

MULTILINGUAL IMPROVISED DALL-E IMAGE GENERATION WITH STREAMLIT AND OPENAI GPT-3

CO8811 – PROJECT REPORT

Submitted by

ARCHANA PRIYA S 211420118008

BACHU GURU RISHIKA 211420118009

MUTHU SHRUTHI S 211420118034

in partial fulfillment for the award the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER AND COMMUNICATION ENGINEERING



PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MARCH 2024

BONAFIDE CERTIFICATE

Certified that this project report “**MULTILINGUAL IMPROVISED DALL-E IMAGE GENERATION WITH STREAMLIT AND OPENAI GPT-3**” is the bonafide work of **ARCHANA PRIYA S (211420118008), BACHU GURU RISHIKA (211420118009), MUTHU SHRUTHI S (211420118034) AND** who carried out the project work under my supervision.

SIGNATURE

Dr.B.ANNI PRINCY M.E., Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR,
COMPUTER AND COMMUNICATION
ENGINEERING,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI, POONAMALLEE,
CHENNAI- 600123.

SIGNATURE

Mrs. J. BRINDHA, M.E., (Ph.D.),

SUPERVISOR

ASSISTANT PROFESSOR,
ELECTRONICS AND COMMUNICATION
ENGINEERING,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI, POONAMALLEE,
CHENNAI- 600123.

Certified that the above candidate(s) was/ were examined in the End Semester

Project Viva-Voce Examination held on 26/03/24

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors Tmt. **C.VIJAYARAJESWARI, Dr. C. SAKTHIKUMAR, M.E., Ph.D.,** and **Dr.SARANYASREE SAKTHIKUMAR B.E., M.B.A., Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.,** for his timely concern and encouragement provided to us throughout the course.

We thank our HOD of Computer and Communication Engineering Department, **Dr. B. ANNI PRINCY, M.E., Ph.D.,** Professor, for the support extended throughout the project.

We would like to thank our supervisor, **Mrs. J. BRINDHA, M.E., (Ph.D.).**, Assistant Professor, and all the faculty members of the Department of Computer and Communication Engineering for their advice and suggestions for the successful completion of the project.

ARCHANA PRIYA S
BACHU GURU RISHIKA
MUTHU SHRUTHI S

ABSTRACT

A text-to-image model is a machine learning model which takes an input natural language description and produces an image matching that description. This paper introduces a novel approach to multilingual image generation by integrating DALL-E, OpenAI's advanced image synthesis model, with the natural language processing capabilities of GPT-3. Influencing the power of both models, we present a comprehensive solution for creating diverse and contextually relevant images based on textual prompts in multiple languages. The core architecture involves embedding multilingual textual prompts into the latent space of DALL-E, allowing for the generation of images that align with the specified linguistic context. This project also introduces a Multilingual DALL-E image generator implemented with Streamlit and OpenAI. The system incorporates a spell checker to validate text prompts. Language independence is achieved through a Language Translator, while image storage and display are facilitated by the SQLite3 Python library. High-resolution and image variation are achieved through OpenCV, enhancing the overall capabilities of the generator. Furthermore, we highlight the potential applications of this integrated system in areas such as cross-cultural communication, language learning, and content creation.

TABLE OF CONTENTS

CHAPTER	TITLE	PG NO
	ABSTRACT	i
	LIST OF FIGURES	v
	LIST OF ABBREVIATIONS	vi
1	INTRODUCTION	1
	1.1 OVERVIEW	1
	1.2 PROBLEM STATEMENT	4
	1.3 OBJECTIVES	5
	1.4 METHODOLOGY	5
2	LITERATURE SURVEY	7
3	SYSTEM DESIGN	14
	3.1 PROPOSED SYSTEM ARCHITECTURE DESIGN	15
	3.1.1 INPUT INTERFACE	16
	3.1.2 STREAMLIT INTEGRATION	17
	3.1.3 TEXT PREPROCESSING	17
	3.1.4 OPENAI API INTERACTION	18
	3.1.5 IMAGE POST-PROCESSING	19

	3.1.6 CREATE VARIATION	20
	3.1.7 CONNECTING TO THE DATABASE	21
	3.1.8 IMAGE OUTPUT INTERFACE	22
	3.2 VALIDATION KEY POINT MATCHING	22
	3.3 MODULE DESIGN	25
4	REQUIREMENT SPECIFICATION	29
	4.1 HARDWARE REQUIREMENTS	29
	4.2 SOFTWARE REQUIREMENTS	30
5	ALGORITHMS AND LIBRARIES	34
	5.1 OPENAI.IMAGE.CREATE	34
	5.2 OPENCV'S `CV2.RESIZE` METHOD	35
	5.3 OPENAI GPT-3'S `OPENAI.IMAGE.CREATE_VARIATION` METHOD	36
	5.4 ORB (ORIENTED FAST AND ROTATED BRIEF) FEATURE DETECTOR AND DESCRIPTOR, AND BRUTE-FORCE(BF) MATCHER	37
	5.5 STREAMLIT FOR WEB APPLICATION	38
	5.6 SQLITE	40
6	TESTING AND MAINTENANCE	43
7	RESULT AND DISCUSSION	51

7.1	IMPROVISED DALL-E	51
7.2	MATCHING KEY POINTS	57
8	CONCLUSION AND FUTURE ENHANCEMENT	60
8.1	CONCLUSION	60
8.2	APPLICATIONS	61
8.3	LIMITATIONS	62
8.4	FUTURE ENHANCEMENTS	63
	ANNEXURE	65
	REFERENCES	74

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO
1	Improvized Dall E	15
2	Key Point Matching	22
3	Module Design	25
4	Input Testing	43
5	Multilingual Support	44
6	Image Generation Quality	45
7	Performance Testing	46
8	Error Handling	47
9	Security Testing	48
10	Compatibility Testing	49
11	UAT	50
12	Basic Prompt	51
13	Customization Prompt	52
14	Abstract Prompt	53
15	Storytelling Prompt	53
16	Combination Prompt	54
17	Fantasy Prompt	55
18	Specific Object Prompt	55

LIST OF ABBREVIATIONS

S.NO	ABBREVIATION	EXPANSION
1	OpenCV	Open Source Computer Vision
2	GPT	Generative Pretrained Transformer
3	NLP	Natural Language Processing
4	URL	Uniform Resource Locator
5	API	Application Programming Interface
6	ORB	Oriented Fast And Rotated Brief

CHAPTER 1

INTRODUCTION

In recent years, the combination of creativity with artificial intelligence (AI) has produced impressive developments in a number of fields in recent years. Of these, image generation is one where AI has advanced the most, as models such as DALL-E have shown to be able to produce extremely realistic and contextually relevant images when given textual instructions. But these models have mostly only been used with English, which leaves a large language space unexplored. Within this framework, our work represents a ground-breaking attempt to increase DALL-E's capacity for producing multilingual images. With the help of Streamlit's intuitive interface and OpenAI's GPT-3, a cutting-edge language model, we have created a platform that enables users to create graphics in several languages by just entering textual descriptions.

1.1 Overview

Significant performance gains in text-to-image generative models have been made possible by recent developments in generative modelling. The problem of picture production has been broken down into smaller, discrete tasks in particular by employing sampling-based techniques such as autoregressive generative modelling, a technique where a model predicts the probability distribution of the next element in a sequence based on previous elements, often employed in tasks such as time series prediction and natural language generation, [1,2,20,27,30] and diffusion processes, which involve iteratively applying a series of transformations to a noise vector to generate high-quality images, where each transformation gradually refines the image by diffusing noise information across different spatial scales, [6,11,12,19,22,25] steps that neural networks can also learn it more easily.

Simultaneously, scientists have discovered methods for constructing image generators from stacks of self-attention layers [3, 4, 15]. Through the well-studied scaling properties of transformers, text-to-image models have been able to dependably improve by decoupling picture formation from the implicit spatial biases of convolutions. When combined with a sizable enough dataset, these methods have made it possible to train sizable text-to-image models that can quickly produce imagery that approaches the quality of photos and artwork created by humans. The controllability of image creation systems, which frequently ignore the words, word ordering, or meaning in a given caption, is a notable difficulty in the discipline. These difficulties are referred to as "prompt following."

This issue has been brought up in other works: DALL-E 2 does not impose a constraint where each word has a single meaning, as noted by Rassin et al. (2022). In order to improve it, Saharia et al. (2022) suggest conditioning on pre-trained language models and introducing the Drawbench assessment, which highlights common problems with prompt following. In parallel, Yu et al. (2022) introduce Parti Prompts, their own benchmark, and demonstrate that a different approach to enhance prompt following is to scale autoregressive picture generators. In this study, we suggest a fresh method for handling prompt following: enhancement of the caption.

We speculate that the low quality of the text and image matching of the datasets that the existing text-to-image models were trained on—a problem that has been highlighted in other studies, is a fundamental problem with these models. Publications like Jia et al. (2021). Our plan is to solve this by creating better captions for the photos in our dataset. To achieve this, we first train a strong image captioner that generates precise, in-depth descriptions of images. After that, we use this captioner on our dataset to generate captions that are more thorough. We eventually use our better dataset to train text-to-image algorithms. The idea of

training on synthetic data is not new. Yu et al. (2022), for instance, state that they use this method to train their scaled autoregressive picture generators. Our contribution consists of developing a new system for descriptively producing images and calculating the effects of employing artificial intelligence. When training generative models, use captions. Additionally, we create a baseline performance profile that can be repeated for a set of evaluations that gauge textual prompt and allow it to understand multiple languages or text prompts. The evaluation of DALL-E 3's enhanced prompt following as a consequence of training on extremely descriptive generated captions is the main objective of this work. It excludes information on DALL-E 3 model installation and training. Text-to-image synthesis differs greatly from simple picture synthesis in that it addresses two issues: textual-visual semantic coherence and visual realism. Since the creation of generative models, particularly GANs, the visual reality of picture synthesis has been thoroughly investigated [5].

Certain methods [1], [6] can produce incredibly lifelike visuals that are even hard for humans to discern. Due to the varied written description, the main obstacle for text-to-image synthesis is the visual-textual semantic consistency. While numerous methods are capable of producing reasonably realistic and fine-grained images, particularly for basic datasets like the CUB dataset [28], they seldom ever focus on the multi-level semantic coherence between the associated texts and the created pictures. While more recent methods may be able to create somewhat realistic visuals, they might not be able to produce images that are semantically compatible with the text. In addition, there is a big problem that needs to be solved regarding how to accurately assess the text-to-image synthesis's performance. As previously stated, the goal of text-to-image synthesis is to produce realistic and semantically coherent visuals. Textual-visual congruence and visual reality should therefore be included in the evaluation score. The prevalent assessment metrics, IS [29] and FID [30], primarily take into account visual reality, which is

extensively employed in image creation and image in-painting. As previously stated, text-to-image synthesis prioritises semantic congruence between the image contents and text description as well as image quality. One of the key pillars of this framework is the utilization of OpenAI GPT-3 for text-to-image synthesis. Leveraging the pre-trained models of GPT-3, the platform harnesses the power of machine learning to interpret textual prompts and generate corresponding visual representations. This process involves encoding textual inputs into latent representations, which are then decoded into image space, resulting in the creation of diverse and contextually relevant images. The integration of Streamlit as the user interface layer adds a layer of accessibility and interactivity to the image generation process. Streamlit's intuitive interface empowers users to seamlessly input textual prompts, select desired parameters, and visualize generated images in real-time. Additionally, Streamlit's extensible framework enables developers to customize and enhance the user experience with interactive widgets, visualizations, and other features, further enriching the creative process.

1.2 Problem Statement

The Multilingual Improvised DALL-E Image Generation system with Streamlit and OpenAI GPT-3 addresses the limitations of existing text-to-image synthesis approaches in terms of multilingual support, user interaction, and contextual relevance. Current methods often struggle to generate high-quality images from textual prompts in languages other than English and lack robust mechanisms for user interaction and customization. Additionally, maintaining contextual relevance and coherence between textual inputs and generated images remains a challenge. Therefore, the objective is to develop an advanced system that seamlessly integrates Streamlit for user-friendly interface design and OpenAI GPT-3 for enhanced text understanding and image generation capabilities. This system aims to empower users to generate high-quality, contextually relevant

images from textual prompts in multiple languages, while providing intuitive controls and customization options through Streamlit's interactive interface. By addressing these challenges, the goal is to create a versatile and user-friendly platform for multilingual text-to-image synthesis, catering to diverse user needs and preferences.

1.3 Objectives

1) To improve the textual-visual consistency between the visual content and textual description as well as the visual-visual consistency between the synthesised image and the original image, we propose a dual semantically consistent text-to-image synthesis framework. This plug-and-play approach for any other text-to-image generations can be used for various projects.

2) To determine the similarity between an image and text, we present a novel Key Matching model that can take into account global-level matching, fine-grained local-level matching, and general-level matching. Metric learning, which can move the text and image into an interpretable representation space, optimises this model. Although DALL-E is quite good at creating visuals in response to textual cues, our effort aims to improve upon this skill. We have refined and optimised the model iteratively until it can now generate visuals with improved coherence, relevance, and aesthetic quality in addition to alignment with the input text.

1.4 Methodology

The methodology devised for the creation of the Multilingual Improvised DALL-E Image Generation system with Streamlit and OpenAI GPT-3 is a comprehensive and systematic approach aimed at overcoming the limitations present in current text-to-image synthesis methods. At its core, this methodology involves several key steps, each carefully designed to address specific challenges and optimize the system's performance.

Firstly, the process begins with the collection and preprocessing of data. A diverse dataset comprising text-image pairs in multiple languages is gathered, ensuring that the model is exposed to a wide range of linguistic contexts and visual representations. The textual data undergoes preprocessing steps such as tokenization, lemmatization, and potentially translation to a common language like English, ensuring uniformity and consistency across different languages. Second the model training phase involves the utilization of OpenAI GPT-3, a cutting-edge language model renowned for its advanced text understanding capabilities. The GPT-3 model is trained on the preprocessed dataset, allowing it to learn and internalize the complex relationships between textual prompts and corresponding images.

