обробляти послідовності даних паралельно завдяки механізму уваги (attention mechanism), зокрема, мультиголовій самоувазі (multi-head self-attention). Це надає трансформерам значні переваги над попередніми архітектурами, такими як рекурентні нейронні мережі (RNN) та довго-короткочасна пам'ять (LSTM). Ось основні переваги трансформерів:

а) паралелізація:

– ефективність обчислень: трансформери дозволяють обробляти всі елементи вхідної послідовності одночасно, що значно прискорює тренування та зменшує час обробки даних;

– використання апаратного прискорення: завдяки можливості паралелізації, трансформери ефективно використовують сучасні графічні процесори (GPU) та тензорні процесори (TPU), що прискорює навчання великих моделей;

б) контекст та довгострокові залежності:

– глобальний контекст: механізм уваги дозволяє моделі враховувати всі елементи послідовності при обчисленні кожного вихідного елемента, що покращує розуміння контексту;

– довгострокові залежності: трансформери можуть ефективно обробляти залежності між далекими елементами в послідовності, що є проблемою для RNN та LSTM через їх обмежену здатність зберігати довгострокову інформацію;

в) гнучкість та узагальнюваність:

– універсальність: трансформери можуть бути використані для широкого спектра задач, включаючи обробку природної мови, машинний переклад, генерацію тексту, розпізнавання зображень та інші;

– масштабованість: архітектура трансформерів добре масштабується, дозволяючи створювати моделі з мільярдами параметрів, такі як GPT-3 від OpenAI або BERT від Google.

Розглянемо деякі приклади застосування трансформерів:

– BERT (Bidirectional Encoder Representations from Transformers): використовується для задач розуміння природної мови, включаючи аналіз настроїв, відповіді на запитання та розпізнавання сутностей;

– GPT (Generative Pre-trained Transformer): використовується для генерації тексту, машинного перекладу, створення діалогових систем та інших задач.

Перш за все перед наданням вхідних даних до трансформерів, з цих даних необхідно зробити послідовні токени. Токени – це основні елементи, на які розбивається текст при обробці природної мови (NLP). Токенізація –

це процес перетворення тексту у список токенів. Кожен токен може бути словом, частиною слова, символом або групою символів залежно від завдання та методів токенізації, що використовуються.

Види токенів:

– слова (Word Tokens): кожне слово є окремим токеном. Наприклад, у реченні «The cat sat on the mat» кожне слово розглядається як окремий токен: [«The», «cat», «sat», «on», «the», «mat»];

– підслова або морфеми (Subword Tokens): слова можуть бути розділені на менші частини або морфеми, особливо у мовах з великою кількістю словоформ. Наприклад, слово «unhappiness» може бути розділене на [«un», «happiness»];

– символи (Character Tokens): кожен символ тексту є окремим токеном. Це корисно для мов з великим алфавітом або для обробки текстів з помилками або нестандартним написанням. Наприклад, «hello» може бути токенізоване як [«h», «e», «l», «l», «o»].

Методи токенізації:

– пробіл-розділена токенізація (Whitespace Tokenization): текст розбивається на слова на основі пробілів. Це простий метод, але він не враховує пунктуацію та інші символи;

– токенізація за регулярними виразами (Regex Tokenization): використовуються регулярні вирази для точнішого розділення тексту на токени. Наприклад, можна виділяти слова, враховуючи пунктуацію;

– Byte-Pair Encoding (BPE): цей метод комбінує часто зустрічаються пари символів у нові токени, що дозволяє ефективно працювати з рідкісними словами. Він часто використовується у трансформерах;

– WordPiece: подібний до BPE, але використовує інший алгоритм для створення токенів. Застосовується в моделях, таких як BERT;

– SentencePiece: метод, який не вимагає попередньої токенізації тексту, що робить його більш універсальним для різних мов і задач.

Застосування токенів у машинному навчанні:

– моделі обробки природної мови: токени є основними вхідними даними для моделей, таких як трансформери, які навчаються на послідовностях токенів;

– векторизація тексту: токени перетворюються у числові вектори (ембеддінги), які використовуються як вхід для нейронних мереж;

– аналіз тексту: токени використовуються для виконання різних аналізів тексту, таких як частотний аналіз, пошук ключових слів, машинний переклад тощо.

Розглянемо простий приклад роботи з трансформерами та токенами на прикладі задачі перетворення певної послідовності токенів, використовуючи словник, який буде відігравати роль таблиці для пошуку. Необхідно зіставити між собою слова та числа, адже можна співставити будь-яке слово з будь-яким числом (рисунок 3.1).

```
Raw text: A black cat
Vocab: {"A": 0, "black": 1, "cat": 2, "cats": 3}
Tokenized text: [0, 1, 2]
```

Рисунок 3.1 – Приклад кодування тексту

Ми можемо бачити найпростіший приклад кодування тексту та можна помітити, що слова cats та cat було закодовано різними токенами, незважаючи на те, що це одне і те слово, але у різній формі множини.

Також існують вже більш продвинуті способи токенізації при якій перед тим як присвоїти індекси словам, їх розбивають на фрагменти. В ході експериментів також була помічена зручність у використання окремих токенів, які б позначали кінець та початок речень, адже це надає більше контексту.

Розглянемо приклад токенізаціі на представленому реченні: «Hello there, isn't the weather nice today in Drosval?»

Використовуючи токенізатор bert-base-uncased (BBU) із бібліотеки «transformers library» можна перетворити речення на послідовність токенів, яку можна побачити на рисунку 2.2.

```
[101, 7592, 2045, 1010, 3475, 1521, 1056, 1996, 4633, 3835, 2651, 1999, 2852, 2891, 10175, 1029, 102]
```

Рисунок 3.2 – Перетворення речення на послідовність токенів

Числа, які було присвоєно словам будуть змінюватися в залежності від обраного способу токенізації та від методу навчання моделі.

Після декодування представлено слова, які було отримано з токенів на рисунку 3.3.

```
[CLS] hello there , isn ’ t the weather nice today in dr ##os ##val ? [SEP]
```

Рисунок 3.3 – Слова представлені токенами

Як можна побачити, результат не такий, як ми очікували, але кінцевий результат не схожий на вхідне речення. Такий результат вийшов через те, що було додано певні додаткові токени та вигадану нами назву модель представила декількома фрагментами. Великі літери також було втрачено з причини використання такого виду моделі, яка їх не підтримує.

Треба враховувати, що трансформери також можуть обробляти не лише текст, а й виконувати певні задачі з візуалізації.

3.1 Ембеддінг токенів

Ембедінги (embeddings) – це спосіб представлення об'єктів (зазвичай слів або фраз) у вигляді векторів у багатовимірному просторі. Вони використовуються в машинному навчанні, зокрема у завданнях обробки

природної мови (NLP) та обробки зображень, щоб перетворити нечислові дані у числовий формат, зручний для використання в нейронних мережах та інших алгоритмах.

Основні концепції ембедінгів:

– контекстуальність: ембедінги дозволяють враховувати контекст слова, тобто значення слова може змінюватися залежно від його оточення в тексті. Це забезпечує кращу семантичну інтерпретацію порівняно з традиційними методами, як-от Bag-of-Words (BoW) або TF-IDF;

– високовимірний простір: слова або інші об'єкти представляються у вигляді векторів у просторі з багатьма вимірами. Вектори, що представляють схожі за значенням слова, розташовані ближче один до одного.

