Text to Image Conversion

Ankita Gandhi, Govardhan Vamsi Tellakula,
Neeraj Babu Thatiparthi, Ramachandra Pentakota and
Zulfiqar Ahmed Syed

March 18, 2024

TEXT TO IMAGE CONVERSION

Prof.Ankita Gandhi
Assistant professor
Dept. Computer Science and Engineering, Parul University,
Vadodara, India.
ankita.gandhi@paruluniversity.ac.in

Tellakula Govardhan Vamsi
Research Scholar,
Dept. Computer Science and Engineering, Parul University,
Vadodara, India.
govardhanvamsi0@gmail.com

Thatiparthi Neeraj Babu
Research Scholar,
Dept. Computer Science and Engineering, Parul University,
Vadodara, India.
nanineerajbabu@gmail.com

Pentakota Ramachandra
Research Scholar,
Dept. Computer Science and Engineering, Parul University,
Vadodara, India.
ramachandrapentakota@gmail.com

Syed Zulfiqar Ahmed
Research Scholar,
Dept. Computer Science and Engineering, Parul University,
Vadodara, India.
syedzulfiqarahmed27@gmail.com

***Abstract*—Text-to-image generation is the process of generating realistic images from textual descriptions.This involves training deep learning models to understand the relationship between textual input and visual output, and using this understanding to generate images that accurately represent the text.Text-to-image generation has applications in a variety of fields, including computer vision, natural language processing, and art.However, this field still faces many challenges, including generating high-quality and diverse images, capturing complex visual concepts, and scaling to handle large datasets.**

Keywords: Python, EasyOCR, Streamlit lib, NLP, DeepLearning, openai

I. Introduction

Text-to-image generation is an emerging domain that revolves around instructing computers to convert text into visual representations. This involves training deep learning models to comprehend the connection between textual descriptions and their visual counterparts, enabling the generation of images that faithfully depict the provided text. Text-to-image generation is versatile and finds utility in various fields, including computer vision, natural language processing, and creative endeavors. For instance, it can be leveraged to automatically produce images based on textual inputs, facilitating tasks such as image retrieval and content creation.

Text-to-image generation involves teaching computers to interpret written descriptions and create corresponding visuals. This capability has wide-ranging applications, from simplifying online image searches to generating fresh content. For instance, typing a description of a sunny beach could prompt the computer to generate an accurate picture, making it valuable for various fields such as computer science, communication with computers, and creative endeavors like gaming and filmmaking.

The main challenge lies in helping computers understand human language well enough to accurately translate it into images. However, recent advancements have resulted in remarkably realistic images. Looking forward, this technology could revolutionize visual content creation and consumption, potentially transforming industries such as art, entertainment, and online commerce. Essentially, it's about teaching computers to visually interpret our verbal descriptions, paving the way for exciting possibilities in human-computer interaction.

A. Problem Statement

The task at hand, using the Python library Streamlit and deep learning, revolves around the generation of images that are coherent and meaningful when given a textual description. Essentially, it's about training a deep learning model to comprehend how words and images relate, extract the pertinent visual elements from the text, and then create images that accurately depict the text's content.

B. Scope

The possibilities in text-to-image generation using Python's Streamlit and deep learning techniques are extensive and offer a broad spectrum of potential applications. One key area for its utilization lies in the realms of virtual and augmented reality, where it can contribute to creating more lifelike and immersive digital environments.

C. Aim and Objectives

The primary goal of text-to-image synthesis using Python is to empower machines to craft authentic images based on written descriptions, essentially closing the gap between human language and visual artistry. The objective is to create models that can comprehend both the meaning and spatial relationships conveyed in text and produce images that not only look credible but also align with the text's message.

II. Literature Survey

Imagine a world where computers can interpret words and transform them into visuals – that's the essence of text-to-image conversion. This review delves into the significance of this technology and its practical applications, particularly when utilized alongside EasyOCR and Python libraries.

A. Utilization of Text-to-Image Conversion

Module 1: Creative Content Generation

Visualizing ideas without artistic skills sounds like a dream, right? Text-to-image conversion makes it a reality. With the assistance of EasyOCR and Python libraries, creators can translate descriptions into captivating visuals effortlessly.

Module 2: E-commerce and Product Visualization

Ever tried to imagine a product based solely on its description? Text-to-image conversion changes that. By converting product details into images, online retailers can offer customers a clearer understanding of their offerings, thanks to EasyOCR and Python's capabilities.