The integration of the trained GPT-3 model with the Streamlit framework is a pivotal step in the methodology. Streamlit provides a user-friendly interface that enables seamless interaction with the image generation system. Users can input textual prompts, specify language preferences, and customize various parameters through intuitive controls offered by Streamlit's interface design. Throughout the development process, user feedback plays a crucial role in iterative refinement and improvement. Interactive features within the Streamlit interface, such as sliders or dropdown menus, allow users to adjust image attributes and provide real-time feedback on the generated images. This iterative feedback loop ensures that the system evolves in response to user needs and preferences, ultimately enhancing its usability and effectiveness.

Finally, the methodology encompasses evaluation and validation stages to assess the system's performance and reliability. Metrics such as image quality, contextual relevance, and user satisfaction are evaluated to gauge the system's effectiveness across different languages and textual prompts.

CHAPTER 2

LITERATURE SURVEY

TITLE [1]: A unified transformer with inter-task contrastive learning for visual dialogue

AUTHOR: C. Chen et al

YEAR: 2022

DESCRIPTION: While not directly related to DALL-E, BigGAN explores techniques for training large-scale GANs (Generative Adversarial Networks) for generating high-quality images. Understanding GANs is crucial for comprehending some aspects of DALL-E's image generation process. One potential disadvantage of a unified transformer with inter-task contrastive learning for visual dialogue could be its increased complexity and computational demands, potentially leading to longer training times and higher resource requirements.

TITLE [2]: RiFeGAN: Rich feature generation for text-to-image synthesis from prior knowledge

AUTHOR: J. Cheng, F. Wu, Y. Tian, L. Wang, and D. Tao

YEAR: 2022

DESCRIPTION: RiFeGAN, standing for Rich Feature Generation for text-to-image synthesis from prior knowledge, represents a cutting-edge approach to bridging the semantic gap between textual descriptions and image generation. By leveraging prior knowledge and semantic representations embedded within textual input, RiFeGAN excels in producing high-fidelity and contextually rich images. Its architecture emphasizes the extraction and integration of rich features from textual descriptions, allowing for more nuanced and accurate image

synthesis. Through innovative training methodologies and optimization strategies, RiFeGAN addresses challenges such as mode collapse and semantic misalignment, resulting in superior image quality and diversity. RiFeGAN, while enriching text-to-image synthesis with prior knowledge, may face challenges in effectively integrating diverse sources of information and maintaining model interpretability.

TITLE [3]: Bridging multimedia heterogeneity gap via graph representation learning for cross-modal retrieval

AUTHOR: Q. Cheng and X. Gu

YEAR: 2021

DESCRIPTION: "Bridging multimedia heterogeneity gap via graph representation learning for cross-modal retrieval" proposes an innovative approach to address the challenge of heterogeneous multimedia data retrieval. By leveraging graph representation learning techniques, the proposed method aims to capture the complex relationships between different modalities, such as text, images, and videos. This approach facilitates the creation of a unified graph-based representation of multimedia data, enabling effective cross-modal retrieval across heterogeneous sources. Through the integration of graph-based features, the method enhances the retrieval accuracy and robustness, overcoming limitations posed by traditional cross-modal retrieval methods. With its ability to handle diverse types of multimedia data, this approach holds promise for applications ranging from content recommendation systems to multimedia search engines. Bridging multimedia heterogeneity gap via graph representation learning for cross-modal retrieval might encounter limitations in handling large-scale and high-dimensional data representations efficiently.

TITLE [4]: Semantic pre-alignment and ranking learning with unified framework for cross-modal retrieval

AUTHOR: Q. Cheng, Z. Tan, K. Wen, C. Chen, and X. Gu

YEAR: 2022

DESCRIPTION: The proposed method for cross-modal retrieval employs a unified framework that integrates semantic pre-alignment and ranking learning. By pre-aligning the semantic representations of different modalities, such as text and images, the method establishes a common semantic space for more effective retrieval. This pre-alignment process enables the model to capture the underlying semantic relationships between modalities, facilitating accurate retrieval across heterogeneous sources. Additionally, the framework incorporates ranking learning techniques to further refine the retrieval process, optimizing the ranking of relevant items based on their semantic similarity. Through this unified approach, the method achieves superior performance in cross-modal retrieval tasks, enhancing the efficiency and accuracy of multimedia search and recommendation systems. Semantic pre-alignment and ranking learning within a unified framework for cross-modal retrieval could face challenges in scalability and generalization across diverse datasets and modalities.

TITLE [5]: UNITER: Universal image-text representation learning,” in Proc. Eur. Conf. Computer Vision

AUTHOR: Y.-C. Chen et al

YEAR: 2020

DESCRIPTION: "UNITER: Universal image-text representation learning" presents a breakthrough approach to image-text representation learning, showcased at the European Conference on Computer Vision. The method aims to create a unified and versatile representation space for images and text, enabling seamless integration across modalities. By leveraging large-scale pre-training on diverse datasets, UNITER learns rich and contextually relevant representations that capture the intricate relationships between images and text. Through advanced techniques such as contrastive learning and multi-modal fusion, the model achieves state-of-the-art performance in various downstream tasks,

including image-text retrieval and visual question answering. UNITER's universal representation learning capabilities hold promise for advancing applications in image understanding, natural language processing, and multi-modal AI systems. A potential disadvantage of UNITER, a model for universal image-text representation learning, could be its computational complexity, which may limit its scalability and applicability to large-scale datasets and real-time applications.

TITLE [6]: Unsupervised text-to-image synthesis

AUTHOR: Y. Dong, Y. Zhang, L. Ma, Z. Wang, and J. Luo

YEAR: 2021

DESCRIPTION: Unsupervised text-to-image synthesis refers to the process of generating realistic images from textual descriptions without the need for paired image-text data during training. This approach typically involves leveraging generative adversarial networks (GANs) or variational autoencoders (VAEs) to learn a mapping between text embeddings and image features. By training the model solely on unpaired text and image data, unsupervised text-to-image synthesis enables the generation of diverse and contextually relevant images that align with textual descriptions. The main disadvantage of unsupervised text-to-image synthesis methods is their tendency to produce visually inconsistent or semantically ambiguous results due to the lack of explicit supervision, which can hinder the generation of high-quality images.

TITLE [7]: VSE++: Improving visual-semantic embeddings with hard negatives

AUTHOR: F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler

YEAR: 2020

DESCRIPTION: VSE++ (Visual-Semantic Embeddings ++) introduces a novel approach to enhancing visual-semantic embeddings by incorporating hard negatives during training. By mining challenging examples from the dataset that

are incorrectly matched in the embedding space, VSE++ aims to improve the discriminative power of the learned representations. Through this process, the model learns to better differentiate between visually similar but semantically distinct instances, leading to more accurate and meaningful embeddings. A limitation of VSE++ is its susceptibility to increased computational overhead and training complexity, potentially hindering scalability and efficiency in large-scale datasets.

TITLE [8]: Lightweight dynamic conditional GAN with pyramid attention for text-to-image synthesis

AUTHOR: L. Gao, D. Chen, Z. Zhao, J. Shao, and H. T. Shen

YEAR: 2021

DESCRIPTION: The Lightweight Dynamic Conditional GAN with Pyramid Attention (LDC-GAN with PA) presents a novel approach to text-to-image synthesis, aiming to generate high-quality images from textual descriptions efficiently. This model utilizes a lightweight architecture, reducing computational complexity while maintaining performance, making it suitable for resource-constrained environments. By incorporating dynamic conditional information, LDC-GAN with PA adapts its generation process dynamically based on the input text, enhancing the relevance and coherence of the synthesized images. Additionally, the introduction of pyramid attention mechanisms enables the model to focus on salient regions at multiple scales, improving the overall visual quality and detail in the generated images. Experimental evaluations demonstrate that LDC-GAN with PA achieves competitive results in terms of image quality and diversity, showcasing its effectiveness in text-to-image synthesis tasks. A possible drawback of a lightweight dynamic conditional GAN with pyramid attention for text-to-image synthesis is the potential trade-off between model complexity and generation quality, which may affect the fidelity and diversity of generated images.

TITLE [9]: Momentum contrast for unsupervised visual representation learning

AUTHOR: K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick

YEAR: 2020

DESCRIPTION: Momentum contrast (MoCo) is a technique developed for unsupervised visual representation learning, which aims to learn powerful image representations without the need for labeled data. The key idea behind MoCo is to contrast positive pairs of augmented views of the same image against a large queue of negative samples, creating a momentum encoder that maintains a moving average of the model's parameters. By leveraging the momentum encoder, MoCo enables the model to capture semantically meaningful features from unlabeled data, effectively learning robust visual representations. Through contrastive learning, where the positive pairs are encouraged to be closer in representation space than the negative samples, MoCo facilitates the discovery of discriminative features that are invariant to various transformations. Experimental results have shown that MoCo achieves state-of-the-art performance in downstream tasks such as image classification and object detection, demonstrating its effectiveness in unsupervised visual representation learning. Furthermore, MoCo's simplicity and scalability make it a promising approach for leveraging large-scale unlabeled datasets to learn rich and meaningful visual representations. A potential limitation of momentum contrast for unsupervised visual representation learning is its sensitivity to hyperparameters and data augmentation strategies, which can affect the quality and generalization ability of learned representations.

TITLE [10]: Semantic object accuracy for generative text-to-image synthesis

AUTHOR: T. Hinz, S. Heinrich, and S. Wermter

YEAR: 2022

DESCRIPTION: Semantic object accuracy is a crucial metric for evaluating the fidelity of generative text-to-image synthesis models. It measures the model's

ability to faithfully represent the objects described in textual descriptions within the generated images. High semantic object accuracy indicates that the generated images not only capture the overall scene depicted in the text but also accurately depict the individual objects and their attributes. Achieving high semantic object accuracy requires the model to effectively understand and translate the semantic content of the text into visual features, ensuring that the generated images closely match the intended description. This metric plays a significant role in assessing the quality and realism of text-to-image synthesis models, providing valuable insights into their performance and suitability for practical applications. A disadvantage of focusing solely on semantic object accuracy for generative text-to-image synthesis is the potential neglect of overall image quality, leading to visually inconsistent or unrealistic results.

CHAPTER 3

SYSTEM DESIGN

The system design for multilingual improvised DALL-E image generation with Streamlit and OpenAI GPT-3 encompasses a comprehensive architecture aimed at enabling seamless interaction and creative exploration for users worldwide. At its core, the integration leverages the capabilities of Streamlit as a user-friendly web application framework, providing an intuitive interface for users to input textual prompts and visualize generated images in real-time. Coupled with the powerful text-to-image synthesis capabilities of OpenAI GPT-3, the system facilitates multilingual creativity by allowing users to express their ideas in their native languages, transcending linguistic barriers. Through a meticulously crafted pipeline, textual prompts are translated, spell-checked, and fed into the GPT-3 model to generate diverse and contextually relevant images.

At the heart of the system lies a modular and extensible architecture designed to facilitate seamless interaction between users and the AI-powered image generation engine. The architecture comprises several interconnected components, including the user interface layer, the text-to-image synthesis module, language processing modules, and auxiliary functionalities such as spell-checking and translation. The system's design prioritizes accessibility, scalability, and inclusivity, fostering a collaborative ecosystem where users can engage in cross-cultural dialogue and creative expression on a global scale. The above-mentioned features are not available in the existing models that are stated in the previous chapter.

3.1 PROPOSED SYSTEM ARCHITECTURE DESIGN

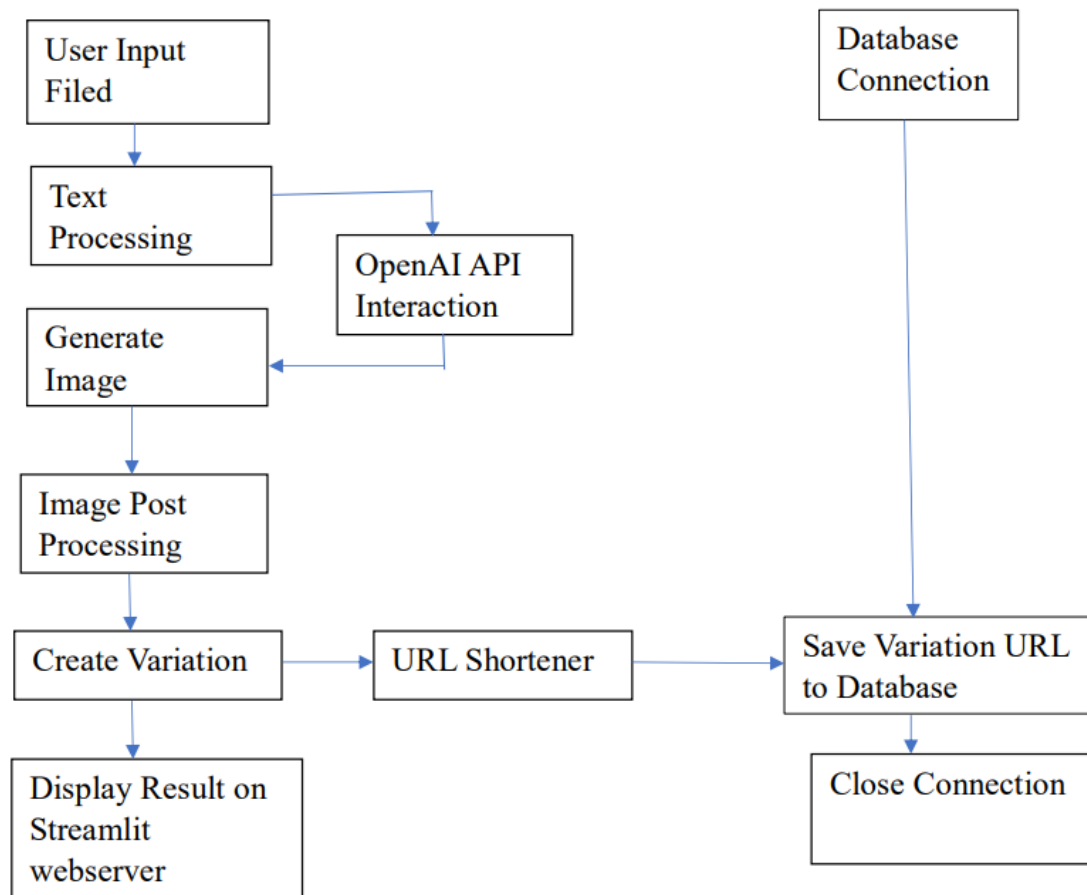


Figure 3.1 Improvised Dall E Work Flow

In addition to the foundational architectural components, as shown in figure 3.1, the Multilingual Improvised DALL-E Image Generation system incorporates advanced features to enhance functionality and user experience. Dynamic resource allocation mechanisms dynamically allocate computing resources based on demand, optimizing performance and cost efficiency. Integration with advanced AI models and algorithms, beyond OpenAI GPT-3, further enriches the image generation process, enabling the synthesis of more diverse and contextually relevant visual outputs.