Методи створення ембедінгів:

– Word2Vec: алгоритм, розроблений у 2013 році командою Google, який створює векторні представлення слів. Існують два основні варіанти: Skip-gram та CBOW (Continuous Bag of Words). Вони використовують нейронні мережі для передбачення контексту слова або слова за його контекстом;

– GloVe (Global Vectors for Word Representation): розроблений командою Stanford, цей метод використовує статистичну інформацію про частоти спільного використання слів у великому корпусі текстів для побудови векторів;

– FastText: розроблений Facebook, цей метод розширює Word2Vec, включаючи підслова, що дозволяє моделі краще працювати з морфологічно складними мовами та рідкісними словами;

– Contextual Embeddings (BERT, GPT, ELMo): ці моделі враховують контекст слова у реченні, що дозволяє створювати різні представлення для одного і того ж слова залежно від його оточення. Наприклад, BERT (Bidirectional Encoder Representations from Transformers) створює

контекстуальні ембедінги, що враховують всю послідовність слів у обох напрямках.

Використання ембедінгів:

– класифікація тексту: ембедінги використовуються як вхідні дані для нейронних мереж, які виконують класифікацію тексту на різні категорії;

– аналіз настроїв: вектори слів можуть допомогти визначити позитивний або негативний настрій тексту;

– машинний переклад: ембедінги використовуються для представлення слів у різних мовах, що полегшує процес перекладу;

– пошук та рекомендації: схожість між ембедінгами можна використовувати для пошуку схожих документів або рекомендацій товарів.

Маючи послідовність цілих чисел, яка представляє собою вхідні дані, то ми цю послідовність можемо перетворити на ембеддінги, адже вони можуть передавати скорочений сенс кожного токена у вигляді послідовності чисел.

Спершу виконується синтез послідовних чисел в якості випадкової послідовності, а вже потім безпосередньо під час навчання формується значиме уявлення. Головним обмеження ембеддингу є те, що він не враховує контекст речення, у якому і було виконано синтез токенів. Такий недолік має спадковий характер, а причини його виникнення формують усього дві.

Головним фактором завжди є вхідна задача, адже саме від неї будуть залежати подальші дії, бо може бути необхідно зберегти порядок токенів під час їх перетворення на ембеддінги. Це має найбільш важливе значення саме під час обробки природних мов. Головна ідея полягає в тому, що ми використовуємо ще один набір ембеддінгів, який надає положення кожного токена у певній послідовності (рисунок 3.4). Такий набір комбінується зазвичай з емеддінгами токенів.

Рисунок 3.4 – Ембединг токенів

Ще однією проблемою є те, що самі токени можуть мати різне значення, яке буде залежати від сусідніх токенів. Наприклад, у реченнях: «It’s dark, who turned off the light?» та «Wow, this parcel is really light!» тут слово light використовується у двох різних контекстах, і тому має абсолютно різні значення. Але найймовірніше, що, залежно від методу токенізації, ембеддінги будуть однакові. У трансформерах це вирішується за допомогою механізму уваги.

3.2 Механізм уваги

Одним з найважливіший механізмів у архітектурі трансформерів є увага. Такий механізм дає можливість нейронній мережі зрозуміти, яка частина вхідної послідовності найрелевантніша до завдання. Механізм уваги визначає для кожного токена послідовності, які інші токени необхідні для його розуміння в даному контексті.

Механізм уваги (рисунок 3.5) можна представити як метод, який замінює кожен ембеддінг токена на ембеддінг, який містить інформацію щодо сусідніх токенів, замість застосування одного й того самого ембеддінга для кожного токена незалежно від контексту.

Знаючи, які токени є релевантними поточному, можна дізнатися його контекст за допомогою середньозваженої – або, в загальному випадку, лінійної комбінації – цих ембеддингів.

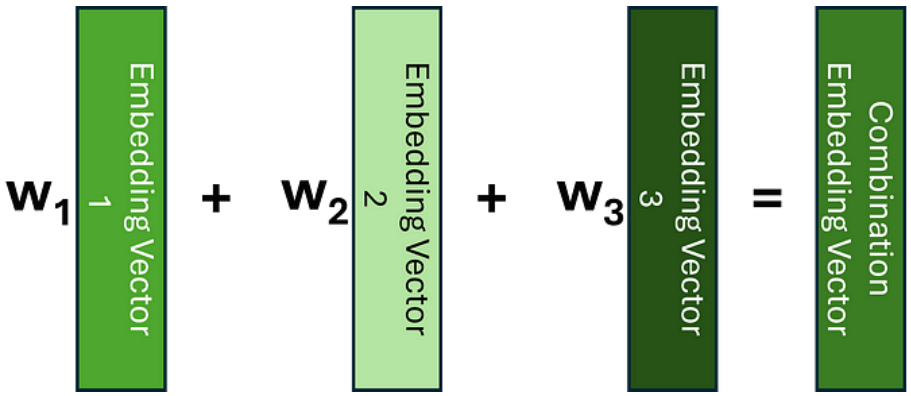

Рисунок 3.5 – Механізм уваги

Перед тим як ми застосуємо механізм уваги, послідовність ембеддінгів не містить контексту їхніх сусідів. А отже, ми маємо можливість візуалізувати ембеддінг для слова light як лінійну комбінацію (рисунок 3.6).

Token Embedding	It's	Dark	Who	Turned	Off	The	Light
Weight	0	0	0	0	0	0	1

Рисунок 3.6 – Візуалізація ембедінгу для слова light

Ваги – це матриця тотожностей. Застосувавши механізм уваги, ми можемо дізнатися матрицю ваг, яка надає нам можливість представити ембеддінг для слова light (рисунок 3.7).

Token Embedding	It's	Dark	Who	Turned	Off	The	Light
Weight	0	0.3	0	0.1	0.1	0	0.5

Рисунок 3.7 – Застосування механізму уваги для виразу ембедінгу слова light

Ми можемо побачити, що ембеддінгам, які відповідають найбільш релевантним для обраного токена частинам послідовності, присвоєно ваги більшого розміру. Це має забезпечити розміщення в новому векторі ембеддинга найважливішого контексту. Ембеддінги, що містять інформацію про їхній поточний контекст, іноді називають контекстуалізованими, і саме такі нам необхідні.

3.3 Розрахунок уваги

Існують досить різноманітні варіанти такого механізму. Головною відмінністтю між ними є спосіб обчислення ваг для лінійної комбінації.Розглянемо детальніше scaled dot-product attention (механізм на основі скалярного добутку), адже це найпопулярніший спосіб. Будемо вважати, що всі наші ембеддінги позиційно закодовані.

Головною метою є створення контекстуалізованих ембеддінгів за допомогою лінійних комбінацій вихідних ембеддингів. Спершу припустимо, що ми маємо можливість закодувати всю необхідну інформацію в наших вивчених векторахта далі необхіідно виконати обчислення вагів. Для цього потрібно визначити, які токени релевантні один одному. Задамо поняття схожості між двома ембеддінгами. Одиним зі способів представлення цього поняття схожості є скалярний добуток. Тобто ми хочемо визначити ембеддінги, при яких чим більший добуток, тим більша схожість двох слів (рисунок 3.8).

Оскільки для кожного токена потрібно обчислити його взаємозв'язок із кожним іншим токеном послідовності, цю задачу можна узагальнити до матричного множення, в результаті якого отримуємо матрицю ваг, відому як attention scores (оцінки уваги). Щоб сума ваг дорівнювала одиниці, застосовуємо багатозмінну логістичну функцію (БЛФ). Однак матричне множення може давати дуже великі числа, через що БЛФ повертатиме дуже маленькі градієнти для великих оцінок уваги, що може призвести до

проблеми зникаючого градієнта під час навчання. Для розв'язання цієї проблеми ми помножимо оцінки уваги на поправочний коефіцієнт перед застосуванням БЛФ (рисунок 3.9).

