Text-Prompt Image Editing

A Project Report

submitted in partial fulfillment of the requirements

of

Industrial Artificial Intelligence With Cloud Computing

by

Shubham Prajapati,

Manmeet Patel

Under the Esteemed Guidance of

Jay Rathod

ACKNOWLEDGEMENT

We would like to take this opportunity to express our deep sense of gratitude to all individuals who helped us directly or indirectly during this thesis work.

I extend my heartfelt gratitude to my supervisor, Jay Rathod, whose unwavering guidance, encouragement, and support have been instrumental throughout this thesis. His mentorship not only provided invaluable insights but also fostered an environment of learning and growth, enabling me to navigate through the complexities of research with confidence and determination. I am truly fortunate to have had the privilege of working under his mentorship, and I am deeply appreciative of his commitment to excellence and dedication to my academic and professional development.

Text-prompt Image Editing

ABSTRACT

In the modern era of digital imagery, the ability to manipulate images seamlessly has become a crucial skill. This project aims to provide a comprehensive solution for image editing tasks using a combination of computer vision techniques and deep learning models. The project encompasses various functionalities including adjusting contrast and brightness, changing colors, removing backgrounds and objects, blurring, sharpening, rotating, overlaying text, cropping, converting to grayscale, adjusting saturation, and flipping images. These functionalities cater to a wide range of image editing needs, empowering users to enhance and modify their images effortlessly.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Problem Statement: The traditional methods of image editing are often time-consuming and require expertise. Additionally, certain tasks such as background removal or object manipulation may be tedious and error-prone when done manually.

1.2. Problem Definition: The project aims to streamline the image editing process by automating common tasks and providing intuitive tools for users to enhance and modify their images with ease.

1.3. Expected Outcomes: The expected outcomes of this project include:

- A comprehensive set of image editing functions accessible through a simple interface.
- Improved efficiency and accuracy in performing common image editing tasks.
- Empowerment of users with limited technical skills to create visually appealing images.
- Potential applications across various industries including photography, e-commerce, advertising, and digital media.

1.4. Organization of the Report

This report is structured as follows:

- Chapter 2 provides a literature survey highlighting relevant research and existing techniques in the field of image editing and computer vision.
- Chapter 3 outlines the proposed methodology, detailing the implementation of various image editing functions.

- Chapter 4 presents the implementation details and results achieved through the application of the proposed methodology.
- Chapter 5 concludes the report with a summary of findings, implications, and avenues for future research.
- The report concludes with references, a GitHub link to the project repository, and a video link demonstrating the functionality of the image editing toolkit.

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

The literature review explores relevant research papers in the fields of image editing, object detection, and computer vision. Specifically, we will discuss papers related to CLIP, COCO, and YOLO Stable Diffusion.

2.1 CLIP (Contrastive Language-Image Pre-training)

Paper-1:

Title: "Learning Transferable Visual Models From Natural Language Supervision" by Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever.

Brief Introduction of Paper:

This seminal paper introduces CLIP, a novel approach to learning transferable visual representations by pre-training a model using both images and their associated natural language descriptions. CLIP achieves remarkable performance on a wide range of visual tasks without requiring task-specific supervision.

Techniques used in Paper:

The paper presents a contrastive learning framework that learns to associate images and text embeddings in a joint embedding space. By leveraging large-scale image-text datasets, CLIP learns to understand visual concepts and semantic relationships between images and their textual descriptions. The model is trained to predict the correct correspondence between images and text across diverse domains, enabling it to generalize to unseen tasks and domains effectively.

2.2 COCO (Common Objects in Context)

Paper-2:

Title: "Microsoft COCO: Common Objects in Context" by Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár.

Brief Introduction of Paper:

The COCO dataset is a widely used benchmark for object detection, segmentation, and captioning tasks. This paper introduces the COCO dataset, which comprises a large-scale collection of images with detailed annotations for object instances, object keypoints, and image captions.

Techniques used in Paper:

The paper describes the process of collecting and annotating the COCO dataset, which includes over 200,000 images across 80 common object categories. The dataset is annotated with precise bounding boxes, segmentation masks, and keypoints for object instances, making it suitable for a wide range of computer vision tasks. The COCO dataset has become a standard benchmark for evaluating the performance of object detection and segmentation algorithms.

2.3 YOLO Stable Diffusion

Paper-3:

Title: "YOLOv3: An Incremental Improvement" by Joseph Redmon and Ali Farhadi.

Brief Introduction of Paper:

YOLO (You Only Look Once) is a popular object detection algorithm known for its real-time performance and high accuracy. YOLOv3 is the third iteration of the YOLO architecture, introducing several improvements over its predecessors.