3.1.1 Input Interface

The application begins with an input screen that allows users to enter prompts or textual descriptions in the language of their choice. Users can interact with the system and start the image generation process through this interface. The Input Interface serves as the initial point of interaction between the user and the multilingual improvised DALL-E image generation program. Its primary function is to provide users with a seamless and intuitive platform to input textual descriptions or prompts in their preferred language. Here's an elaboration on the components and functionality of the Input Interface:

User Input Field

Users can type or paste text descriptions or prompts into the user-friendly input field of the Input Interface. Depending on the user's creative needs, this input box can accept a wide range of input formats, such as brief phrases, sentences, or even extended paragraphs.

Language Selection Option

The Input Interface has an option to pick the preferred input language, making it suitable for users with a variety of linguistic backgrounds. To input textual descriptions in their native language or any other language they feel comfortable with, users can select their favourite language from a list of available languages or a dropdown menu.

Accessibility and User Experience

By utilising simple and intuitive design concepts, the Input Interface places a high priority on accessibility and user experience. For users with varying needs and preferences, it might have features like keyboard shortcuts, accessibility settings, and responsive design to provide a smooth and pleasurable experience across platforms and devices.

3.1.2 Streamlit Integration

Our project involves creating a user-friendly interface using Streamlit that allows users to interact with DALL-E and GPT-3 models to generate images based on natural language descriptions in multiple languages. This integration enables users to input text prompts in their preferred language and receive corresponding images generated by DALL-E.

Web Application Framework

The web application infrastructure that underpins the multilingual, improvised DALL-E picture creation programme is called Streamlit. It offers a simple and effective method for creating interactive web apps with Python scripts without the need for deep knowledge of web development.

Integration with Python Code

With Streamlit's smooth integration with Python code, developers may include data processing techniques, AI models, and other features straight into the web application. The logic of the programme, such as data preprocessing, model inference, and result visualisation, may be defined by developers in Python scripts.

Real Time Updates

When underlying data changes or user interaction occurs, Streamlit makes it possible for the web application interface to be updated in real-time. As they work with the programme, users can see instant feedback and outcomes thanks to its dynamic updating feature, which improves the user experience overall.

3.1.3 Text Preprocessing

The Streamlit app collects the text prompt entered by the user. If the user selects a target language other than English, the text prompt is translated accordingly. Spell checking is performed on the text prompt if the selected language is English.

If spelling errors are found, they are displayed as warnings, and the image generation process is halted.

3.1.4 OpenAI API Interaction

The DALL-E is a sophisticated AI model developed by OpenAI. DALL-E is trained to generate images from textual descriptions, exhibiting a remarkable ability to understand and create visual content based on the provided input.

DALL-E Model

OpenAI created the DALL-E model, a cutting-edge neural network architecture made especially for the purpose of producing visuals from textual descriptions. DALL-E is able to comprehend the semantic meaning of textual descriptions and convert them into equivalent visual representations because to its extensive training dataset of image-text pairs.

Text-to-Image Translation

Textual descriptions or prompts are fed into the DALL-E Image Generation Module, which uses the DALL-E model to translate text to images. DALL-E creates visuals that closely match the semantic content of the input text by applying its learned understanding of the relationship between text and images.

Semantic Understanding

The DALL-E model's capacity to comprehend and interpret textual descriptions' semantic meaning in a nuanced and contextually relevant way is one of its main advantages. DALL-E creates visuals that reflect the text's creative intent and broader conceptual context in addition to depicting the literal interpretation by grasping the underlying concepts and relationships expressed in the input text.

Creative Image Synthesis

DALL-E does inventive picture synthesis as opposed to merely duplicating the visual components that are stated in the input text. It can bring together seemingly unrelated ideas, create original visual compositions, and yield creative interpretations that go beyond the input text's literal interpretation.

3.1.5 Image Post-Processing

Post-processing in the context of DALL-E image generation refers to the additional steps taken after the initial generation of images to refine, enhance, or modify them further. While DALL-E produces high-quality and contextually relevant images directly from textual prompts, post-processing techniques can be employed to improve visual coherence, adjust aesthetics, or add artistic effects.

Image Enhancement

The goal of image enhancement techniques is to raise the generated images' general quality and aesthetic appeal. To improve the clarity and vibrancy of the photographs, this may entail making modifications to brightness, contrast, saturation, and sharpness. To lessen artefacts and enhance visual integrity, methods including tone mapping, denoising, and histogram equalisation might be used.

Color Correction

To achieve desired aesthetic effects, colour correction procedures are applied to the generated photos, adjusting the colour balance and tone. This could entail selective colour editing to highlight or de-emphasize particular hues, white balance adjustment to remove colour casts, and colour grading to create a particular mood or ambiance.

Composition Refinement

The goal of composition refinement approaches is to enhance the generated images' visual composition and arrangement. This could entail increasing the spatial relationships between pieces, cropping or resizing the photographs to better framing and focus, and modifying the location or scale of things inside the image.

Artistic Filters and Effects

The created photographs can be given a creative touch and style by applying artistic filters and effects. Applying painterly effects to imitate conventional art mediums, enhancing visual interest with texture overlays or brushstrokes, or creating a particular artistic style with stylized filters are a few examples of how to do this.

Semantic Editing

While maintaining the coherence of the final image, semantic editing approaches allow users to modify individual objects or elements within the created images. To alter the content of the photographs in accordance with user preferences, this may entail object removal or addition, object replacement or substitution, and object modification (e.g., resizing, rotation).

3.1.6 Create Variation

The generated image URL is used to download the image content from the web. The downloaded image is resized to a fixed resolution using OpenCV. The resized image is converted to PNG format. If the size of the PNG image is less than 4 MB, it is sent to the OpenAI API again to generate variations of the image. The variation URL(s) are obtained from the API response. The original image URL and its variation URL(s) are stored in the SQLite database.

3.1.7 Connecting To The Database

Connecting to the database is a crucial step in the Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3 system. This process involves establishing a connection between the application and the database management system to store and retrieve data efficiently.

Establishing a Connection

The database connection begins with the `connect()` function provided by the `sqlite3` module in Python. The `connect()` function requires the database file name or path as an argument. In this case, the file name is `'image_variations.db'`. The connection object (`conn`) represents a connection to the SQLite database.

Creating a Cursor

After establishing a connection, a cursor object is created using the `cursor()` method on the connection object (`conn`). The cursor acts as a handle or pointer to the database, allowing the execution of SQL commands and retrieval of results.

Executing SQL Commands

Once a cursor is created, SQL commands can be executed using methods like `execute()`, `executemany()`, or `executescript()` provided by the cursor object. In the provided code, the `execute()` method is used to create a table (`image_variations`) if it does not already exist. The SQL command for table creation is passed as an argument to the `execute()` method.

Committing Changes

After executing SQL commands that modify the structure or content of the database (e.g., creating a table, inserting data), changes are committed using the `commit()` method on the connection object (`conn`). Committing changes ensures that modifications are saved permanently to the database.

Closing the Connection

Once all database operations are completed, it's essential to close the database connection using the close() method on the connection object (conn). Closing the connection releases the database resources and ensures that no further operations can be performed on the connection.

3.1.8 Image Output Interface

Finally, the generated images are presented to the user through an output interface. This interface displays the images alongside their corresponding textual descriptions, allowing users to explore and evaluate the results of the image generation process. Users can interact with the interface to view images in different languages, compare multiple generated images, and provide feedback for further refinement.

3.2 Validation Key Point Matching

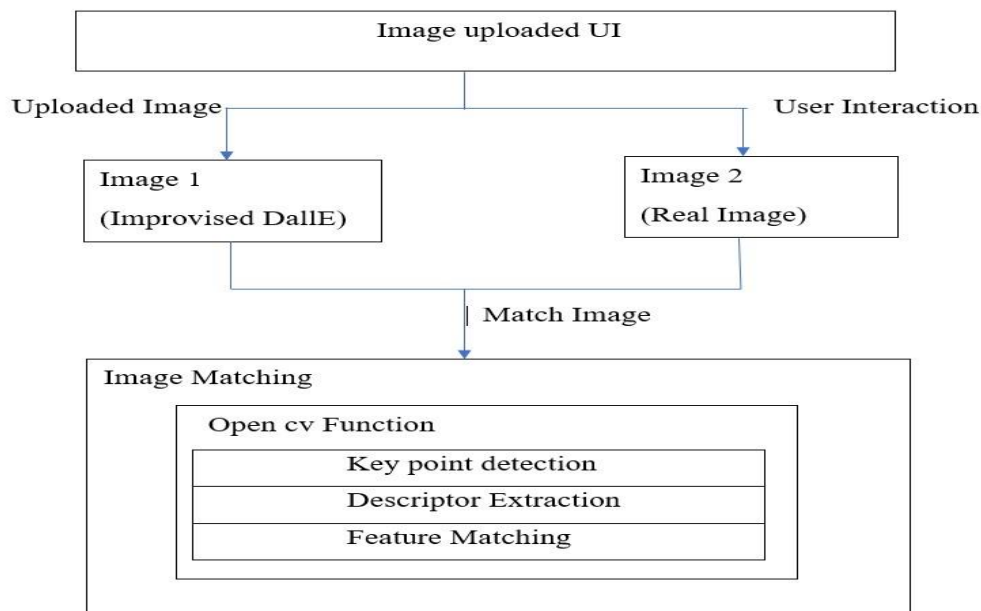


Figure 3.2 Key Point Matching System

The architecture for the Key point Matching Program, figure 3.2, within the Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3 involves several components working together to perform key point detection, feature extraction, matching, and image manipulation.

Streamlit User Interface (UI)

Streamlit provides the user interface for the application, allowing users to interact with the system seamlessly through a web browser. Users input textual descriptions or prompts through Streamlit's input widgets, triggering the image generation process.

OpenAI GPT-3 Integration

In a system utilizing OpenAI's GPT-3 model for image generation, users input textual descriptions of desired images through an application interface. These descriptions are then forwarded to the GPT-3 API, where the model processes them with its advanced natural language understanding capabilities. GPT-3 comprehends the nuances of the input text, extracting key details and context to envision the image described by the user. Leveraging its vast knowledge and linguistic prowess, GPT-3 generates a response that includes a description of the envisioned image. This description is subsequently converted into a format suitable for image generation by the application. Whether through pre-trained image generation models like DALL-E or custom synthesis techniques, the application transforms the textual description into a visual representation. Finally, the generated image is returned to the user, completing the cycle of transforming natural language descriptions into compelling visual content with the aid of GPT-3's sophisticated capabilities.

Key point Matching Program

Once the images are generated by OpenAI GPT-3, they undergo keypoint detection and feature extraction using computer vision techniques. The keypoint

matching program identifies distinctive key points and computes descriptors for each image using algorithms like ORB (Oriented FAST and Rotated BRIEF). It then matches key points between pairs of images using methods such as Brute-Force (BF) matching.

Image Manipulation and Visualization

After keypoint matching, the application may perform further image manipulation or analysis based on the matched key points. It can visualize the matched keypoints and their corresponding matches, allowing users to understand the similarities and differences between the generated images effectively.

Feedback Mechanism

The system may incorporate a feedback mechanism where users can provide input on the quality or relevance of the generated images. Feedback from users can be used to improve the image generation process iteratively.

External Libraries and Tools

The architecture may utilize external libraries such as OpenCV for computer vision tasks, Streamlit for web application development, and OpenAI's Python client for interacting with the GPT-3 API. These tools and libraries provide the necessary functionality for implementing different components of the key point matching program.

Deployment Environment

The entire architecture is deployed on a suitable environment, such as a web server or cloud platform, to make the application accessible to users over the internet. Deployment considerations include scalability, performance, and security to ensure smooth operation of the application in production.

3.3 Module Design

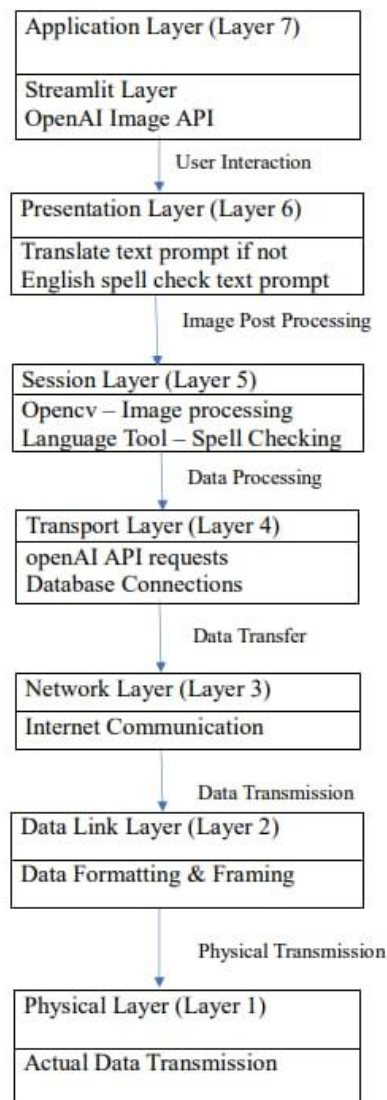


Figure 3.3 Module Design

In the module design, as shown in figure 3.3 of the Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3 system, a modular and extensible architecture is employed to facilitate seamless interaction and integration between components. The design encompasses distinct modules responsible for key functionalities such as user interface management, text-to-image synthesis, language processing, and image variation generation. Each module is designed to encapsulate specific tasks and responsibilities, promoting

code maintainability, scalability, and reusability. Through well-defined interfaces and dependencies, modules communicate and collaborate to orchestrate the image generation process, ensuring a cohesive and efficient workflow. Additionally, the modular design allows for flexibility and adaptability, enabling future enhancements and integrations to be seamlessly incorporated into the system architecture.