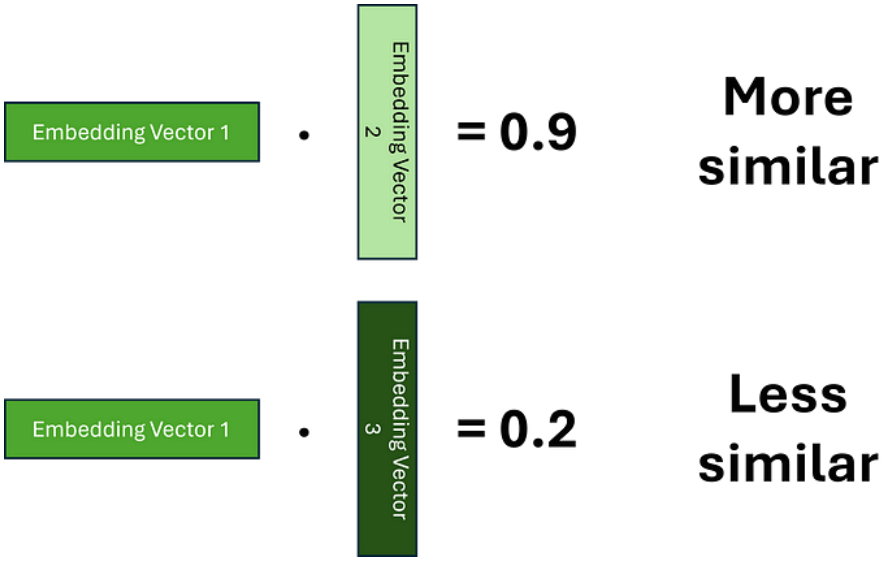

Рисунок 3.8 – Визначення ембеддінгів

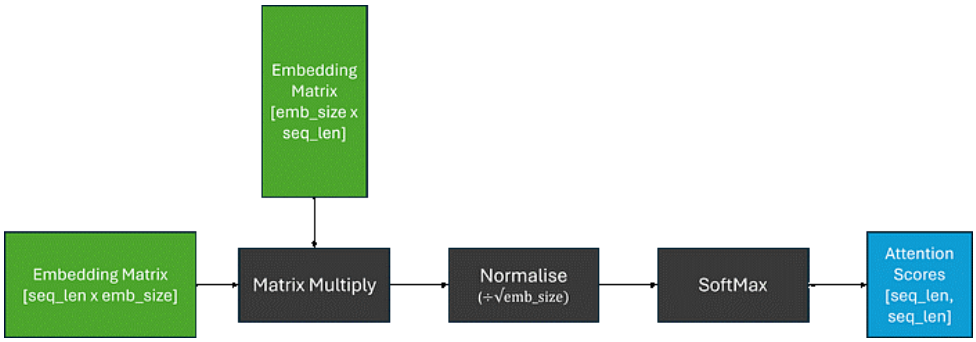

Рисунок 3.9 – Помножимо оцінки уваги на поправочний коефіцієнт

Тепер, щоб отримати матрицю контекстуалізованих ембеддінгів, ми можемо помножити оцінки уваги на матрицю вихідних ембеддінгів (рисунок 3.10). Це еквівалентно взяттю лінійних комбінацій ембеддінгів. Хоча модель може навчитися створювати складні ембеддінги для синтезу оцінок уваги та контекстуальних ембеддінгів, це може призвести до перевантаження інформацією.

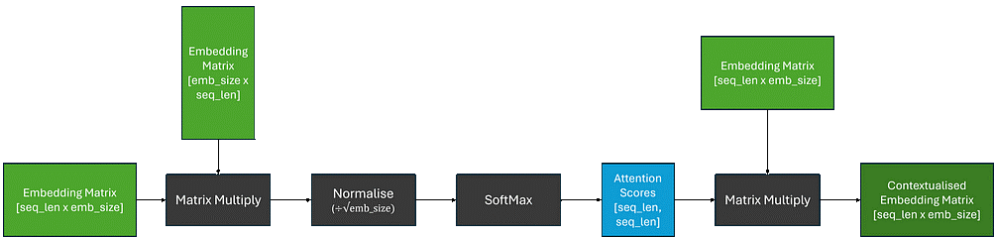

Рисунок 3.10 – Помноження оцінки уваги на матрицю вихідних ембеддингів

Тому, щоб полегшити навчання моделі, замість прямого використання матриці ембеддінгів ми будемо передавати її через три незалежні лінійні шари (матричні множення, рисунок 3.11). Це дозволить моделі «зосередитися» на різних аспектах ембеддінгів.

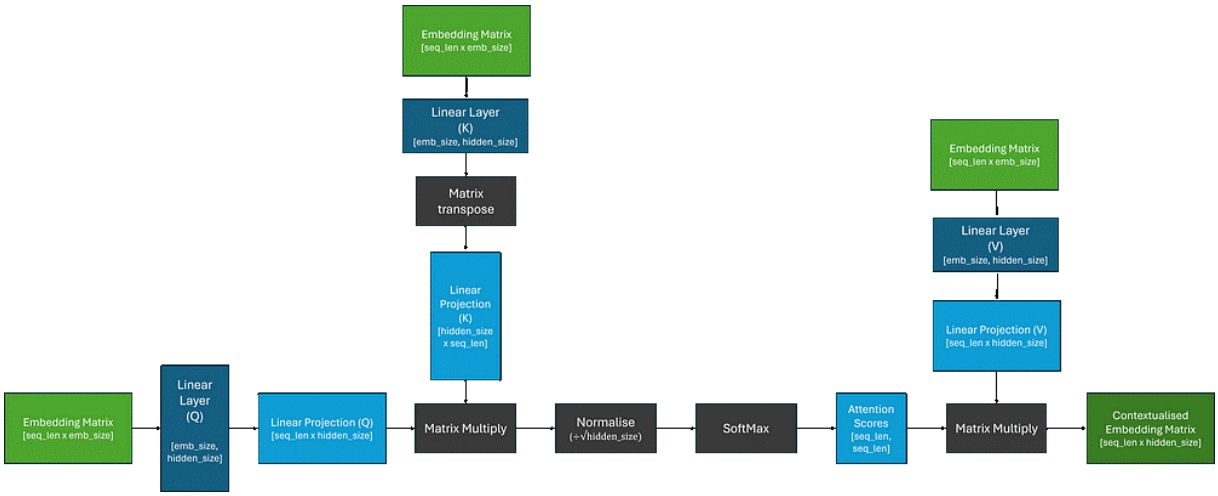

Рисунок 3.11 – Передача матриці через три незалежні шари

На лінійні проекції позначені як Q, K і V. Ми розібралися в роботі процесу, і тепер обчислення уваги можна уявити як блок із трьома входами, які подаються в Q, K і V (рисунок 3.12).

Коли ми подаємо одну матрицю в Q, K і V, це називається self-attention тобто виявлення закономірностей тільки між вхідними даними.

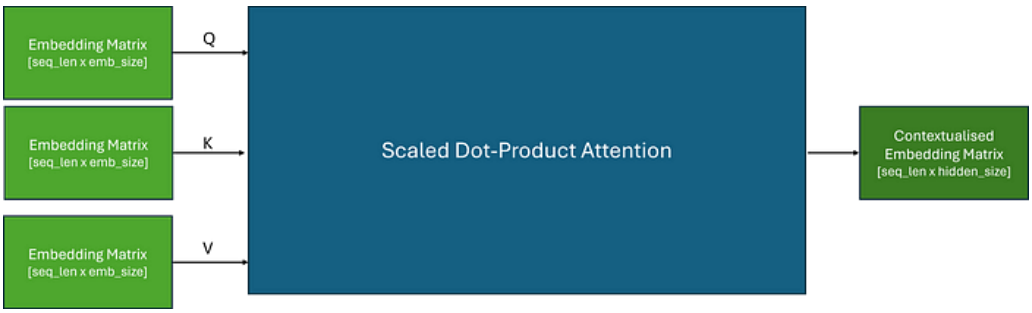

Рисунок 3.12 – Обчислення уваги

3.4 Multi-head attention

Зазвичай дуже часто паралельно запускається декілька блоків self-attention, це робиться для того, щоб наш трансформер одночасно міг обробляти різні частини вхідної послідовності – це називається multi-head attention. Виходи кількох незалежних блоків self-attention конкатенуються і передаються через лінійний шар. Він дозволяє моделі комбінувати контекстуальну інформацію з кожного блоку уваги.