Module 3: Data Augmentation for Computer Vision

Teaching machines to recognize objects requires ample examples. Text-to-image conversion aids in this by expanding image datasets. With EasyOCR and Python, researchers can enhance datasets, empowering computers to perceive the world more accurately.

Module 4: Content Creation for Social Media

Crafting engaging social media content can be challenging. However, text-to-image conversion simplifies the process. Marketers can now effortlessly turn text into visually striking posts using EasyOCR and Python, capturing attention and conveying messages effectively.

Module 5: Accessibility and Assistive Technologies

For individuals with visual impairments, navigating the digital world can be daunting. Text-to-image conversion changes the narrative by providing descriptive images. Leveraging EasyOCR and Python, assistive technologies can offer a more inclusive online experience.

B. Integration of EasyOCR and Python

EasyOCR serves as a powerful tool for extracting text from images. Paired with Python, it becomes even more potent, enabling developers to seamlessly generate images from text using Python's extensive image processing capabilities.

III. Methodology

A. Software and Libraries

As you set out to create a Text-to-Image conversion application using EasyOCR and Streamlit, there's a lineup of crucial software components and libraries you'll need for the project's construction and operation. Below, I'll furnish you with a list of these indispensable elements and a foundational code structure for your Streamlit-based application.

1. Python: Ensure you have Python installed on your system.

2. Streamlit: Install Streamlit, a Python library for creating web applications with minimal code.

pip install streamlit

3. EasyOCR: Install the EasyOCR library for OCR functionality.

pip install easyocr

4. Pillow: You'll need Pillow for image manipulation. pip install Pillow

B. System Architecture

Let's delve into the system architecture of a Text to Image conversion application using EasyOCR. The typical setup involves a front-end web interface, constructed with a user-friendly framework such as Streamlit. This interface is where users can get creative by inputting their text and finetuning image attributes like font size, text color, background color, and image dimensions. It's the place where the magic unfolds.

The front-end seamlessly communicates with the backend, which serves as the operational core of the system. The backend is like the powerhouse behind the scenes. Here, we harness the capabilities of the robust EasyOCR library for Optical Character Recognition, ensuring the generated image's text content is impeccably accurate. When it comes to creating and customizing the images, we turn to the dependable Pillow library.

The entire architecture is meticulously designed to deliver a smooth user experience. User inputs effortlessly flow into the image generation process, with an additional step dedicated to OCR verification. This step plays a crucial role in guaranteeing the precision of the text extraction process.

C. Data Flow

Let's take a step-by-step journey through how data flows in a text-to-image conversion process using EasyOCR. It all begins with the user's input. The user provides the text they want to transform and personalizes the image's appearance, tweaking elements like font size, text color, and background color. This customization is made super easy through a user-friendly interface designed to give users full creative control.

Once the user decides to start the conversion process, their input text and all the custom details are passed over to the core logic of the application. This is where the real wizardry happens. We employ the Pillow library to craft a fresh image, thoughtfully incorporating all the attributes the user has chosen. Their text is elegantly placed on this canvas.

The end result? A remarkable image that's both a work of art and a visual representation of the user's creativity. We save this masterpiece and proudly present it to the user for their admiration and sharing pleasure. It's a journey that transforms user ideas into visually striking reality.

D. Text-to-Image Conversion Algorithm

1. Initialize Parameters: Initialize variables with customization options, e.g., font, font size, text color, background color, image width, and image height.

Fig. 1. Dataset

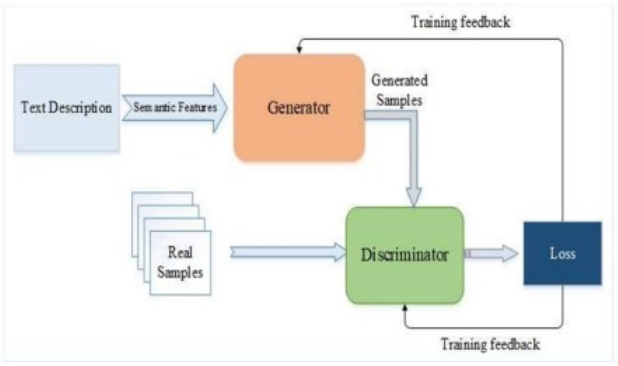


Fig. 2. Design Architect

2. Create a Blank Image: Create a new blank image using an image processing library like Pillow (PIL). Set the image dimensions to the specified width and height. Set the image background color to the specified background color.