Application Layer (Layer 7)

The application logic and user interface layer of the picture creation software serve as the primary point of interaction between users and the system. This layer is responsible for facilitating seamless communication between the user and the underlying technology. Users input textual descriptions of the desired images through an intuitive interface provided by the application. This textual input is then processed and interpreted by the application's logic, which may involve parsing, analysis, and transformation of the user's descriptions. Subsequently, the application leverages various image generation techniques, such as deep learning models or custom algorithms, to convert the textual descriptions into visual representations. Once the images are generated, the application displays them to the user through its interface, allowing for immediate feedback and interaction. This layer plays a crucial role in ensuring a user-friendly experience, enabling users to effortlessly create and explore graphics based on their textual descriptions.

Presentation Layer (Layer 6)

Within the framework of the image creation software, the presentation layer assumes responsibility for data formatting and encoding to facilitate effective transmission and interpretation between different components of the system. This layer acts as an intermediary between the user interface and the underlying processing mechanisms, ensuring seamless communication and interaction.

Specifically, the presentation layer manages activities such as preparing the images created by the application for user display, including tasks like resizing, cropping, or enhancing the visuals for optimal presentation. Moreover, it plays a crucial role in transforming textual descriptions provided by users into a format that can be processed by lower levels of the system, such as natural language understanding algorithms or image generation models. By performing these tasks, the presentation layer ensures that data is appropriately structured and encoded to support efficient processing and interpretation, ultimately enhancing the overall functionality and usability of the image creation software.

Session Layer (Layer 5)

In the Multilingual Improvised DALL-E Image Generation system with Streamlit and OpenAI GPT-3, the Session Layer serves as the intermediary facilitating communication between the user interface and the image generation engine. It manages data exchange, commands, and responses, ensuring seamless interaction for users. Key functions include handling user inputs such as text prompts and language preferences, transmitting them to the back-end for processing, and presenting generated images back to the user. Challenges include optimizing data transmission protocols and maintaining responsiveness, especially with large data volumes. The Session Layer is crucial for enabling real-time interaction, asynchronous requests, and parallel processing, ensuring users receive prompt feedback and visualization of image generation results. Its efficient operation is essential for delivering a smooth user experience while leveraging the capabilities of Streamlit and OpenAI GPT-3 for multilingual and contextually relevant image generation. The communication sessions between the client (user interface) and the server (image generating engine) are managed by the session layer. It creates, keeps, and breaks connections to guarantee dependable and efficient data transfer.

Transport Layer (Layer 4)

End-to-end communication between the client and server is facilitated by the transport layer, which guarantees dependable and effective data transfer. It oversees flow control, error detection, and data segmentation. This layer of the picture production programme may use protocols like User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) to receive generated images and provide text descriptions.

Network Layer (Layer 3)

Data packet forwarding and routing between various network nodes are handled by the network layer. This layer might not be immediately applicable to the picture generation programme if it runs on a single device or local network. However, this layer might be involved in data packet routing over the internet if the programme communicates with distant servers or cloud-based services.

Data Link Layer (Layer 2)

The physical network media is used to transmit data frames, which are controlled by the data link layer. It offers protocols for physical network access as well as error detection and correction techniques. This layer might not be immediately relevant to the image creation programme unless it communicates with hardware or network interfaces to transfer data.

Physical Layer (Layer 1)

The physical network infrastructure, which consists of network interface cards, cables, and connectors, is represented by the physical layer. Bits of raw binary data are transmitted over the network medium by it. This layer is usually abstracted away in the picture creation programme since it does not directly modify actual network components; instead, it communicates with higher-level network protocols.

CHAPTER 4

REQUIREMENT SPECIFICATION

4.1 Hardware Requirement

Hardware requirements for Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3 involve a powerful server or workstation with modern CPUs/GPUs, ample RAM, and storage. These resources support the intensive computational tasks of running AI models like DALL-E and GPT-3, ensuring efficient image generation and smooth interaction via the Streamlit interface.

GPU (Graphics Processing Unit)

Using a GPU can greatly speed up the training and inference procedures, however it's not necessary, especially for large-scale AI models like DALL-E and GPT-3. Deep learning activities are frequently performed on NVIDIA GPUs because of their optimised performance when coupled with frameworks like Streamlit, FastAPI, Flask especially those with CUDA support.

Memory (RAM)

To load and work with big datasets and model parameters, you need enough RAM. For basic operation, a minimum of 16GB of RAM is advised; however, 32GB or more may be preferred for best performance and processing bigger models or datasets.

Storage

To store generated images, training data, and model parameters, there must be sufficient storage space. When working with huge datasets and model files, Solid-

State Drives (SSDs) are preferable over Hard Disc Drives (HDDs) for faster data access and retrieval.

Internet Connectivity

Downloading model checkpoints or updates from the OpenAI API requires a dependable, fast internet connection. A dependable internet connection guarantees smooth communication with outside services and information sources.

Operating System

Numerous operating systems, such as Windows, macOS, and Linux, are compatible with the programme. For deep learning projects, however, Linux distributions are frequently chosen because of their efficiency, adaptability, and GPU driver and framework compatibility.

Additional Considerations

Power supply and cooling needs should be taken into account, particularly when utilising high-performance GPUs. In order to avoid overheating when performing extended computing operations, it is recommended to have a dedicated workstation or server equipped with sufficient ventilation and cooling systems.

4.2 Software Requirements

The software requirements for running a Multilingual Improvised DALL-E Image Generation program with Streamlit and OpenAI GPT-3 encompass various components, including programming languages, libraries, frameworks, and dependencies.

Python

The main programming language utilised to create the image creation software is Python. It was picked for of its abundance of libraries and frameworks, ease of

reading, and simplicity. Because of its simple and intuitive syntax, Python is suitable for both inexperienced and seasoned developers. Python has strong support for deep learning frameworks like TensorFlow and PyTorch, which makes it a popular choice in the fields of artificial intelligence and machine learning. Python is also a great alternative for connecting with Streamlit to design the user interface because of its versatility in web development activities.

Streamlit

Streamlit is a Python library created especially for creating interactive web apps with little to no work, even though it is not a programming language. Streamlit removes the need for complicated HTML, CSS, or JavaScript code by enabling developers to create user interfaces that are responsive and intuitive using straightforward Python scripts. Users may interact with the picture creation programme more easily by creating interactive components like buttons, sliders, input forms, and charts thanks to its declarative syntax and built-in widgets.

OpenAI GPT-3

The OpenAI GPT-3 is a sophisticated language model designed for natural language processing applications. It may be accessed via an API. Although GPT-3 is implemented in multiple programming languages and technologies, developers usually use Python and the OpenAI Python module to communicate and interact with it. Text generating and processing features are made possible by the easy-to-use Python interface that integrates GPT-3 into the picture generation programme.

Additional Libraries and Frameworks

Streamlit is a Python library that simplifies the creation of interactive web applications for data science and machine learning projects. PyShorteners is a Python library that provides utilities for shortening URLs. LanguageTool is a powerful proofreading and grammar checking library for multiple languages. It

helps developers ensure the correctness and clarity of textual content by detecting and suggesting corrections for grammar, punctuation, and spelling errors. Integrating Language Tool enhances the quality and professionalism of text-based applications and content.

Text Processing Libraries

Text processing libraries are helpful for NLP tasks like tokenization and language recognition, such as Natural Language Toolkit (NLTK) or SpaCy. Textual input can be preprocessed using NLTK and SpaCy, two well-liked Python text processing libraries, before being fed to the GPT-3 model.

Other Python Libraries

NumPy: NumPy is essential for numerical computations and array operations, commonly used in deep learning applications. PIL or OpenCV: Libraries for image processing and manipulation, which may be required for handling generated images. Requests: A library for making HTTP requests, useful for interacting with the OpenAI API and other external services.

Python Virtual Environment

In order to maintain reproducibility and manage project dependencies, setting up a Python virtual environment is an important and necessary step. By separating project-specific dependencies from system-wide packages using virtual environments, developers may avoid conflicts and guarantee consistency between the projects and other works.

Integrated Development Environment (IDE)

Integrated development environments (IDEs) offer feature-rich development environments that include project management, code completion, syntax highlighting, and debugging tools. Among Python developers, PyCharm, Visual Studio Code (VS Code), and JetBrains are well-liked IDEs. By providing a

unified workspace for coding, testing, and debugging, integrated development environments (IDEs) expedite the development process and improve code quality and efficiency.

Visual Studio Code

Using Visual Studio for developing a Multilingual Improved DALL-E Image Generation program with Streamlit and OpenAI GPT-3 is feasible, although some considerations need to be addressed due to the specific requirements of the project. Visual Studio is a versatile integrated development environment (IDE) primarily associated with Microsoft technologies, but it supports various programming languages and frameworks, including Python, which is essential for the project.

Version Control System (VCS)

Git is a version control system that is necessary for maintaining project history, managing code changes, and team collaboration. Git repositories can be hosted by platforms such as GitHub, GitLab, or Bitbucket, which facilitate code sharing and teamwork. In order to ensure code integrity and dependability, version control enables developers to work on many features or branches at once, integrate changes, and roll back to earlier versions as needed.

CHAPTER 5

ALGORITHMS AND LIBRARIES

Algorithms and libraries for Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3 refer to the computational methods and software tools utilized in the project. This includes algorithms for image generation, natural language processing, and user interface design, as well as libraries for implementing these algorithms efficiently. Specifically, it involves leveraging algorithms like those used in DALL-E for image generation and OpenAI GPT-3 for natural language understanding. Additionally, libraries such as Streamlit are used to build interactive web applications, enabling users to interact with the system seamlessly. Overall, these algorithms and libraries form the backbone of the project, facilitating the integration of advanced AI technologies for multilingual image generation with user-friendly interfaces.

5.1 `openai.Image.create`

This method is used for generating images based on a given text prompt. The code sends a prompt to GPT-3 requesting a realistic image, and GPT-3 responds with an image URL.

Input Preparation

Gather prompts or textual descriptions from the user or through application input. Make sure the text input is correctly formatted and adheres to any restrictions or standards provided by the OpenAI API.

API Request

Make a POST request to the `openai.Image.create` endpoint, providing the following parameters : (i) `model`: Specify the model ID or name to be used for

image generation. This can include specific versions of the model or variations tailored for image creation. (ii)prompt: Pass the textual descriptions or prompts as input to the model. (iii)max_tokens (Optional): Set the maximum number of tokens to generate in the output text. This parameter helps control the length and specificity of the generated descriptions. (iv) temperature (Optional): Modify the temperature parameter to influence the output's inventiveness and randomness. Text produced at higher temperatures is more varied but may also be less cohesive. (v)stop (Optional): To tell the model when to stop producing text, provide a stopping sequence. When defining precise parameters or restrictions for the output, this can be helpful. (vi)n (Optional): Based on the provided text, indicate how many images to create. For a single input prompt, many images can be requested.

Response Handling

Take note of the image URLs that correspond to the created images in the API response. Depending on the requirements of the application, extract the image URLs from the response and treat them appropriately. When in streaming mode, analyse and show partial results as soon as they are available to give the user feedback in real time while creating images.

Error Handling

To deal with any failures or exceptions that might arise during the API request, provide error handling. To ascertain the reason behind any failures, examine the response status code and error messages that the API has returned. Give the user the proper error messages or fallback options so that unusual occurrences can be handled politely.

5.2 OpenCV's cv2.resize

The generated image is resized to a user-selected resolution (1024x1024 in this case) using OpenCV. This ensures a consistent resolution for further processing.

Input Image Loading

Use the `cv2.imread` function in OpenCV to load the input picture. By doing this, the image is read from the given file path.

Image Resizing

Choose the scaling factor or the desired output image's width and height. To resize the supplied image to the required size or scale, use the `cv2.resize` function. The `cv2.resize` method takes two arguments: the input picture and the target size (width and height) or scale factor. Choose between the `cv2.INTER_LINEAR`, `cv2.INTER_NEAREST`, or `cv2.INTER_CUBIC` interpolation methods when resizing. The interpolation of pixel values during resizing is controlled by this parameter. The scaled image is the output of the `cv2.resize` function.

Output Image Saving or Displaying

If necessary, use OpenCV's `cv2.imwrite` function to save the scaled image to a file. Use the `cv2.imshow` function in OpenCV or any other visualisation tool to view the resized image.

5.3 openai.Image.create_variation

In the Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3 system, the `openai.Image.create_variation` function plays a critical role in generating diverse and contextually relevant image variations based on user input. This function operates by taking an initial image generated from the textual prompt and applying modifications to create variations that explore different visual interpretations and styles.

Input Image Retrieval

Obtain the base image that will serve as the basis for all variants. This picture could be given as binary data, a file path, or a URL.

Variation Parameters

Establish the limits or settings for producing different versions of the supplied image. Specifications like the intended variation degree, the number of variations to generate, or certain attributes to adjust could be included in these parameters.

API Request

Send a request with the base picture and any pertinent parameters to the `openai.picture.create_variation` endpoint. Indicate the desired attributes or adjustments—such as adjustments to colour, style, content, or composition—in order to produce variations.

Variation Generation

Make use of the GPT-3 model's ability to produce several versions of the input image depending on the given parameters. The GPT-3 model could produce a variety of contextually relevant modifications by utilising its comprehension of artistic styles and visual concepts.

5.4 ORB (Oriented FAST and Rotated BRIEF) feature detector and descriptor, and Brute-Force (BF) matcher

The ORB (Oriented FAST and Rotated BRIEF) feature detector and descriptor, along with the Brute-Force (BF) matcher, are widely used in computer vision tasks, such as image matching, object detection, and tracking.

Image Loading

Load the two images that you want to compare using OpenCV's `cv2.imread` method.

Feature Detection and Description

Initialize the ORB detector and descriptor using OpenCV's `cv2.ORB_create()` method. Detect key points and compute descriptors for each image using the ORB

detector and descriptor. This step identifies distinctive points in the images and computes descriptors (feature vectors) for each key point, representing its local neighbourhood.

Matching

Initialize the Brute-Force matcher using OpenCV's `cv2.BFMatcher_create` (`cv2.NORM_HAMMING`, `crossCheck=True`) method. Perform feature matching between the descriptors of the two images using the BF matcher. For each descriptor in the first image, find the best matching descriptor in the second image based on distance metrics such as Hamming distance. Optionally, perform additional filtering or refinement steps on the matched key points to remove outliers or improve the quality of matches.