Зазвичай прихований розмір виміру в кожному блоці self-attention обирають в якості розміру вихідного ембеддінга, який поділено на певну кількість блоків уваги, задля того, щоб вдалося зберегти структуру матриці цих ембеддінгів (рисунок 3.13).

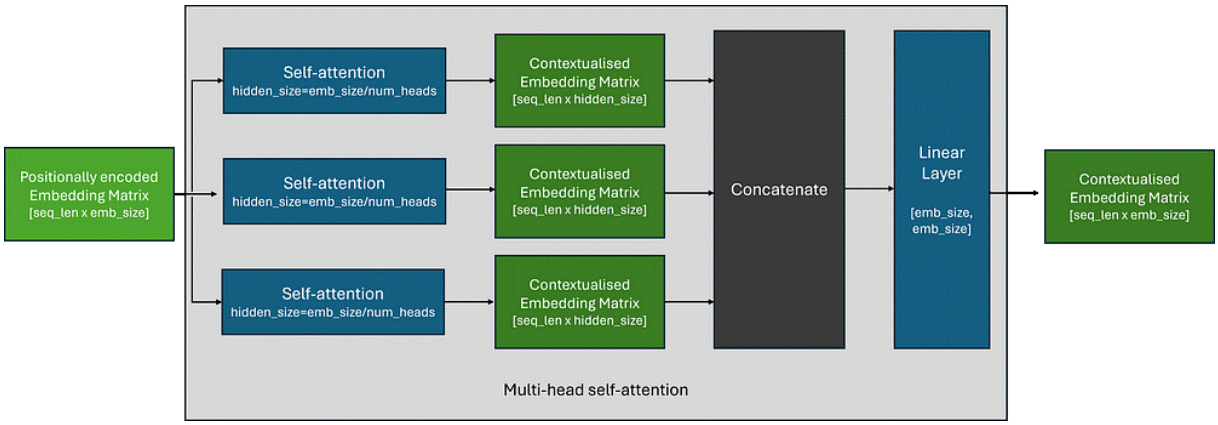

Рисунок 3.13 – Прихований розмір виміру в кожному блоці self-attention

3.5 Склад трансформеру

Відомо, що трансформери складаються з багатьох компонентів, а саме таких як:

– нейромережа з прямим зв'язком (Feedforward Neural Network, FFN): двошарова нейромережа, що застосовується незалежно до кожного ембеддінга токена в пакеті та послідовності. Її завдання – внести в трансформер додаткові досліджувані параметри, що дають змогу переконатися, що контекстуальні ембеддінги розділені та розподілені. У вихідній науковій роботі використано функцію активації GeLU, але компоненти FFN можуть змінюватися залежно від архітектури;

– нормалізація шарів (Layer Normalization): допомагає стабілізувати навчання нейромереж, зокрема трансформерів. Він нормалізує активації для кожної послідовності, не дозволяючи їм стати занадто великими або маленькими під час навчання, що може призводити до проблем на кшталт градієнта, що зникає або вибухає. Така стабільність вкрай важлива для ефективного навчання дуже глибоких моделей-трансформерів;

– зв'язки з пропуском (Skip connections): як і в архітектурі ResNet, для боротьби з проблемою зникаючого градієнта і підвищення стабільності навчання використовуються залишкові підключення.

З плином часу архітектура перша архітектура трансформерів не сильно зазнала змін, але розміщення блоків нормалізації шарів може бути змінено в залежності від того, яка версія архітектури. Ось такий вигляд має початкова версія, яка відома як post-layer norm (рисунок 3.14).

В наші часи зазвичай блоки нормалізації розміщені до self-attention і FFN, всередині зв'язку з пропусками – це називається pre-layer norm (рисунок 3.15).

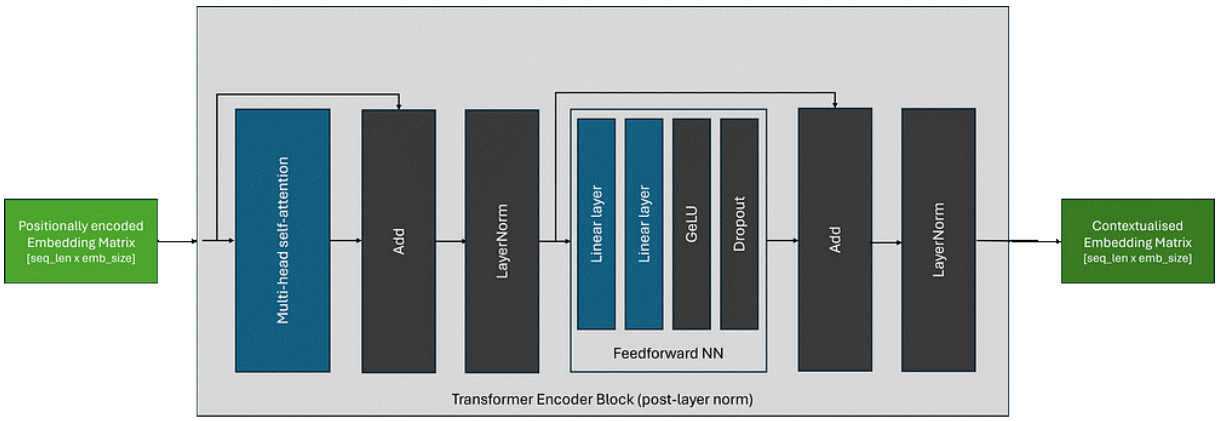

Рисунок 3.14 – Початкова версія

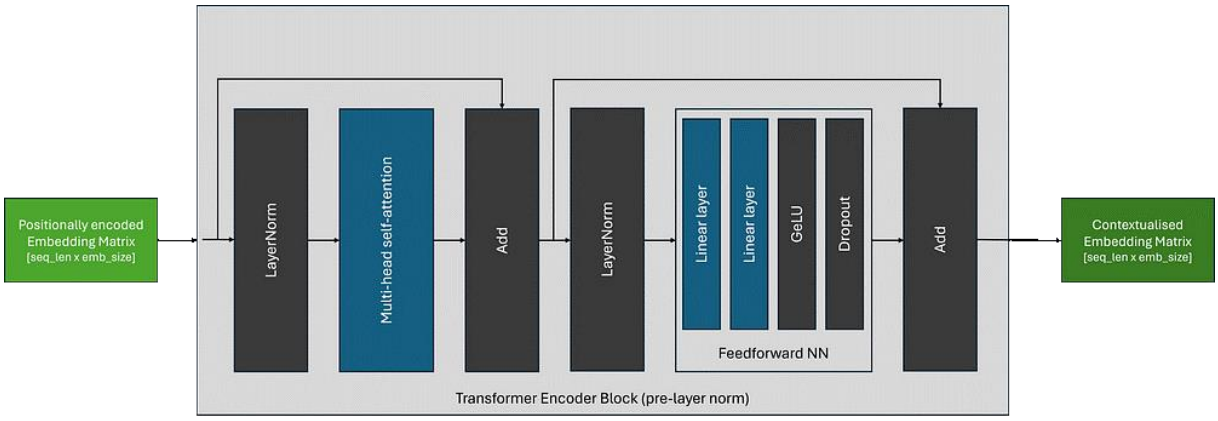

Рисунок 3.15 – Сучасна версія

3.6 Трансформерні архітектури

Існує багато різних трансформерних архітектур, і більшість можна розділити на три типи: енкодери, декодери та енкодери-декодери.