3. Load Font: Load the selected font to be used for rendering the text. The font should be compatible with the image processing library being used.

4. Calculate Text Position: Determine the position to place the text on the image to ensure it is centered or aligned as desired. This may involve calculating the width and height of the text and the image and positioning it accordingly.

5. Draw Text on Image: Using the loaded font and customization options, draw the text on the blank image. Use the calculated text position from step 4.

6. Save or Display Image: Save the generated image to a file for later use. Optionally, display the image to the user in the application's user interface.

7. Optional Post-Processing: Depending on the application, you can apply additional post processing steps to the image, such as adding effects or overlays.

8. End of Algorithm: The algorithm completes, and the resulting image contains the text as specified by the user.

E. OCR Verification

In the realm of Text-to-Image conversion using EasyOCR, there's a crucial step known as OCR Verification. This step is all about guaranteeing the precision and trustworthiness of the text extracted from the images we've created.

Here's how it plays out: First, we craft these stunning images, each adorned with the text customized by the user. Once these visual wonders are ready, we call upon OCR, which stands for Optical Character Recognition. EasyOCR is our trusty companion for this task. It meticulously scans the images, carefully retrieving the text content.

The essence of OCR Verification lies in the evaluation process. We rigorously cross-check the extracted text against the original input to ensure a perfect match. But that's not all. EasyOCR sweetens the deal by providing us with confidence scores, giving us insights into how confident it is about the recognition. This way, we're equipped to gauge the reliability of the OCR results. It's a safety net that helps us identify and gracefully handle any instances of uncertainty or inaccuracy.

In a nutshell, OCR Verification stands as a pillar of the text-to-image conversion process. It's the guardian of accuracy, ensuring that the images we produce faithfully reflect the intended text. It's our way of maintaining fidelity to the user's creative vision.

IV. Implementation

A. User Interface

In our Text-to-Image conversion application, driven by the dynamic duo of EasyOCR and Streamlit, we've put a premium on user experience. When users step into our world, they'll find an interface that's not only user-friendly but also visually inviting.

Here's what they get: A clean canvas where they can easily enter the text they want to see turned into a stunning image. Right next to the text input area, there's a friendly sidebar, kind of like their creative toolkit. It's where they can play around with things like font size, text color, background color, and image dimensions. This puts them in the driver's seat, allowing them to customize the generated image to their heart's content.

To set the wheels in motion, there's a prominent "Generate Image" button, ready for their click. It's the launchpad for turning their text into a visual masterpiece. And when that image is ready, we don't just drop it there. We present it elegantly, but there's more. We also reveal the Optical Character Recognition (OCR) results, like showing them how the magic trick works. This holistic experience empowers users to effortlessly create personalized textual images.

In essence, our application is all about making creativity a breeze while putting users at the center of it all. It's a realm where ideas transform into beautiful visuals with ease and sophistication.

B. Code Implementation

In our code implementation, we've utilized Streamlit to create a web interface that's all about user-friendliness. Here's the breakdown of how it all works:

- Users arrive at the interface, where they can smoothly input their text and let their creativity flow when it comes to image details. The place for this creative exploration is a nifty sidebar where they can fine-tune elements like font size, text color, and background color. It's like a canvas for their imagination.

- When they're ready to see the magic unfold, all they need to do is hit the "Generate Image" button. It's their cue to witness their customized image come to life. We rely on Pillow to weave this image, ensuring it matches their precise specifications.

- But we don't just stop there. We present the image to the user with style. However, that's not the grand finale. We invite EasyOCR into the act. It's like the magician's big reveal. EasyOCR takes a meticulous look at the created image, extracting the text with expert precision. And it doesn't stop at mere text recognition; it also hands us confidence scores, showing how certain it is about the results.

- And we make sure that the user gets to see all of this in a way that's simple to grasp and enjoy.

In essence, our code isn't just about turning text into images. It's about giving users an experience that's smooth, delightful, and dependable. It's like pulling off the perfect magic trick that leaves everyone amazed.

C. Error Handling

In a Text-to-Image conversion application that harnesses the power of EasyOCR via Streamlit, the art of error handling is a pivotal piece in the puzzle. It plays a starring role in ensuring the application's strength and preserving a user experience that's as smooth as silk. Let's dive into the critical areas where error handling truly shines:

1. **User Inputs:** Errors can sneak in when users input text or fiddle with image settings. Managing these gracefully is key to keeping the user experience frustration-free.