Visualization

Draw the matched key points between the two images using OpenCV's `cv2.drawMatches` method. Visualize the matched key points and their corresponding matches to assess the quality of the feature matching.

Result Analysis

Analyse the matched key points and their distances to determine the similarity or correspondence between the two images. Compute metrics such as the number of matches, percentage of inliers, or geometric transformation between matched key points. Use the matching results for further tasks such as object recognition, image alignment, or 3D reconstruction.

5.5 Streamlit for Web Application

Streamlit is a popular Python library used for creating interactive web applications for data science and machine learning projects. It allows developers to build powerful web applications with minimal code, enabling rapid prototyping and deployment.

Installation

Install Streamlit using pip: `pip install streamlit`.

Importing Streamlit

Import the Streamlit library at the beginning of your Python script: `import streamlit as st`.

Application Layout

Define the layout and structure of your web application using Streamlit's widgets and layout components. Use Streamlit's layout primitives such as `st.title`, `st.sidebar`, `st.write`, `st.markdown`, etc., to add text, images, charts, and other elements to the application.

User Input

Add interactive widgets to allow users to input data or interact with the application. Streamlit provides a wide range of input widgets such as sliders, text inputs, dropdowns, checkboxes, etc., which can be easily added using functions like `st.slider`, `st.text_input`, `st.selectbox`, `st.checkbox`, etc.

Data Processing

Write the backend logic of your application to process the user input and perform any necessary data processing or computations. Use Python code to manipulate and analyze data, train machine learning models, or perform any other computations required by the application.

Output Display

Display the results or output of the data processing step to the user. Use Streamlit's display functions such as `st.write`, `st.table`, `st.plotly_chart`, `st.pyplot`, etc., to show data visualizations, tables, plots, and other output to the user.

Running the Application

Run the Streamlit application using the command: `streamlit run your_script.py`. This command starts a local web server and opens the web application in your default web browser.

5.6 SQLite

SQLite is a lightweight, self-contained SQL database engine that requires minimal setup and configuration, making it a popular choice for small-scale applications, prototyping, and testing.

Installation

Ensure that the `sqlite3` module is installed with Python. It is usually included by default in Python installations.

Database Connection

Establish a connection to the SQLite database using the `sqlite3.connect()` method. Provide the file path to the SQLite database file as an argument to the `connect()` method. If the database file does not exist, SQLite will create a new database file at the specified path.

Cursor Creation

Create a cursor object using the `cursor()` method of the database connection object. The cursor object allows you to execute SQL queries and fetch results from the database.

Table Creation

Execute SQL queries to create tables in the database using the cursor object. Use the `CREATE TABLE` statement to define the structure of each table, including column names and data types.

Data Manipulation

Perform data manipulation operations such as inserting, updating, deleting, and querying data in the tables. Use SQL INSERT, UPDATE, DELETE, and SELECT statements to manipulate data in the tables.

Transaction Management

Manage database transactions to ensure data integrity and consistency. Use the commit() method of the database connection object to commit changes made within a transaction. Use the rollback() method to roll back changes in case of errors or exceptions.

Error Handling

Implement error handling to handle exceptions that may occur during database operations. Use try-except blocks to catch exceptions raised by SQLite operations and handle them appropriately.

Closing Connection

Close the database connection using the close() method of the database connection object when finished. Closing the connection releases any resources associated with the connection and ensures that changes are saved to the database file.

Streamlit's `st.file_uploader`

Streamlit's `st.file_uploader` function is used to create a file uploader widget in Streamlit web applications. This widget allows users to upload files from their local system to the Streamlit application.

Widget Creation

Use the `st.file_uploader` function to create a file uploader widget in your Streamlit application. Specify the label text and optional parameters such as the file types allowed, maximum file size, and multiple file uploads.

File Upload

Run the Streamlit application and display the file uploader widget to the user. Users can click on the file uploader widget to select and upload files from their local system.

File Processing

Access the uploaded file data returned by the `st.file_uploader` function. Process the uploaded file data as needed, such as reading the file contents, performing data analysis, or displaying the file contents to the user.

User Interaction

Provide feedback to the user about the status of the file upload process, such as displaying a progress bar or success message. Handle user interactions with the uploaded file, such as displaying file content, performing analysis, or triggering further actions based on the uploaded file.

Visualization or Output Display

Visualize or display the uploaded file content or analysis results to the user. Use Streamlit's display functions such as `st.write`, `st.dataframe`, `st.image`, `st.audio`, `st.video`, etc., to show the file content or analysis results in the application interface.

CHAPTER 6

TESTING AND MAINTENANCE

Testing the Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3 involves validating various aspects of the application, ensuring functionality, performance, and user experience.

Input Testing

Input testing, as shown in figure 6.1, for Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3 involves evaluating the effectiveness and accuracy of the system's response to various textual descriptions provided by users. This process typically includes feeding the system with a diverse range of input texts in multiple languages to assess its ability to comprehend and generate corresponding images accurately.

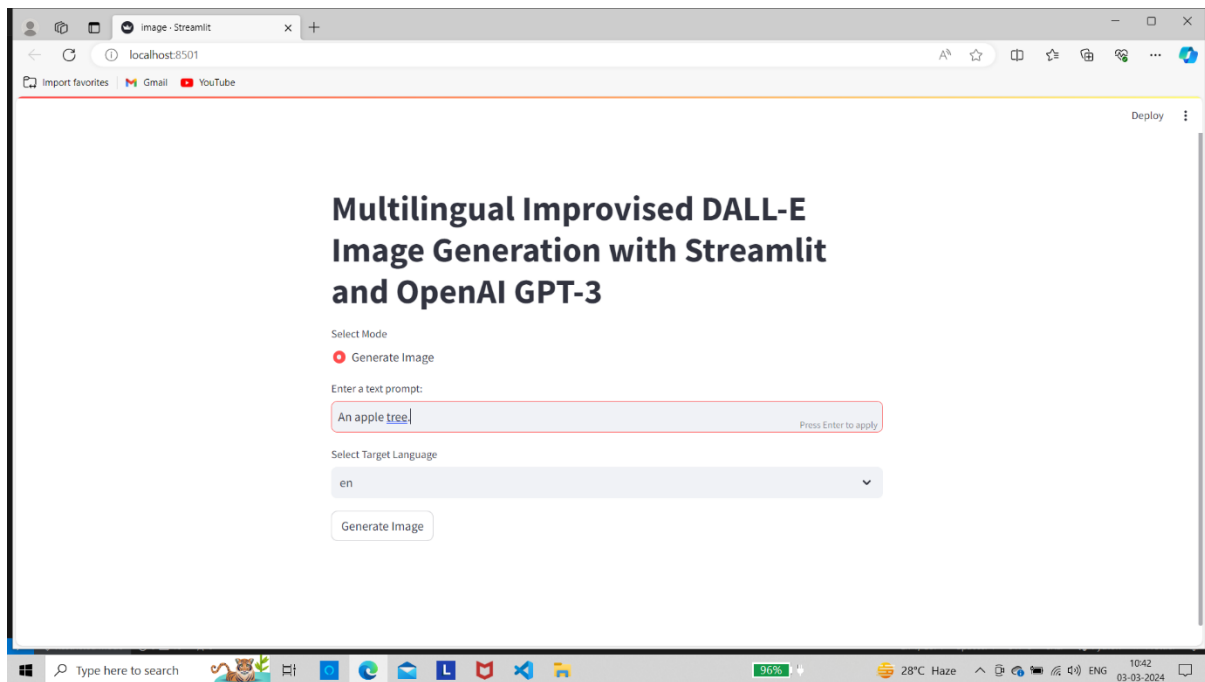
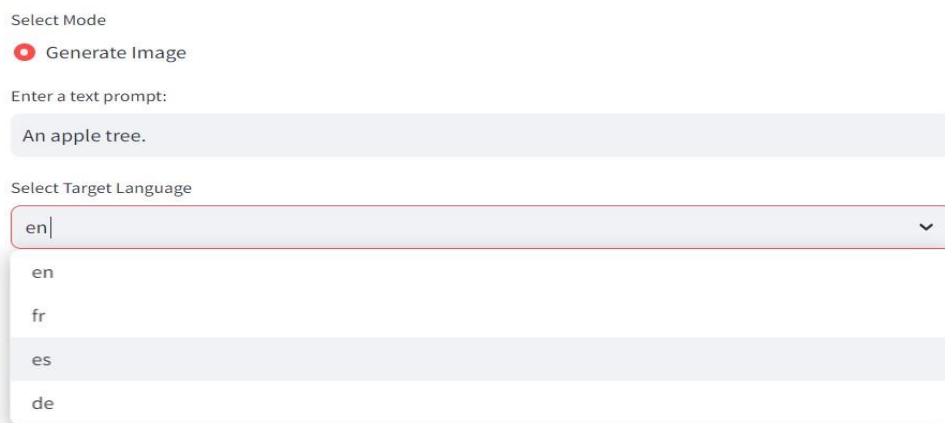


Figure 6.1 Input Testing

Multilingual Support

Multilingual support, as shown in figure 6.2, in the context of Multilingual Improved DALL-E Image Generation with Streamlit and OpenAI GPT-3 refers to the system's capability to comprehend and process textual descriptions provided by users in multiple languages. This feature enables users from diverse linguistic backgrounds to interact with the system seamlessly, generating images based on their descriptions in their preferred language.

Multilingual Improved DALL-E Image Generation with Streamlit and OpenAI GPT-3



The screenshot displays a web interface for generating images. At the top, there is a 'Select Mode' section with a radio button selected for 'Generate Image'. Below this is a text input field labeled 'Enter a text prompt:' containing the text 'An apple tree.'. Underneath the prompt field is a 'Select Target Language' dropdown menu. The dropdown is open, showing a list of language codes: 'en', 'fr', 'es', and 'de'. The 'en' option is currently selected and highlighted.

Figure 6.2 Multilingual Support

Image Generation Quality

Image generation quality in Multilingual Improved DALL-E Image Generation with Streamlit and OpenAI GPT-3 refers to the fidelity, accuracy, and aesthetic appeal of the images produced by the system in response to user-provided textual descriptions. Ensuring high-quality image generation involves various factors, including the ability of the AI models like DALL-E and GPT-3 to accurately understand and interpret the user's input, as well as the effectiveness of the

underlying image generation algorithms in translating textual descriptions into visually coherent and contextually relevant images, as shown in figure 6.3.



Figure 6.3 Image Generation Quality

Performance Testing

This process encompasses various tests aimed at assessing how the system performs under different conditions, including varying loads, input complexities, and language combinations. Performance tests, as shown in figure 6.4, may include measuring the time taken for the system to generate images in response to user inputs, monitoring resource consumption such as CPU and memory usage, and evaluating the system's ability to handle multiple concurrent requests efficiently. Additionally, scalability testing may involve assessing the system's performance as the user base grows or when faced with increased demand for multilingual image generation. Through rigorous performance testing, developers can identify potential bottlenecks, optimize resource utilization, and ensure that the system can deliver reliable and responsive image generation capabilities across different usage scenarios, thereby enhancing overall user satisfaction and system reliability.

PROMPT: "Un paisaje urbano futurista con coches voladores y rascacielos". ("A futuristic cityscape with flying cars and skyscrapers.")-SPANISH

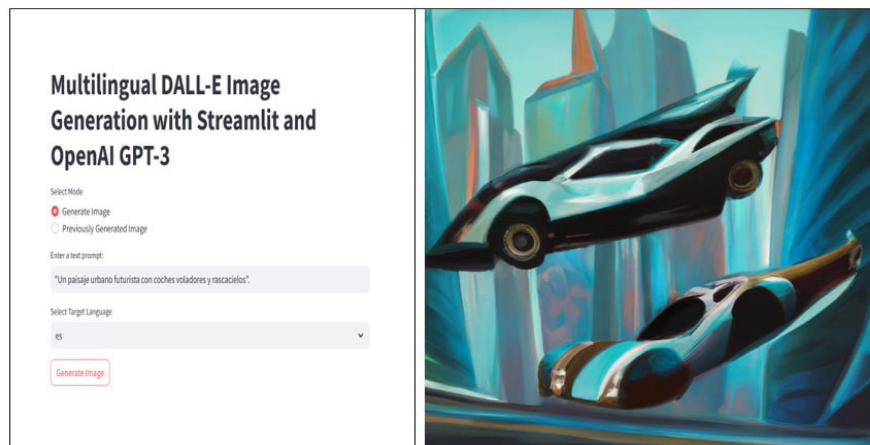


Figure 6.4 Performance Testing

Error Handling

Error handling testing, as shown in figure 6.5, for Multilingual Improved DALL-E Image Generation with Streamlit and OpenAI GPT-3 involves evaluating the system's ability to detect, report, and recover from various types of errors or exceptions that may occur during the image generation process. This testing process encompasses a range of scenarios designed to simulate potential errors, such as invalid user inputs, network connectivity issues, or errors arising from the integration of AI models like DALL-E and GPT-3. Test cases may include deliberately providing incorrect or malformed textual descriptions to assess how the system handles such inputs, as well as simulating unexpected errors in the underlying algorithms or libraries used for image generation. Additionally, error handling testing evaluates the system's responsiveness in providing informative error messages or prompts to users, guiding them on how to rectify the issue or proceed with the image generation process. By conducting thorough error handling testing, developers can ensure that the system remains robust and resilient in the face of unforeseen errors or disruptions, thereby enhancing user trust and confidence in the reliability of the image generation functionality.