3.6.1 Енкодери

Енкодери синтезовані контекстуальними ембеддінгами, які можуть бути використані у задачах класифікації або розпізнавання іменованих сутностей, адже механізм уваги може обробляти всю послідовність, яка

входить. BERT та його різновиди – є найпопулярнішим видом чистих енкодерів-трансформерів.

Виконавши передачу даних через через один або декілька блоків-трансформерів, ми маємо складну матрицю контекстуалізованих ембеддінгів, що містить відповідно ембеддінги до кожного токену послідовності. Необхідним етапом є виконання передбачення, щоб мати можливість використовувати ці дані у наступних задачах типу класифікації. Зазвичай беруть перший токен і передають через класифікатор, у якому є шари Dropout і Linear. Результат роботи цих шарів можна пропустити через БЛФ для перетворення на ймовірності класів (рисунок 3.16).

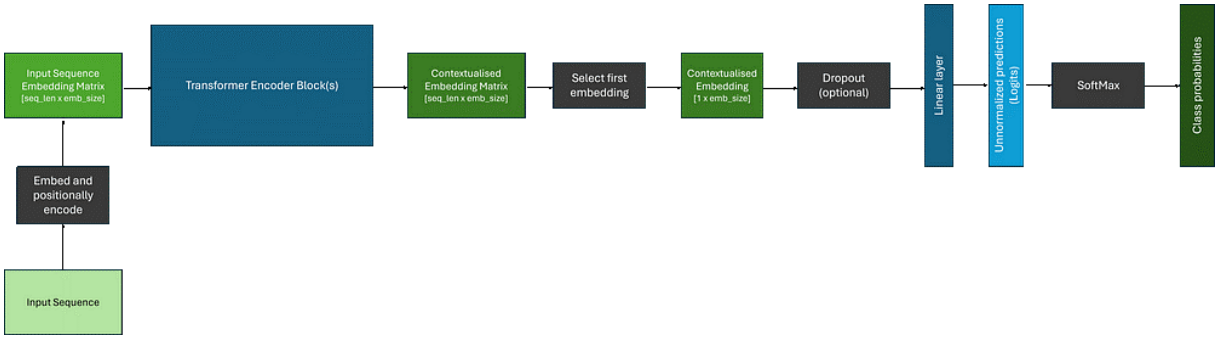

Рисунок 3.16 – Результат роботи шарів

3.6.2 Декодери

Такий тип архітектур майже ідентичний до попереднього (рисунок 3.17), але його головною відмінністю є те, що декодери використовують маскований шар self-attention (рисунок 3.18), тому механізм уваги може приймати тільки поточний і попередні елементи вхідної послідовності. Це означає, що контекстуальні ембеддінги можуть враховувати лише попередній контекст. До одних х популярних моделей-декодерів належить GPT.

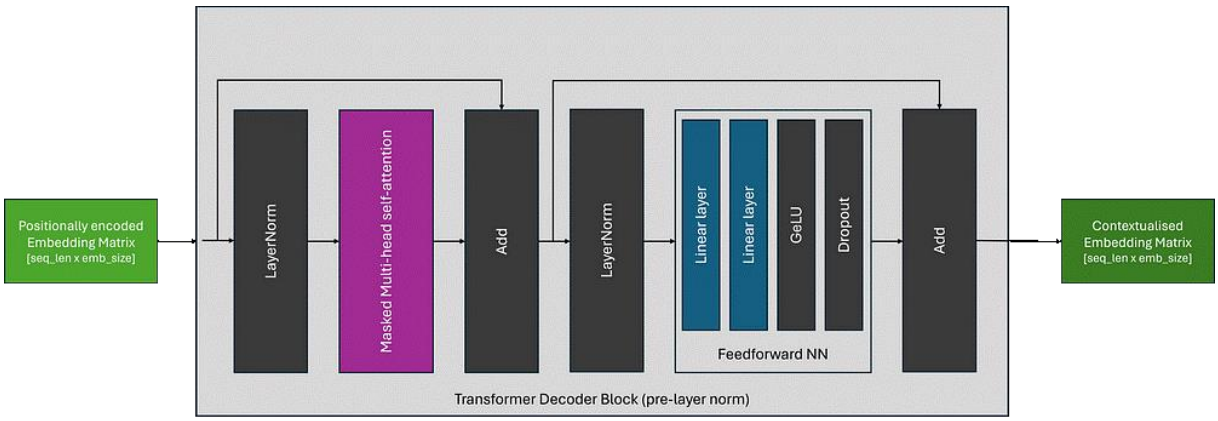

Рисунок 3.17 – Архітектура декодеру

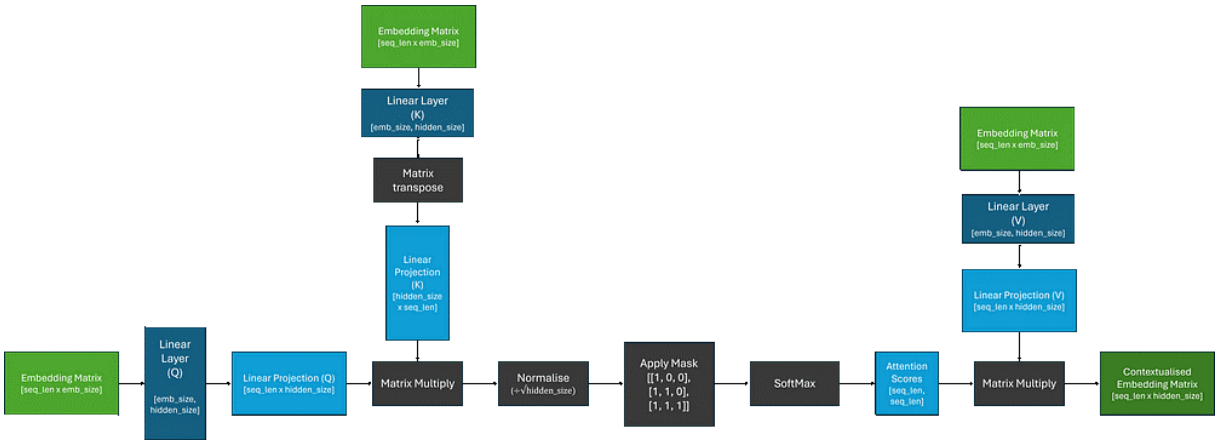

Рисунок 3.18 – Схема self-attention

Зазвичай цього досягають за допомогою маскування оцінок уваги, використовуючи двійкової нижньотрикутної матриці та заміни немаскірованих елементів негативною нескінченністю (потім під час прогону через БЛФ отримують для цих позицій оцінки уваги, що дорівнюють нулю).

Через те, що декодери мають можливість оперувати лише поточною та попередніми позиціями, то вони зазвичай використовуються для авторегресійних завдань, наприклад генерування послідовностей. Але тоді при використанні контекстуальних ембеддінгів необхідні додаткові операції на відміну від енкодерів.

Хоча й декодер створює контекстуальний ембеддінг для кожного токена вхідної послідовності, під час генерування послідовностей ми зазвичай використовуємо ембеддінги, які відповідають фінальним токенам, як вхідні дані для наступних шарів (рисунок 3.19).