2. **Image Generation:** Whether it's technical glitches or clashes in customization, handling errors during image creation is essential for maintaining the creative flow.

3. **OCR Journey:** Hiccups might arise during the Optical Character Recognition process, and addressing these ensures that the extracted text is accurately interpreted.

4. **Confidence Scores:** When dealing with OCR confidence scores, it's paramount to tackle any doubts or inaccuracies to guarantee the precision of the recognized text.

5. **User Feedback:** The door to feedback and support should always be open. If users encounter errors or challenges, effective error handling includes mechanisms for them to seek assistance.

6. **Application Stability:** Technical mishaps or application hiccups can disrupt the user experience. Astute error handling aims to minimize these disruptions.

To sum it up, error handling is the unsung hero that keeps the Text-to-Image conversion application running smoothly, even when unexpected twists and turns come into play. It's all about ensuring that users have a reliable and user-friendly platform where they can bring their creative ideas to life with unwavering confidence.

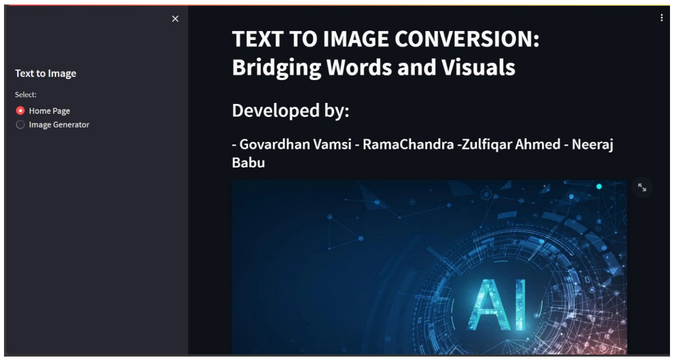
- OCR Failures
- Missing Fonts
- Image Creation Errors
- Input Validation
- File Handling Errors
- Exception Handling

D. Key features and Results

Key Features

- Text Input: Users can enter detailed textual descriptions, allowing for creative expression and specificity in generating images.
- Image Quantity Control: Users have the flexibility to specify the number of images they want the system to generate based on the given text.
- Image Size Customization: Users can choose their preferred dimensions for the generated images, enabling compatibility with various platforms and use cases.
- Advanced AI Algorithms: The system employs state-of-the-art AI models to convert textual information into coherent visual depictions, ensuring a high level of accuracy and realism.
- Creative Diversity: The generated images cover a wide spectrum of subjects, settings, and styles, providing users with a diverse range of visual outputs.
- Real-Time Preview: Users can preview generated images before finalizing their selections, enabling them to make adjustments to the text or settings as needed.
- Download and Sharing: Once satisfied with the generated images, users can download them in their chosen size and format, making them suitable for various projects. Images can also be easily shared on social media platforms.

The Text-to-Image Generator is an innovative web application that transforms textual descriptions into captivating visual representations. Users can simply input a description in natural language, specify the desired number of images, and select the image size preferences. The application leverages cutting-edge image generation techniques to create high-quality images that closely match the provided text.



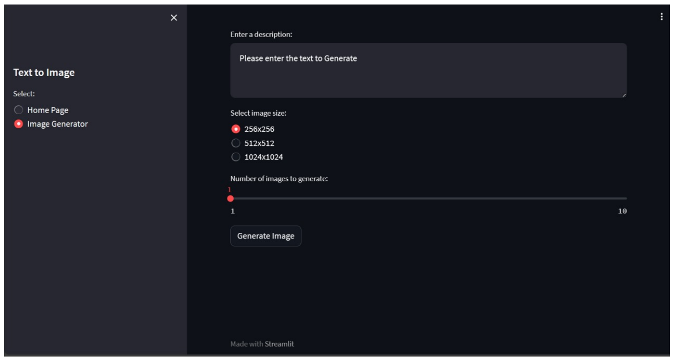
Select: [✓ Home Page] Image Generator

TEXT TO IMAGE CONVERSION: Bridging Words and Visuals

Developed by:

- Govardhan Vamsi - RamaChandra -Zulfiqar Ahmed - Neeraj Babu

Fig. 3. Landing Page-1



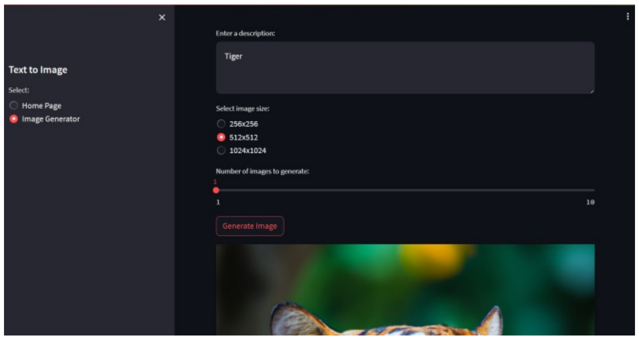
Select: Home Page [✓ Image Generator]

Enter a description: Please enter the text to Generate

Select image size: [✓ 256x256] 512x512 1024x1024

Number of images to generate: 1 (1–10)

Fig. 4. Landing Page-1



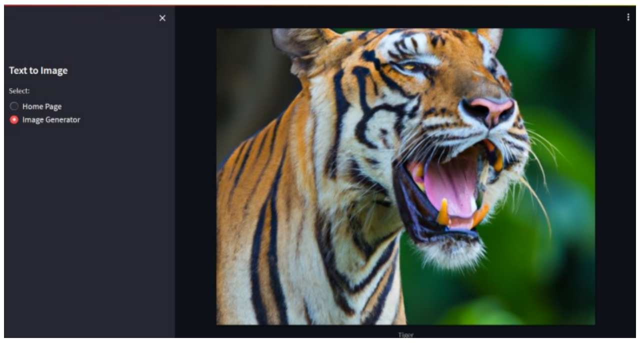
Select: Home Page [✓ Image Generator]

Enter a description: Tiger

Select image size: 256x256 [✓ 512x512] 1024x1024

Number of images to generate: 1 (1–10)

Fig. 5. Landing Page-1



Select: Home Page [✓ Image Generator]

Fig. 6. Landing Page-1

Conclusion and Future Work

Text-to-image conversion is like a magic wand that transforms words into pictures, making communication more vibrant and engaging. With tools like EasyOCR and Python libraries, this technology becomes accessible to everyone, fueling creativity and breaking down barriers in various areas, from sparking imagination to making digital content more inclusive. As it evolves, text-to-image conversion promises to revolutionize our digital experiences, paving the way for innovation and ensuring that everyone can participate in the digital world.

The field of text-to-image generation is rapidly evolving, and there are many exciting directions for future research. Some potential areas for future work include:

- Elevating Image Quality
- Tackling Complex Text
- Embracing Multimodal Input
- Real-Time Image Generation
- Enhancing Interpretability and Control

References

[1] Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X. (2017). StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision (pp. 5907-5915).

[2] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H. (2016). Generative adversarial text to image synthesis. In Proceedings of the 33rd International Conference on Machine Learning (pp. 1060-1069).

[3] Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X. (2018). Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1316-1324).

[4] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I., Reed, S., Akata, Z., Lee, H., Schiele, B. (2021). Zero-shot text-to-image generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp.10665-10675).

[5] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik1 Amit H. Bermano Gal Chechik, Daniel CohenOr, Hao Tang, Xinxiao Wu, An-An Liu, Yi Jin, and Nicu Sebe. "An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 10670-10679.

[6] Akanksha Singh, Sonam Anekar, Ritika Shenoy, Han Zhang, Jun-Yan Zhu, Ting-Chun Wang, and Lior Wolf. "Text to Image using Deep Learning." Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW), 2017, pp. 1146-1154.

[7] Wentong Liao, Kai Hu, Michael Ying Yang, Bodo Rosenhahn, Jiebo Luo, Chen Chen, Qiaoyong Zhong, and Hongyu Yang. "Text to Image Generation with Semantic-Spatial Aware GAN." Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2019, pp. 10734-10743.

[8] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H. (2016). Generative adversarial text to image synthesis. In ICML'16 Proceedings of the 33rd International Conference on International Conference on Machine Learning (pp. 1060-1069). University of Michigan, Ann Arbor, MI, USA.

[9] Shukla, S., Mitra, P., Chaudhuri, B. B., Kulkarni, C. R., Barbadekar, A. B. (2019). Text Detection and Recognition: A Review. In 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 1206-1211). IEEE.