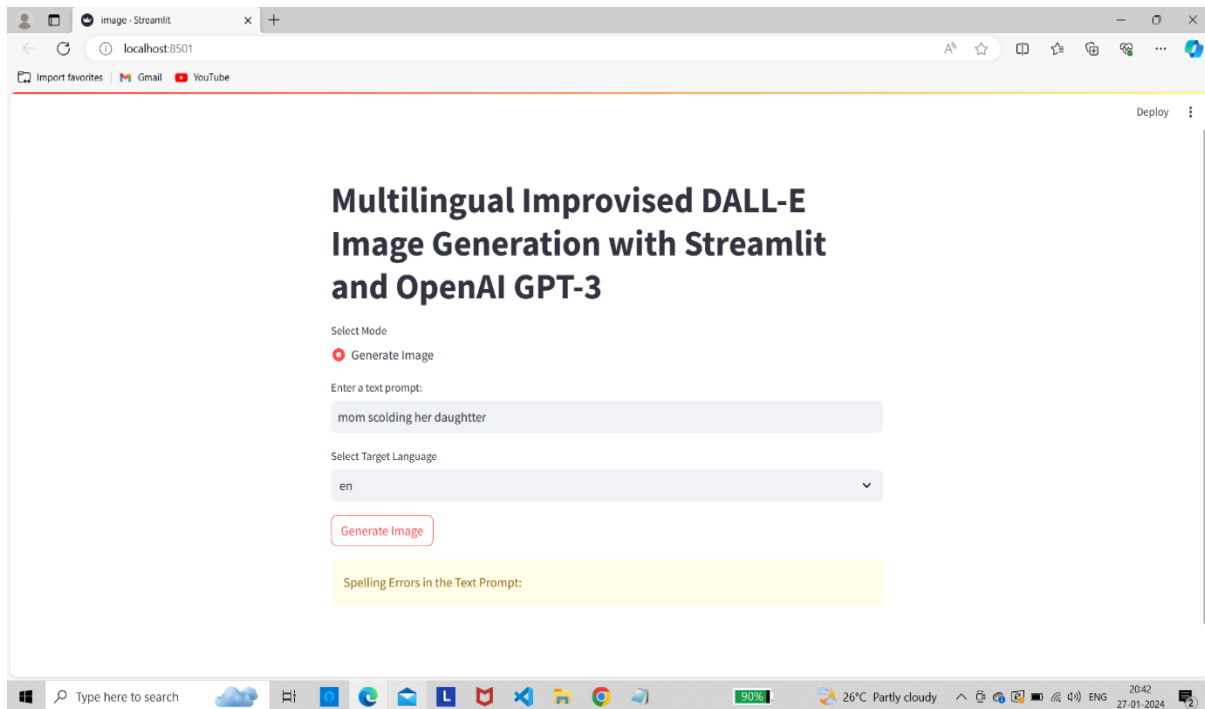


Figure 6.5 Error Handling

Security Testing





Security testing for Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3 involves assessing and enhancing the system's resilience against potential security threats and vulnerabilities. This comprehensive testing process, as shown in figure 6.6, encompasses various aspects, including data privacy, access control, and protection against malicious attacks. Test scenarios may include evaluating the system's handling of sensitive user data, such as textual descriptions or generated images, to ensure compliance with privacy regulations and prevent unauthorized access or leakage. Additionally, security testing involves assessing the robustness of authentication and authorization mechanisms implemented within the system to prevent unauthorized access to sensitive functionalities or resources.

API keys

Your secret API keys are listed below. Please note that we do not display your secret API keys again after you generate them.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically disable any API key that we've found has leaked publicly.

Enable tracking to see usage per API key on the [Usage page](#).

NAME	SECRET KEY	TRACKING ⓘ	CREATED	LAST USED ⓘ	PERMISSIONS	
img_gen	sk-...ej17	Enabled	Dec 22, 2023	Never	All	 
image_generator	sk-...DJTd	Enabled	Dec 22, 2023	Dec 22, 2023	All	 

[+ Create new secret key](#)

Default organization

If you belong to multiple organizations, this setting controls which organization is used by default when making requests with the API keys above.

Personal

Note: You can also specify which organization to use for each API request. See [Authentication](#) to learn more.

Figure 6.6 Security Testing

Compatibility Testing

Compatibility testing for Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3 involves assessing the system's ability to function correctly across different platforms, browsers, and devices. This comprehensive testing process, as shown in figure 6.7, ensures that the application's functionality and user experience remain consistent and reliable regardless of the user's choice of operating system, web browser, or device type. Test scenarios may include verifying the system's compatibility with popular web browsers such as Google Chrome, Mozilla Firefox, and Safari, as well as assessing its performance on various operating systems such as Windows, macOS, and Linux. Additionally, compatibility testing evaluates the system's responsiveness and usability on different device form factors, including desktops, laptops, tablets, and smartphones, to ensure optimal user experience across a wide range of devices.

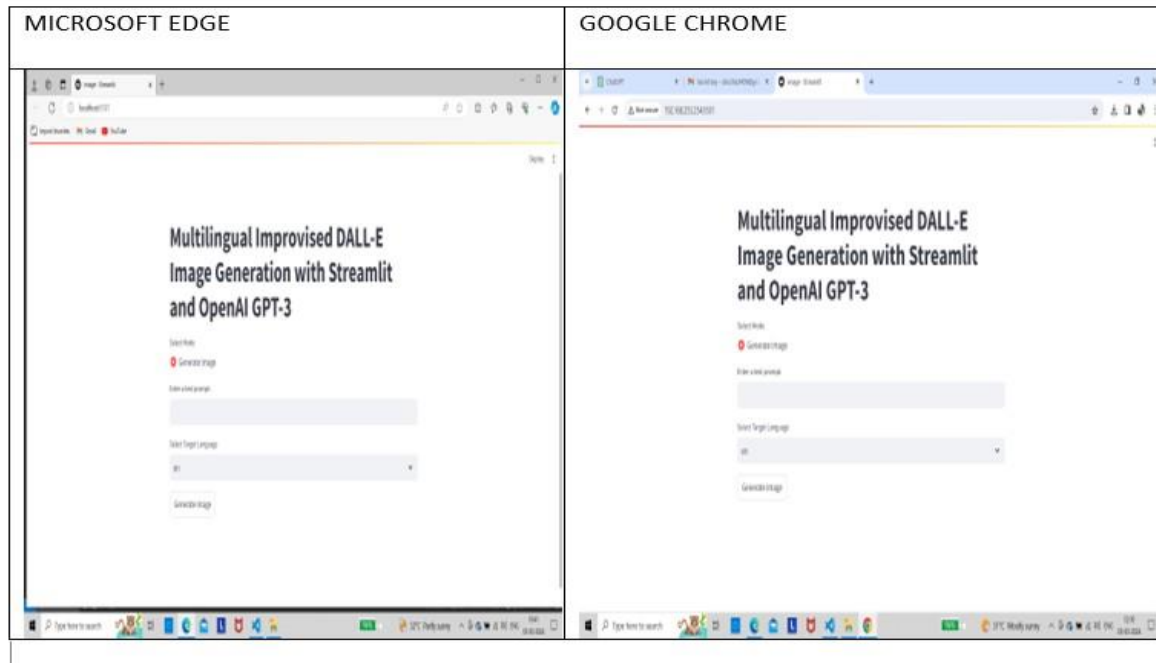


Figure 6.7 Compatibility Testing

User Acceptance Testing (UAT)

User Acceptance Testing (UAT) for Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3 involves evaluating the system's functionality and usability from the perspective of end-users. This testing process allows real users to interact with the application, providing feedback and validating whether it meets their requirements and expectations. Test scenarios, as shown in figure 6.8, may include users from diverse linguistic backgrounds entering textual descriptions in various languages to generate images, assessing the accuracy and relevance of the generated images, and evaluating the overall user interface design and ease of navigation. Additionally, UAT may involve testing the system's multilingual support, ensuring that users can interact seamlessly with the application in their preferred language.

PROMPT: "Imagine a misty forest shrouded in early morning fog, with tall, ancient trees standing like sentinels and sunlight filtering through the dense canopy. Envision a meandering stream flowing peacefully, surrounded by moss-covered rocks and wildflowers. Convey the ethereal beauty and tranquility of this enchanted forest."

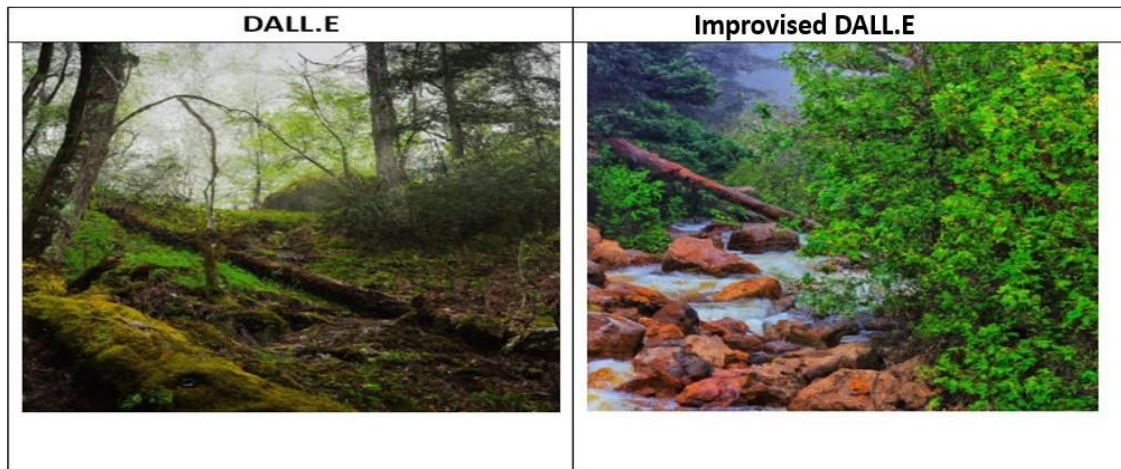


Figure 6.8 UAT

CHAPTER 7

RESULT AND DISCUSSION

The result section provides a visual representation of the system's performance in generating images based on user-provided textual descriptions. In this section, a selection of images generated by the system is presented alongside their corresponding input texts. These images showcase the system's ability to accurately interpret and translate diverse textual descriptions into visually compelling and contextually relevant images. By showcasing the output, this section offers tangible evidence of the system's capabilities in multilingual image generation and provides users with a firsthand look at the results of their interactions with the application.

7.1 IMPROVISED DALL.E

Basic Prompt

- Input: "A peaceful mountain landscape"
- Output: The model generates an image of a peaceful mountain landscape.

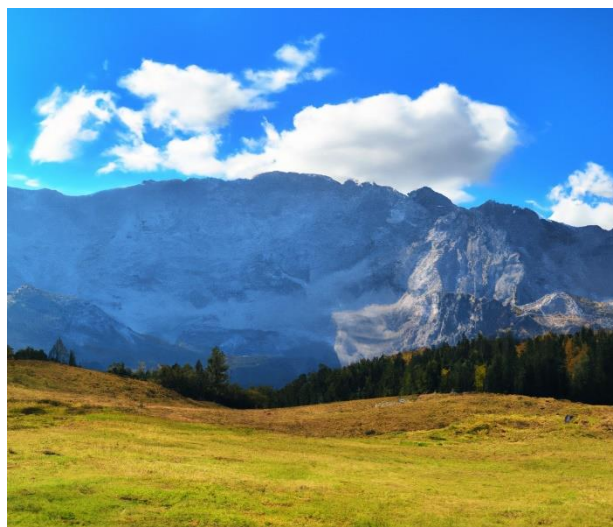


Figure 7.1 Basic Prompt

Upon receiving the input prompt, the model leverages its knowledge of various visual concepts and artistic representations to generate an image. It may consider elements such as mountainous terrain, vegetation, sky, lighting conditions, and other details to create a realistic and visually appealing landscape. The model's ability to understand the context and mood specified in the prompt helps it generate an image that aligns with the user's expectations as shown in figure 7.1.

Customization Prompt

- Input: "A futuristic city skyline with a touch of sci-fi"
- Output: The model generates an image of a futuristic city skyline with sci-fi elements.



Figure 7.2 Customization Prompt

Leveraging the input prompt, the model generates an image that aligns with the specified criteria. It considers various elements such as futuristic skyscrapers, innovative transportation systems, holographic displays, and other sci-fi-inspired features to create a visually compelling city skyline, as shown in figure 7.2.

Abstract Prompt

- Input: "Abstract art with vibrant colors and dynamic shapes"

- Output: The model generates an abstract art piece with vibrant colors and dynamic shapes.



Figure 7.3 Abstract Prompt

Leveraging the input prompt, the model generates an abstract art piece that embodies the specified criteria. It employs various artistic techniques, such as bold brushstrokes, geometric patterns, fluid lines, and contrasting hues, to create a visually stimulating composition, as shown in figure 7.3.

Storytelling Prompt

- Input: "Illustrate a story where a cat goes on a space adventure"
- Output: The model creates an image depicting a cat on a space adventure.



Figure 7.4 Storytelling Prompt

Leveraging the storytelling prompt, the model generates an image that visualizes the narrative scenario described. It depicts the cat protagonist in a space-themed environment, incorporating elements such as stars, spacecraft, and cosmic landscapes to convey the sense of a space adventure, as shown in figure 7.4.

Combination Prompt

- Input: "Combine a forest with a cityscape in a harmonious blend"
- Output: The model generates an image that seamlessly combines elements of a forest and a cityscape.

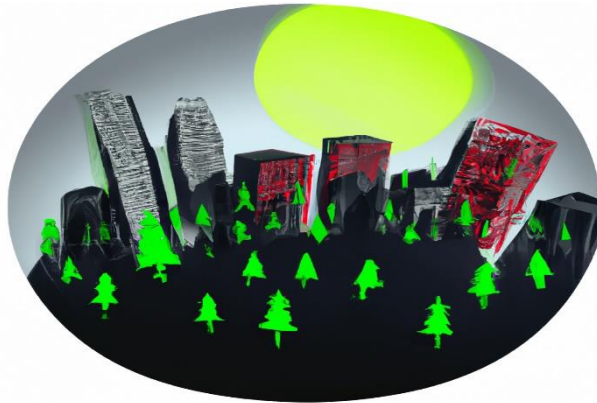


Figure 7.5 Combination Prompt

The model interprets the input prompt and understands the task of seamlessly blending elements of a forest and a cityscape. It recognizes the need to balance the contrasting characteristics of both environments while creating a visually pleasing composition, as shown in figure 7.5.

Fantasy Prompt

- Input: "Create a fantasy world with floating islands and mythical creatures"
- Output: The model generates an image of a fantastical world featuring floating islands and mythical creatures.



Figure 7.6 Fantasy Prompt

Leveraging the fantasy prompt, the model embarks on a creative journey to bring the envisioned fantasy world to life. It draws inspiration from fantasy literature, art, and mythology to design a landscape adorned with floating islands suspended in the sky and populated by an array of mythical creatures. The image may feature fantastical landscapes, ethereal lighting, imaginative architecture, and wondrous flora and fauna, evoking a sense of wonder and enchantment, as shown in figure 7.6.