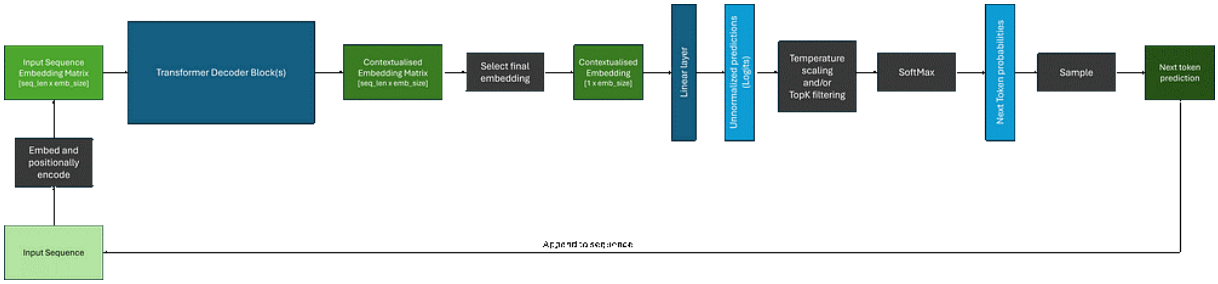

Рисунок 3.19 – Завдання з генерування послідовності

Крім того, після застосування БЛФ без фільтрації ми маємо розподіл ймовірностей у словнику моделі за кожним токеном. Застосовуючи різні фільтри, ми можемо зменшити кількість можливих варінтів, адже може бути досить великий обсяг:

– регулювання температури: температура – це параметр, що застосовується в операції БЛФ. Він впливає на випадковість генерованого тексту, і за допомогою зміни розподілу ймовірностей вихідних слів визначає, наскільки творчим буде результат роботи моделі. Підвищення температури згладжує розподіл і збільшує різноманітність тексту;

– вибірка топ-P: кількість потенційних кандидатів для наступного токена фільтрується на основі заданого порога ймовірності та змінює розподіл імовірностей залежно від кандидатів, які перевищили поріг;

– вибірка топ-K: кількість потенційних кандидатів обмежується кількістю найімовірніших токенів K на основі їхніх логітів або значення ймовірності (залежно від реалізації).

Після зменшення розподілу ймовірностей за потенційними кандидатами для наступного токена, ми можемо зробити з нього вибірку для

отримання передбачення: це вибірка з поліноміального розподілу. Передбачений токен потім додається до вхідної послідовності і подається назад у модель, доки не буде згенеровано бажану кількість токенів або доки модель не синтезує токен зупинки, що позначає кінець послідовності.

3.6.3 Енкодери-декодери

Спершу трансформери були представлені як архітектура для машинного перекладу та використовували як енкодери, так і декодери. За допомогою енкодерів створюється проміжне подання, перш ніж за допомогою декодера перекладати в бажаний формат. Хоча енкодери-декодери сьогодні менш поширені, архітектури як T5 показують, що задачі типу відповідей на запитання, підбиття підсумків та класифікації можна подати у вигляді перетворення послідовності на послідовність і розв'язати за допомогою описаного підходу.

Головна відмінність архітектур типу енкодер-декодер (рисунок 3.20) полягає в тому, що декодер використовує енкодер-декодерну увагу: під час обчислення уваги використовують результат енкодера (K і V) і вхідні дані декодера (Q).

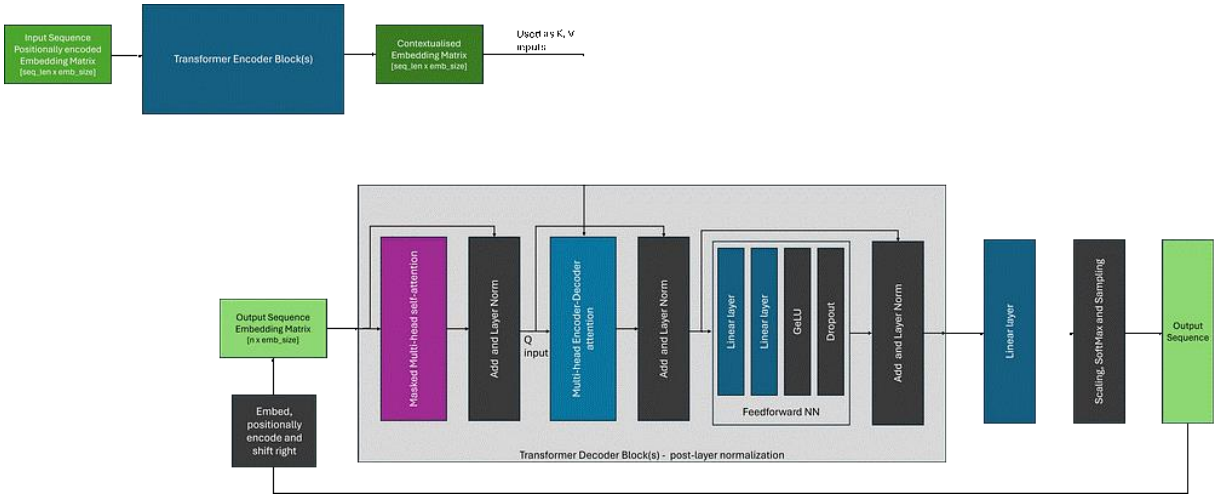

Рисунок 3.20 – Схема архітектури енкодера-декодера

Коли для всіх вхідних даних використовується одна й та сама вхідна матриця ембеддінгів, то при цьому загальний процес синтезу дуже схожий на процес в архітектурах декодерів.

Для спрощення цієї схеми показано варіант post-layer norm з вихідної наукової статті, в якому шар нормалізації розташований після блоків уваги.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

В рамках даної кваліфікаційної роботи було створено нейронну мережу для генерації зображень по тексту, застосовуючи архітектуру трансформерів. Така архітектура складається з п’яти шарів. Також такий вид архітектури добре допомагає прослідковувати взаємозв’язки та послідовності в тексті.

Для реалізації було обрано мову Python, адже вона як раз створена та ідеально підходить для ефективного вирішення поставлених задач. Було використано бібліотеку «Transformers», адже вона розроблена компанією Hugging Face, яка спеціалізується на трансформерах. Вона підтримує багато популярних моделей трансформерів, таких як GPT, BERT, T5 і інші, і дозволяє легко використовувати їх для різних задач, включаючи генерацію зображень. «PyTorch» було використано також, адже це один з найпопулярніших фреймворків для глибокого навчання, розроблений Facebook AI Research. PyTorch відомий своєю гнучкістю і зручністю у використанні, що робить його відмінним вибором для досліджень та розробки нових моделей, включаючи трансформери. «OpenCV» – це бібліотека для обробки зображень, яка використана для попередньої та постобробки зображень у додатку.

NumPy та Pandas: бібліотеки для роботи з числовими даними і таблицями, які часто використовуються для підготовки даних.

Matplotlib та Seaborn: бібліотеки для візуалізації даних, які можуть бути корисні для аналізу результатів моделі.

Тому архітектура додатку виглядає приблизно так:

а) попередня обробка даних:

– використання OpenCV для обробки зображень;

– підготовка та нормалізація даних за допомогою NumPy та Pandas;

б) розробка моделі:

– використання PyTorch для створення моделі трансформера;

– налаштування архітектури моделі та її тренування на GPU для пришвидшення процесу;

в) постобробка:

– обробка вихідних зображень за допомогою OpenCV;

– візуалізація результатів за допомогою Matplotlib;

г) деплоймент:

– використання FastAPI для створення веб-сервісу, який дозволить інтегрувати модель у додаток;

– використання Docker для контейнеризації додатку і спрощення його розгортання.

Також для зручності використання було розроблено інтуїтивний інтерфейс, який допоможе користувачу легко зорієнтуватися у використанні нейронної мережі (рисунок 4.1).

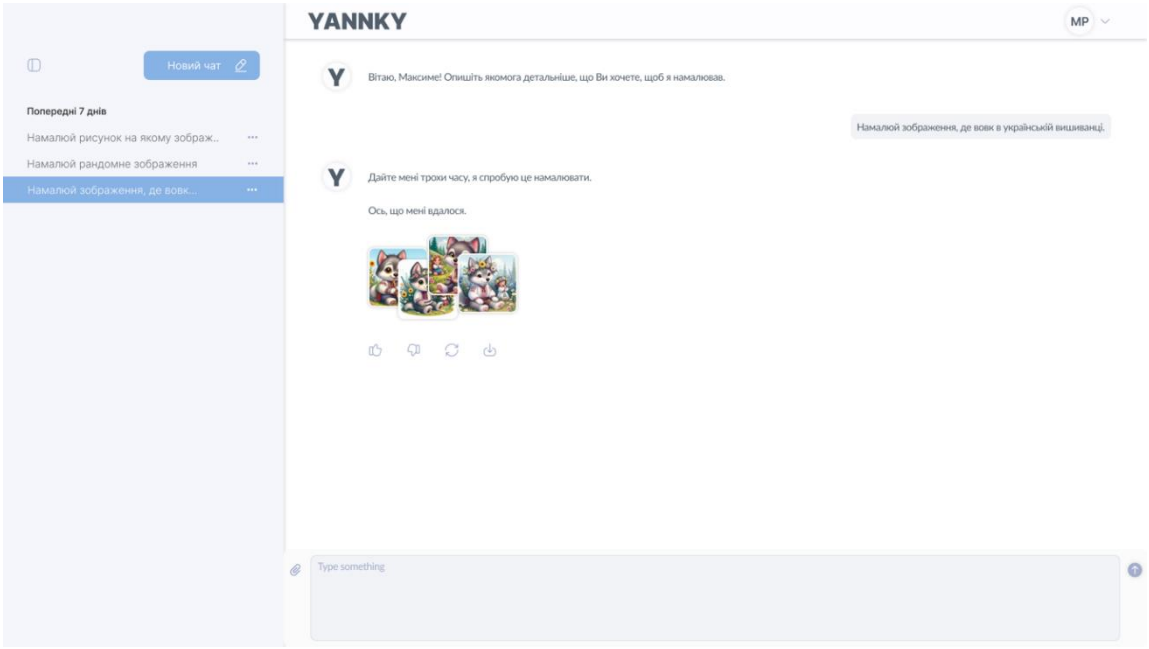

Рисунок 4.1 – Інтерфейс нейронної мережі Yannky

Створена нейронна мережа називається Yannky (Янкі). Це вигадана назва, яка не має певного сенсу, але досить добре запам’ятовується та

завдяки цьому дуже добре позначає те, про яку саме нейронну мережу йде мова.

В розробленій нейромережі Yannky можна створювати декілька потоків чатів для виключення змішування тематик чатів. Так як нейромережа аналізує написане, уловлює сенс та продовжує навчатися на відповідях (підходить чи ні зображення) користувача, то таке розділення на потоки значно покращує роботу мережі та її ефективність, якість результатів значно висока.

Розроблені зображення несуть реалістичний характер (рисунок 4.2), тобто виключено можливість малювання вовка з п’ятьма лапами і т.п. Такий результат було досягнено завдяки якісному навчанню мережі на багатьох різних зображеннях вовка, але в різних позах, різний сілует, колір хутра та багато інших характеристик.

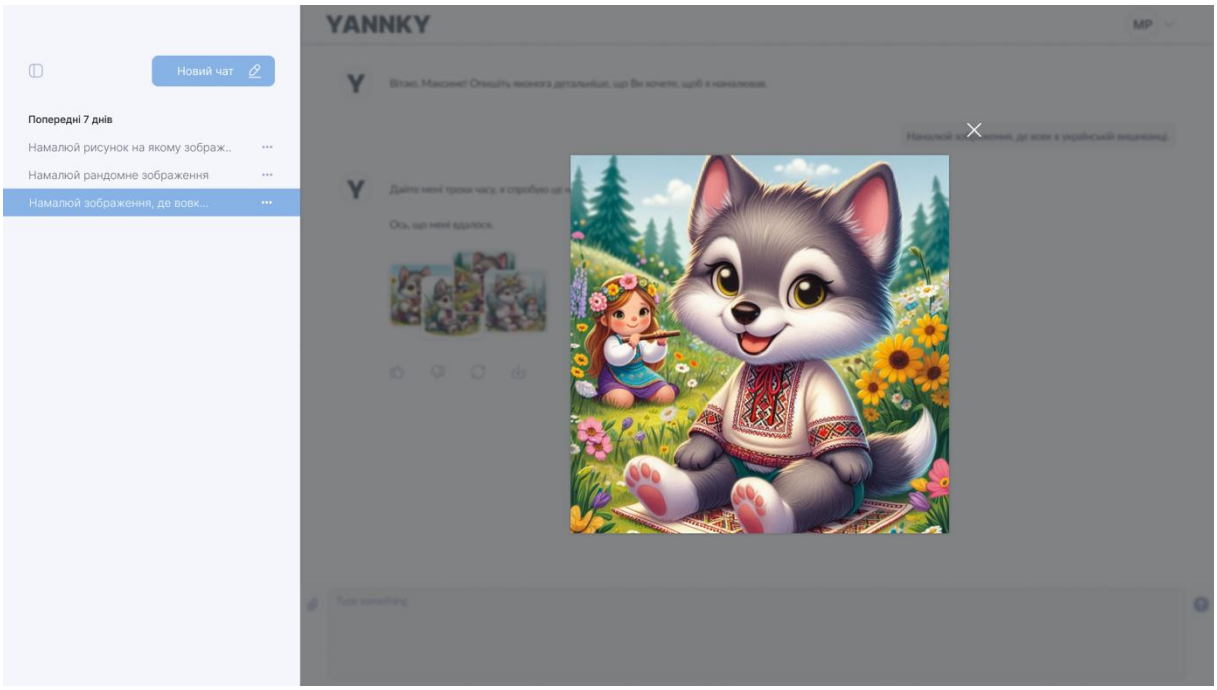

Рисунок 4.2 – Результат виконання запиту користувача

Від користувача необхідно максимально детально описати що саме він хоче, щоб намалювала нейромережа.

Сама ж нейронна мережа навчена на великій кількості пар текстових описів та відповідних зображень, наприклад як Open Images Dataset та MS COCO.

COCO (Common Objects in Context): набір даних, який містить понад 300,000 зображень з понад 80 категоріями об'єктів. Він також включає анотації для задач сегментації та визначення об'єктів.

Open Images Dataset: великий набір зображень від Google, який містить приблизно 9 мільйонів зображень з анотаціями для класифікації, виявлення об'єктів та сегментації.

Наступним кроком є попередня обробка тексту, а саме токенізація текстового опису, тобто перетворення тексту на послідовність. Токенів. Для цього було використано токенайзер GPT-3.

Токенайзер GPT-3 – це компонент, який перетворює текстові дані у формат, зручний для обробки моделлю GPT-3. Токенізація є важливим етапом підготовки даних, де текстові дані розбиваються на менші частини – токени. Це дозволяє моделі обробляти текст на рівні слів, частин слів або символів.

Далі токени перетворено на вектори за допомогою ембеддінгів.

Ембеддінги (embeddings) – це спосіб представлення об'єктів (зазвичай слів або фраз) у вигляді векторів у багатовимірному просторі. Вони використовуються в машинному навчанні, зокрема у завданнях обробки природної мови (NLP) та обробки зображень, щоб перетворити нечислові дані у числовий формат, зручний для використання в нейронних мережах та інших алгоритмах.