Specific Object Prompt

- Input: "An old-fashioned bicycle by a vintage lamppost"
- Output: The model generates an image featuring an old-fashioned bicycle next to a vintage lamppost.



Figure 7.7 Specific Object Prompt

Leveraging the specific object prompt, the model endeavors to bring the envisioned scene to life with meticulous attention to detail. It draws upon imagery associated with old-fashioned bicycles and vintage lampposts, considering elements such as design, materials, colors, and textures to imbue the scene with authenticity and atmosphere. The image may feature additional contextual elements such as cobblestone streets, aged architecture, foliage, and ambient lighting to enhance the nostalgic ambiance, as shown in figure 7.7.

Seasonal Prompt

- Input: "A winter scene with a cozy cabin and falling snow"
- Output: The model generates an image of a winter landscape with a cozy cabin and falling snow.



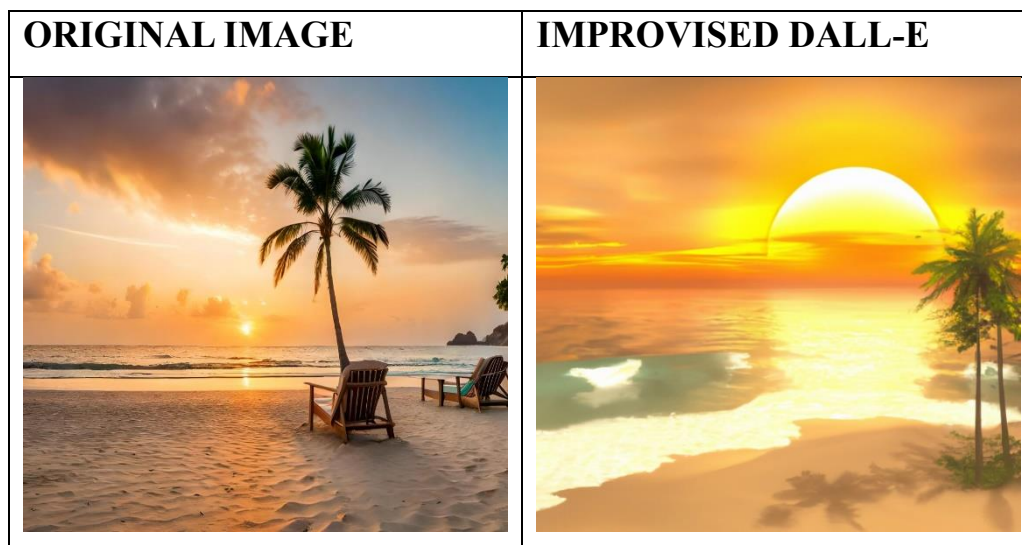
Figure 7.8 Seasonal Prompt

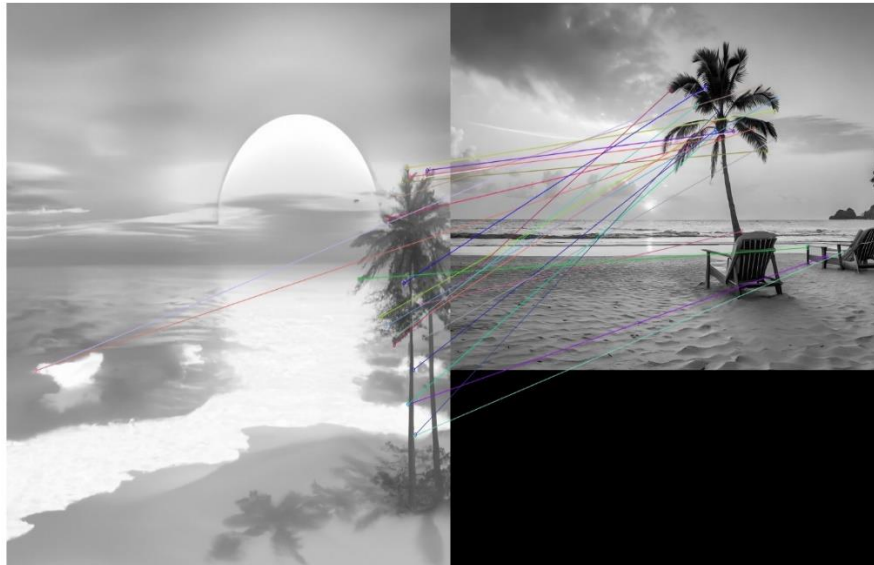
Leveraging the seasonal prompt, the model embarks on a creative endeavor to depict a quintessential winter landscape. It envisions a serene scene featuring a rustic cabin nestled amidst snow-covered trees, with gentle snowflakes cascading from the sky. The image may evoke feelings of coziness, serenity, and nostalgia, inviting viewers to immerse themselves in the tranquil beauty of a winter wonderland, as shown in figure 7.8.

7.2 Matching Key Points

In evaluating the Matching Key Points between generated images and original images in Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3, critical aspects are meticulously examined to assess the fidelity, accuracy, and relevance of the generated visuals. This comparison involves scrutinizing key elements such as content representation, visual coherence, and contextual relevance. Each generated image is evaluated against its original textual description to determine the extent to which the system accurately interprets and translates the input into a visually coherent image. Attention is given to details, such as the presence of objects, scenes, colors, and arrangements depicted in the images, ensuring alignment with the user's intended description.

TEXT PROMPT: A Serene Beach at sunset with palm trees and gentle waves





Number of good matches: 24

Matched keypoints: (924.4801025390625, 298.94403076171875) -> (748.2240600585938, 195.26402282714844)

Matched keypoints: (941.41455078125, 785.89453125) -> (903.0, 453.0)

Matched keypoints: (924.4800415039062, 299.52001953125) -> (747.6000366210938, 196.8000030517578)

Matched keypoints: (879.6000366210938, 391.20001220703125) -> (546.0, 262.8000183105469)

Matched keypoints: (74.64961242675781, 665.6256713867188) -> (588.0, 157.0)

Matched keypoints: (914.4577026367188, 325.55523681640625) -> (729.216064453125, 267.84002685546875)

Matched keypoints: (954.7200317382812, 547.2000122070312) -> (671.8464965820312, 414.7200622558594)

Matched keypoints: (926.2080688476562, 729.216064453125) -> (867.0, 460.0)

Matched keypoints: (925.9200439453125, 728.6400146484375) -> (611.7120971679688, 236.3904266357422)

Matched keypoints: (864.0, 570.0) -> (573.0, 241.0)

Matched keypoints: (963.3600463867188, 316.8000183105469) -> (634.1760864257812, 236.73602294921875)

Matched keypoints: (874.29638671875, 587.641845703125) -> (752.0, 172.0)

Matched keypoints: (939.0, 666.0) -> (611.7120971679688, 230.1696319580078)

Matched keypoints: (960.0000610351562, 609.6000366210938) -> (750.0, 246.00001525878906)

Matched keypoints: (898.800048828125, 604.800048828125) -> (609.140869140625, 214.99090576171875)

Matched keypoints: (66.35520935058594, 667.6992797851562) -> (700.0, 228.0)

Matched keypoints: (943.073486328125, 783.8209838867188) -> (619.2000122070312, 194.40000915527344)

Matched keypoints: (902.0, 353.0) -> (717.0, 150.0)

Matched keypoints: (930.0000610351562, 315.6000061035156) -> (621.6000366210938, 246.00001525878906)

Matched keypoints: (916.800048828125, 508.8000183105469) -> (588.9600219726562, 151.20001220703125)

Matched keypoints: (813.0, 501.0) -> (824.1318359375, 444.3145446777344)

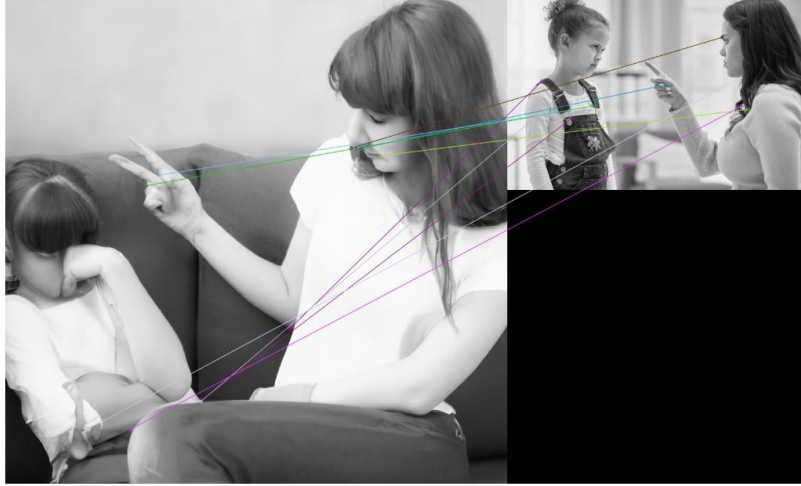
Matched keypoints: (894.0, 621.0) -> (508.81182861328125, 157.66000366210938)

Matched keypoints: (970.56005859375, 303.84002685546875) -> (653.0, 233.0)

Matched keypoints: (900.0000610351562, 603.3600463867188) -> (568.800048828125, 184.8000030517578)

TEXT PROMPT: Mom scolding her daughter





Number of good matches: 31

Matched keypoints: (726.589599609375, 338.4115905761719) -> (322.4863586425781, 209.01893615722656)

Matched keypoints: (765.0, 354.0) -> (490.19915771484375, 258.78533935546875)

Matched keypoints: (288.2304382324219, 414.7200622558594) -> (174.0, 252.00001525878906)

Matched keypoints: (400.8000183105469, 852.0000610351562) -> (132.71041870117188, 275.7888488769531)

Matched keypoints: (579.2810668945312, 707.6784057617188) -> (70.50241088867188, 201.13922119140625)

Matched keypoints: (194.91842651367188, 897.868896484375) -> (475.20001220703125, 194.40000915527344)

Matched keypoints: (312.0, 393.6000061035156) -> (127.0, 261.0)

Matched keypoints: (705.0241088867188, 337.9968566894531) -> (437.0, 112.0)

Matched keypoints: (398.4000244140625, 855.6000366210938) -> (116.4000015258789, 224.40000915527344)

Matched keypoints: (253.44000244140625, 912.9600219726562) -> (480.1463623046875, 250.82272338867188)

Matched keypoints: (857.0880737304688, 480.384033203125) -> (472.9800109863281, 232.9068145751953)

Matched keypoints: (188.11703491210938, 597.1969604492188) -> (77.0, 363.0)

Matched keypoints: (770.6880493164062, 397.4400329589844) -> (492.0000305175781, 237.60000610351562)

Matched keypoints: (770.0, 396.0) -> (107.49545288085938, 262.76666259765625)

Matched keypoints: (164.0, 607.0) -> (116.45340728759766, 283.6685485839844)

Matched keypoints: (895.2000122070312, 852.0000610351562) -> (79.0, 362.0)

Matched keypoints: (163.20001220703125, 607.2000122070312) -> (118.24500274658203, 290.23773193359375)

Matched keypoints: (854.4000244140625, 475.20001220703125) -> (472.7809143066406, 194.08900451660156)

Matched keypoints: (254.01602172851562, 914.112060546875) -> (474.7715759277344, 244.8507537841797)

Matched keypoints: (146.31326293945312, 618.098876953125) -> (110.48143768310547, 235.89279174804688)

Matched keypoints: (322.0, 468.0) -> (69.12000274658203, 203.04000854492188)

Matched keypoints: (193.20001220703125, 903.6000366210938) -> (181.64739990234375, 311.0400695800781)

Matched keypoints: (288.5760192871094, 416.4480285644531) -> (122.42537689208984, 256.794677734375)

Matched keypoints: (711.0, 364.0) -> (465.0, 263.0)

Matched keypoints: (705.0240478515625, 336.96002197265625) -> (77.0, 361.0)

Matched keypoints: (189.11236572265625, 903.2603759765625) -> (164.16001892089844, 330.04803466796875)

Matched keypoints: (379.8172912597656, 827.7150268554688) -> (334.4302978515625, 259.7806701660156)

Matched keypoints: (499.0, 853.0) -> (106.0, 373.0)

Matched keypoints: (194.0, 598.0) -> (84.60289764404297, 196.57733154296875)

Matched keypoints: (56.160003662109375, 901.4400634765625) -> (81.60000610351562, 74.4000015258789)

Matched keypoints: (727.3859252929688, 340.40228271484375) -> (441.0, 111.0)

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 Conclusion

The Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3 offers a powerful solution for generating images based on textual descriptions in multiple languages. Through the integration of Streamlit's user-friendly interface and OpenAI GPT-3's advanced natural language processing capabilities, users can effortlessly create diverse and contextually relevant images with just a few clicks. The application provides an intuitive platform for users to express their ideas and concepts through text and transform them into visually stunning images, thereby facilitating creativity, communication, and collaboration across linguistic barriers.

Multilingual Support

The application enables users to generate images from textual descriptions in multiple languages, fostering inclusivity and accessibility for users worldwide.

Ease of Use

With Streamlit's intuitive interface, users can interact with the application seamlessly, providing input text and viewing generated images effortlessly.

Quality Outputs

Leveraging OpenAI GPT-3's state-of-the-art language understanding and image generation capabilities, the application produces high-quality and contextually relevant images that reflect the input text accurately.

8.2 Applications

The Multilingual Improvised DALL-E Image Generation system with Streamlit and OpenAI GPT-3 has diverse applications across various domains. Here are some potential applications:

Content Creation and Marketing

Content creators, marketers, and social media managers can use the system to generate visually appealing images for online content, social media posts, advertisements, and marketing campaigns. The multilingual capability allows them to target diverse audiences globally.

E-commerce and Product Visualization

E-commerce platforms can utilize the system to generate product images from textual descriptions, enhancing the shopping experience for customers. This enables them to visualize products in different languages and variations before making a purchase.

Education and E-learning

Educators and e-learning platforms can create engaging visual aids, diagrams, and illustrations for educational materials and online courses. The system can generate images corresponding to course content in multiple languages, catering to a global audience.

Multilingual Chatbots and Virtual Assistants

Chatbots and virtual assistants can leverage the system to provide rich multimedia responses to user queries in multiple languages. This enhances user engagement and improves the overall user experience in various applications, including customer support and information retrieval.