Тепер переходимо до попередньої обробки зображень. Система змінює розмір зображень до єдиного формату та перетворює їх у тензори.

Сама ж архітектура моделі складається з текстового енкодеру та декодеру зображення. В текстовому енкодері використано трансформер GPT-3 для кодування текстових описів у контекстуальні ембеддінги. Завдяки цьому модель розуміє семантичне значення тексту.

В декодері зображення було обрано генеративну модель VQ-VAE-2, яка перетворює вихід текстового енкодера у зображення.

VQ-VAE-2 (Vector Quantized Variational AutoEncoder-2) – це вдосконалена версія варіаційного автокодера з векторною квантізацією (VQ-VAE), розроблена для генеративного моделювання зображень з високою роздільною здатністю. Модель VQ-VAE-2 поєднує ідеї квантованих автокодерів та багаторівневої структури, що дозволяє ефективно навчати моделі для генерації якісних зображень.

Для оптимізації моделі було використано алгоритм оптимізації Adam для мінімізації функції втрат і налаштування параметрів моделі.

Також використовуються методи постобробки зображень для поліпшення їх якості, а саме суперрезолюція.

Спочатку YANNKY генерує зображення з низькою роздільною здатністю на основі текстового опису. Це робиться за допомогою трансформера, який перетворює текстові описи у послідовності пікселів або патчів зображень.

На наступному етапі генерується версія зображення з вищою роздільною здатністю. Для цього використовується окрема нейронна мережа, натренована на покращення роздільної здатності низькоякісних зображень. Ця мережа може використовуватися для апскейлінгу (збільшення роздільної здатності) зображень і відновлення деталей.

Було обрано суперрезолюцію, а не усуненя шуму, адже суперрезолюція дозволяє збільшити роздільну здатність зображень, роблячи їх більш детальними і чіткими. Також цей метод допомагає відновити дрібні деталі, які можуть бути втрачені при генерації зображень з низькою роздільною здатністю. Використання суперрезолюції дозволяє досягти кращої візуальної якості зображень, що є важливим для додатків, де важливий зовнішній вигляд і деталізація зображень.

Усунення шуму може використовуватися як додатковий етап обробки, якщо зображення містять значний шум.

Одним з головних етапів є нормалізація шарів (Layer Normalization), яка допомагає стабілізувати навчання нейромереж, зокрема трансформерів. Вона нормалізує активації для кожної послідовності, не дозволяючи їм стати занадто великими або маленькими під час навчання, що може призводити до проблем на кшталт градієнта, що зникає або вибухає. Така стабільність вкрай важлива для ефективного навчання дуже глибоких моделей-трансформерів.

Також використано зв'язки з пропуском (Skip connections) для боротьби з проблемою зникаючого градієнта і підвищення стабільності навчання використовуються залишкові підключення.

ВИСНОВКИ

В ході написання кваліфікаційної магістерської роботи було досліджено архітектури нейронних мереж для виконання задач зі створення зображень за текстовим описом. Було визначено дві найпоширеніші архітектури, що використовуються в популярних додатках для генерації зображень за описом це GAN та трансформери.

В ході дослідження було визначено переваги трансформерів по відношенню до GAN, а саме те, що трансформери вміють обробляти більш великі обсяги тексту, можуть ефективно обробляти послідовності даних паралельно, що дозволяє їм працювати швидше за GAN. Також навчання трансформерів більш стабільне, ніж навчання GAN.

Також було розглянуто які з існуючих відомих додатків використовують яку архітектуру чи їх суміш, але, звичайно, повністю ми не можемо бути впевненими в їхній архітектурі, адже вона не розкривається розробниками.

В результаті проведеного аналізу було визначено, що для виконання задач з генерації зображень за текстовим описом найкраще буде використовувати автокодери з трансформерами.

Було розглянуто детально архітектуру трансформерів та їх роботу. Також було визначено що таке токенізація та її роль у генерації зображень за текстовим описом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Adversarial Training of Variational Auto-encoders for High Fidelity Image Generation / Khan S.H., et al. *arxiv preprint*. 2018. URL: https://arxiv.org/pdf/1804.10323.pdf (date of access: 10.05.2024).

2) Karpathy A., Fei Fei L. Deep Visual-Semantic Alignments for Generating Image Descriptions. URL: https://cs.stanford.edu/people/karpathy/cvpr2015.pdf (date of access: 10.05.2024).

3) Kingma D.P., Welling M. Auto-Encoding Variational Bayes. 2014. URL: https://arxiv.org/abs/1312.6114 (date of access: 10.05.2024).

4) Attention is all you need / Vaswani A. et al. 2017. URL: https://papers.nips.cc/paper/7181-attention-is-all-youneed.pdf (date of access: 10.05.2024).

5) AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks / Xu T. et al. 2017.

6) AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks / Xu T. et al. URL: https://arxiv.org/pdf/1711.10485.pdf (date of access: 10.05.2024).

7) Attention Is All You Need / Vaswani A. et al. 2023. URL: https://arxiv.org/abs/1706.03762 (date of access: 10.05.2024).

8) Recent Advances in Google Translate. *Google Research - Explore Our Latest Research in Science and AI*. URL: https://research.google/blog/recent-advances-in-google-translate/ (date of access: 10.05.2024).

9) How GitHub Copilot is getting better at understanding your code. *The GitHub Blog*. URL: https://github.blog/2023-05-17-how-github-copilot-is-getting-better-at-understanding-your-code/ (date of access: 10.05.2024).

10) Introducing ChatGPT. 2022. URL: https://openai.com/index/chatgpt/ (date of access: 10.05.2024).

11) Transformers. *Hugging Face – The AI community building the future.* URL: https://huggingface.co/docs/transformers/index (date of access: 10.05.2024).

12) Getting Started With Embeddings. *Hugging Face – The AI community building the future.* URL: https://huggingface.co/blog/getting-started-with-embeddings (date of access: 10.05.2024).

13) Roformer: enhanced transformer with rotary position embedding / Su J. et al. 2023. URL: https://arxiv.org/abs/2104.09864 (date of access: 10.05.2024).

14) Layer Normalization / Ba J. L. et al. 2016. URL https://arxiv.org/abs/1607.06450 (date of access: 10.05.2024).

15) Deep Residual Learning for Image Recognition / He K. et al. 2015. URL: https://arxiv.org/abs/1512.03385 (date of access: 10.05.2024).

16) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / Devlin J. et al. 2019. URL: https://arxiv.org/abs/1810.04805 (date of access: 10.05.2024).

17) Language Models are Few-Shot Learners / Brown T.B. et al. 2020. URL: https://arxiv.org/abs/2005.14165 (date of access: 10.05.2024).

18) Token selection strategies: Top-K, Top-P, and Temperature. *Peter Chng*. URL: https://peterchng.com/blog/2023/05/02/token-selection-strategies-top-k-top-p-and-temperature/ (date of access: 10.05.2024).

19) Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer / Raffel C. et al. 2023. URL: https://arxiv.org/abs/1910.10683 (date of access: 10.05.2024).

20) Vidrih M. New AI Breakthrough – Google's Muse: Text-To-Image Generation via Masked Generative Transformers. 2023. URL: https://vidrihmarko.medium.com/new-ai-breakthrough-googles-muse-text-to-image-generation-via-masked-generative-transformers-6bedd5b0cad9 (date of access: 10.05.2024).

21) Fotedar N. A., Wang J. H. Bumblebee: Text-to-Image Generation with Transformers. URL:

https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/15709283.pdf (date of access: 10.05.2024).

22) Дімура М. Нейромережі для малювання та створення зображень – огляд 2023 року. 2023. https://www.site2b.ua/web-blog/nejroseti-dlya-risovaniya-i-sozdaniya-izobrazhenij.html (дата звернення: 10.05.2024).