Art and Design

Artists, designers, and creative professionals can use the system as a tool for inspiration and exploration in multilingual creative projects. It allows them to quickly generate visual representations of ideas and concepts expressed in different languages, facilitating cross-cultural collaboration and expression.

Medical Imaging and Healthcare

Healthcare professionals can employ the system to generate medical illustrations and diagrams from textual descriptions in different languages. This aids in medical education, patient communication, and the creation of multilingual healthcare resources.

Localization and Globalization

Companies and organizations operating in multiple countries can utilize the system to create localized visual content for their products, services, and marketing campaigns. This ensures cultural relevance and resonance with diverse target audiences worldwide.

8.3 Limitations

While the Multilingual Improvised DALL-E Image Generation system with Streamlit and OpenAI GPT-3 offers numerous benefits, it also has some limitations:

Interpretability and Transparency

The inner workings of the AI models used in the system, particularly OpenAI GPT-3, may lack transparency and interpretability, making it challenging to understand how decisions are made and potentially hindering trust and accountability.

Language Specificity

Despite being multilingual, the system's performance may vary across different languages. It may struggle with languages for which it has limited training data or linguistic nuances that are not well-represented in the training corpus.

Complexity and Computational Resources

Implementing and deploying a system with Streamlit and OpenAI GPT-3 can be computationally intensive and resource-demanding, requiring robust infrastructure and sufficient computational resources.

Limited Fine-Grained Control

While users can provide textual prompts and customize some parameters, the system may lack fine-grained control over specific image attributes or styles, limiting its suitability for certain use cases requiring precise customization.

8.4 Future Enhancements

Explore advanced image generation techniques and models to enhance the diversity, realism, and creativity of the generated images further.

Enhanced Multimodal Capabilities

Integrate additional modalities such as audio, video, or structured data to create more diverse and expressive multimedia content.

Customization Options

Provide users with more customization options to control various aspects of the image generation process, such as style, color palette, composition, and image size.

Collaborative Features

Implement collaborative features that allow users to share, edit, and remix generated images collaboratively, promoting teamwork and idea exchange.

Performance Optimization

Optimize the application's performance, scalability, and resource utilization to handle larger datasets, concurrent users, and complex queries efficiently.

Feedback Mechanisms

Incorporate feedback mechanisms to gather user input, preferences, and suggestions for continuous improvement and iteration of the application.

ANNEXURE

SAMPLE CODE

IMPROVISED DALL.E IMAGE GENERATION

```
#pip install streamlit openai pyshorteners requests translate opencv-python
numpy language-tool-python

import streamlit as st

import openai

import pyshorteners

from PIL import Image

import io

import requests

import sqlite3

from translate import Translator

import cv2

import numpy as np

from language_tool_python import LanguageTool

# Set your OpenAI API key

openai.api_key = "sk-
HD4dmXRLEG0gPQupqqNkT3B1bkFJADOXxrZ1ihUpZkY2q4Lw"

# Create or connect to the SQLite database
```

```

conn = sqlite3.connect('image_variations.db')

cursor = conn.cursor()

# Create a table to store image variations

cursor.execute("""

    CREATE TABLE IF NOT EXISTS image_variations (

        id INTEGER PRIMARY KEY,

        original_image_url TEXT,

        variation_url TEXT

    )

""")

conn.commit()

# Streamlit app title

st.title("Multilingual Improvised DALL-E Image Generation with Streamlit and OpenAI GPT-3")

# Add a mode selection radio button

mode = st.radio("Select Mode", ["Generate Image"])

# To generate shortened URL

def generate_short_url(image_url):

    return pyshorteners.Shortener().tinyurl.short(image_url)

# To generate AI image

def generate_image():

    text_prompt = st.text_input("Enter a text prompt:")

```

```

target_language = st.selectbox("Select Target Language", ["en", "fr", "es",
"de"], index=0) # Set English as default language

if st.button("Generate Image") and text_prompt:

    try:

        # Translate the text prompt to English if the selected language is not
English

        if target_language != "en":

            translator = Translator(to_lang=target_language)

            text_prompt = translator.translate(text_prompt)

            # Spell check the text prompt only if the selected language is English

            if target_language == "en":

                tool = LanguageTool('en-US')

                matches = tool.check(text_prompt)

                # Display spelling suggestions, if any

                if matches:

                    st.warning("Spelling Errors in the Text Prompt:")

                    return None # Return None to prevent image generation when
there are spelling errors

            # Generate image using translated prompt

            response_gpt3 = openai.Image.create(

                prompt=f"{text_prompt} ",

                n=1

```

```

    )

    image_url = response_gpt3['data'][0]['url']

    return image_url

except Exception as e:

    st.error(f'Error generating image: {str(e)}')

# To generate image variation and save to the database

def create_variation(image_url):

    try:

        # Download the image

        image_content =

Image.open(io.BytesIO(requests.get(image_url).content)).convert("RGB")

        # Resize the image using OpenCV to the fixed resolution

        user_selected_resolution = 1024 # Change this value if needed

        image_cv2 = cv2.cvtColor(np.array(image_content),

cv2.COLOR_RGB2BGR)

        resized_image_cv2 = cv2.resize(image_cv2, (user_selected_resolution,

user_selected_resolution))

        resized_image_pil = Image.fromarray(cv2.cvtColor(resized_image_cv2,

cv2.COLOR_BGR2RGB))

        # Convert the image to PNG format

        png_buffer = io.BytesIO()

        resized_image_pil.save(png_buffer, format="PNG")

        png_image = png_buffer.getvalue()

```

```

# Check if the image is less than 4 MB

if len(png_image) < 4 * 1024 * 1024:

    response_gpt3 = openai.Image.create_variation(

        image=png_image,

        n=1

    )

    variation_url = response_gpt3['data'][0]['url']

    # Save to the database

    cursor.execute('INSERT INTO image_variations (original_image_url,
variation_url) VALUES (?, ?)',

                    (image_url, variation_url))

    conn.commit()

    st.image(variation_url, caption='Generated Image',
use_column_width=True)

else:

    st.error("Error: Converted PNG image size exceeds 4 MB.")

except Exception as e:

    st.error(f"Error creating variation: {str(e)}")

# Generate image and create variation

if mode == "Generate Image":

    generated_image_url = generate_image()

    if generated_image_url:

```

```

        create_variation(generated_image_url)

# Close the database connection

conn.close()

```

MATCHING KEY POINTS

```

import cv2

import numpy as np

import streamlit as st

import matplotlib.pyplot as plt

# Function to display OpenCV images in Streamlit using Matplotlib

def st_cv2_imshow(image, format=".png"):

    fig, ax = plt.subplots()

    ax.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

    ax.axis("off")

    st.pyplot(fig)

# Function to load and match images

def match_images(image1, image2):

    # Convert the uploaded files to OpenCV format

    image1_cv = cv2.imdecode(np.frombuffer(image1.read(), np.uint8),
cv2.IMREAD_GRAYSCALE)

    image2_cv = cv2.imdecode(np.frombuffer(image2.read(), np.uint8),
cv2.IMREAD_GRAYSCALE)

```

```

# Initialize keypoint detectors and descriptors

detector = cv2.ORB_create() # You can use other detectors such as SIFT or
SURF

matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

# Detect keypoints and compute descriptors

keypoints1, descriptors1 = detector.detectAndCompute(image1_cv, None)

keypoints2, descriptors2 = detector.detectAndCompute(image2_cv, None)

# Check if keypoints and descriptors are found

if keypoints1 is None or keypoints2 is None or descriptors1 is None or
descriptors2 is None:

    st.error("Error: Keypoints or descriptors not found.")

    return None

# Match descriptors between the two images

matches = matcher.match(descriptors1, descriptors2)

# Sort matches based on their distances

matches = sorted(matches, key=lambda x: x.distance)

# Draw only good matches, set a threshold for matching distances

good_matches = [match for match in matches if match.distance < 50]

# Check if good matches are found

if not good_matches:

    st.error("Error: No good matches found.")

    return None

```



```

# Draw the matches

result_image = cv2.drawMatches(

    image1_cv, keypoints1, image2_cv, keypoints2, good_matches, None,
    flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS

)

# Print the number of good matches

st.write(f'Number of good matches: {len(good_matches)}')

# Print the coordinates of matched keypoints

for match in good_matches:

    kp1 = keypoints1[match.queryIdx].pt

    kp2 = keypoints2[match.trainIdx].pt

    st.write(f'Matched keypoints: {kp1} -> {kp2}')

return result_image

# Streamlit app

def main():

    st.title("Image Matching with Keypoints")

    # File uploader for the two images

    image1 = st.file_uploader("Improvized Dall.E Image", type=["png", "jpg",
"jpeg"])

    image2 = st.file_uploader("Original Image", type=["png", "jpg", "jpeg"])

    # Check if both images are uploaded

    if image1 and image2:

```

```

# Display the uploaded images

st.image(image1, caption="Improvized Dall.E", use_column_width=True)

st.image(image2, caption="Real Image", use_column_width=True)

# Match images when the user clicks the button

if st.button("Match Images"):

    # Perform matching and get the result image

    result_image = match_images(image1, image2)

    # Display the result image

    if result_image is not None:

        st_cv2_imshow(result_image, format=".png")

# Run the app

if __name__ == "__main__":

    main()

```

REFERENCES

- 1) C. Chen et al., “UTC: A unified transformer with inter-task contrastive learning for visual dialog,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022.
- 2) J. Cheng, F. Wu, Y. Tian, L. Wang, and D. Tao, “RiFeGAN: Rich feature generation for text-to-image synthesis from prior knowledge,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 10908–10917.
- 3) Q. Cheng and X. Gu, “Bridging multimedia heterogeneity gap via graph representation learning for cross-modal retrieval,” *Neural Network.*, vol. 134, pp. 143–162, 2021.
- 4) Q. Cheng, Z. Tan, K. Wen, C. Chen, and X. Gu, “Semantic pre-alignment and ranking learning with unified framework for cross-modal retrieval,” *IEEE Trans. Circuits Syst. Video Technol.*, early access, Jun. 13, 2022.
- 5) Y.-C. Chen et al., “UNITER: Universal image-text representation learning,” in Proc. Eur. Conf. Computer Vision., 2020.
- 6) Y. Dong, Y. Zhang, L. Ma, Z. Wang, and J. Luo, “Unsupervised text-to-image synthesis,” *Pattern Recognit.*, vol. 110, 2021, Art. no. 107573.
- 7) F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler, “VSE++: Improving visual-semantic embeddings with hard negatives,” in Proc. Brit. Mach. Vis. Conf., 2020.

8) L. Gao, D. Chen, Z. Zhao, J. Shao, and H. T. Shen, “Lightweight dynamic conditional GAN with pyramid attention for text-to-image synthesis,” *Pattern Recognition* vol. 110, 2021.

9

9) K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *Proc. IEEE/CVF Conf.Comput. Vis. Pattern Recognit.*, 2020, pp. 9729–9738.

10) T. Hinz, S. Heinrich, and S. Wermter, “Semantic object accuracy for generative text-to-image synthesis,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 3, pp. 1552–1565, Mar. 2022.

11) S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 2020.

12) K. Joseph, A. Pal, S. Rajanala, and V. N. Balasubramanian, “C4Synth: Cross-caption cycle-consistent text-to-image synthesis,” in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2020,

13) R. Li, N. Wang, F. Feng, G. Zhang, and X. Wang, “Exploring global and local linguistic representation for text-to-image synthesis,” *IEEE TransMultimedia*, vol. 22, no. 12, pp. 3075–3087, Dec. 2020.

14) Z. Li, F. Ling, C. Zhang, and H. Ma, “Combining global and local similarity for cross-media retrieval,” *IEEE Access*, vol. 8, pp. 21847–21856, 2020.

15) J. Liang, W. Pei, and F. Lu, “CPGAN: Content-parsing generative adversarial networks for text-to-image synthesis,” in *Proc. Eur. Conf. Computer Vis.*, 2020, pp. 491–508.

- 16) D. Peng, W. Yang, C. Liu, and S. Lü, "SAM-GAN: Self-attention supporting multi-stage generative adversarial networks for text-to-image synthesis," *Neural Netw.*, vol. 138, pp. 57–67, 2021.
- 17) J. Qi, Y. Peng, and Y. Yuan, "Cross-media multi-level alignment with relation attention network," in *Proc. 27th Int. Joint Conf. Artif. Intell 2020*, pp. 892–898.
- 18) Sauer, K. Chitta, J. Müller, and A. Geiger, "Projected GANs converge faster," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 17480–17492.
- 19) C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern*, 2020,
- 20) Ramesh et al., "Zero-shot text-to-image generation," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8821–8831.
- 21) Z. Tan, C. Chen, K. Wen, Y. Qin, and X. Gu, "A unified two-stage group semantics propagation and contrastive learning network for co-saliency detection," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2022, pp. 1–6.
- 22) H. Tan, X. Liu, M. Liu, B. Yin, and X. Li, "KT-GAN: Knowledge-transfer generative adversarial network for text-to-image synthesis," *IEEE Trans Image Process.*, vol. 30, pp. 1275–1290, 2021.
- 23) M. Tao et al., "DF-GAN: A simple and effective baseline for text-to-image synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16515–16525.
- 24) P. Velic'cković et al., "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2020.
- 25) M. Wang et al., "End-to-end text-to-image synthesis with spatial constraints," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 4, pp. 1–19, 2020.

- 26) H. Wang, C. Deng, F. Ma, and Y. Yang, “Context modulated dynamic networks for actor and action video segmentation with language queries,” in Proc. AAAI Conf. Artif. Intell., 2020.
- 27) K. Wen, X. Gu, and Q. Cheng, “Learning dual semantic relations with graph attention for image-text matching,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 7, pp. 2866–2879, Jul. 2021.
- 28) M. Yuan and Y. Peng, “Bridge-GAN: Interpretable representation learning for text-to-image synthesis,” IEEE Trans. Circuits Syst. Video Technology vol. 30, no. 11, pp. 4258–4268, Nov. 2020.
- 29) T. Xu et al., “AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks,” in Proc. IEEE Conf. Computer Vision Pattern Recognition., 2018, pp. 1316–1324.
- 30) H. Zhang et al., “StackGAN++: Realistic image synthesis with stacked generative adversarial networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 8, pp. 1947–1962, Aug. 2020.