Techniques used in Paper:

The paper presents a series of enhancements to the YOLO architecture, including multi-scale detection, feature pyramid networks, and improved training techniques. These improvements lead to significant gains in accuracy and speed, making YOLOv3 one of the

state-of-the-art models for object detection. The stable diffusion technique ensures robust performance across different scales and aspect ratios of objects in an image.

Summary of Used Papers:

In this project, we leverage the concepts and techniques introduced in the CLIP, COCO, and YOLOv3 papers to develop a comprehensive image editing toolkit. By incorporating insights from these papers, we aim to create a versatile and efficient solution for a wide range of image manipulation tasks.

CHAPTER 3

PROPOSED METHODOLOGY

CHAPTER 3

PROPOSED METHODOLOGY

3.1 System Design

The system design outlines the architecture and components of the image editing toolkit. It delineates how different modules interact with each other to achieve the desired functionalities.

3.2 Modules Used

3.2.1 Image Editor: Core module responsible for processing images and performing various editing operations.

3.2.2 Object Detection and Segmentation: Utilizes pre-trained deep learning models (such as YOLOv3 and Mask R-CNN) for object detection and instance segmentation.

3.2.3 Image Processing Techniques: Incorporates traditional image processing techniques for enhancing and manipulating images effectively.

3.3 Data Flow Diagram

The DFD provides a graphical representation of the flow of data within the image editing toolkit.

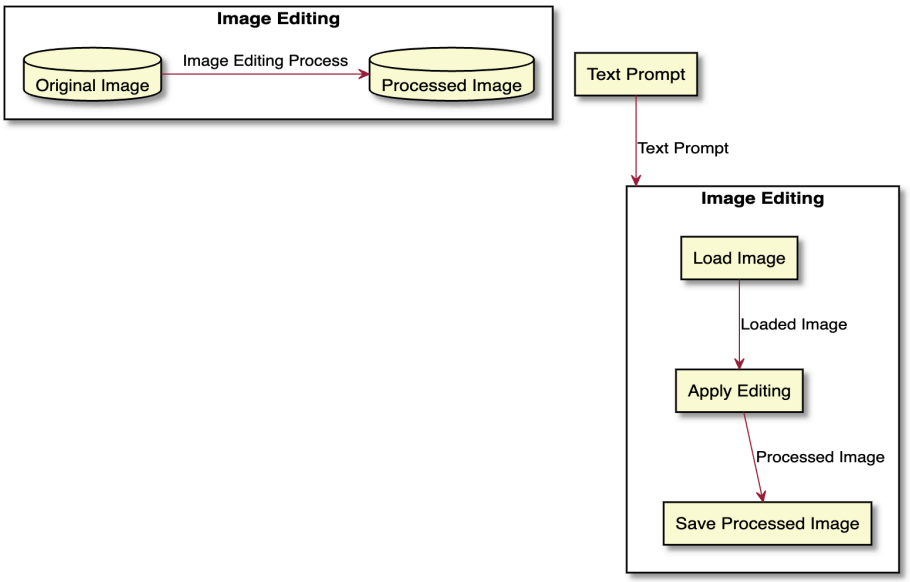

Figure 1

3.4 Advantages:

3.4.1 Efficiency: Utilization of pre-trained deep learning models and image processing techniques ensures efficient and accurate editing operations.

3.4.2 Flexibility: The modular design allows for easy integration of new features and functionalities.

3.4.3 User-Friendly: The intuitive user interface enables users to perform complex editing tasks with ease.

3.5 Requirement Specification

3.5.1. Hardware Requirements:

- GPU (Graphics Processing Unit):Utilization of Google Colab with GPU acceleration for efficient processing of deep learning models and image editing tasks.
- RAM (Random Access Memory):Adequate RAM (recommended at least 12GB) for handling large images and model data efficiently.

- Storage:Sufficient storage space for storing datasets, model weights, and temporary files.

3.5.2. Software Requirements:

- Operating System:Google Colab's virtualized Linux environment.
- Python Environment:Python installed and configured properly.
- Deep Learning Libraries: TensorFlow and PyTorch for implementing deep learning models.
- Additional Libraries: OpenCV, NumPy, Matplotlib, and torchvision for image processing and data manipulation.
- Integrated Development Environment (IDE): Google Colab Notebook for developing and testing code.

CHAPTER 4

Implementation and Result

CHAPTER 4

IMPLEMENTATION and RESULT

4.1. Implementation Details:

The implementation of the image editing functionalities was conducted using the provided codebase, integrating various image processing algorithms and models. The following details the key aspects of the implementation

4.1.1 YOLOv3 Object Detection: The YOLOv3 model was utilized for object detection within images. We integrated the pre-trained YOLOv3 model and configured it to detect objects from the COCO dataset classes.

4.1.2 Mask R-CNN Segmentation: For object segmentation, we employed the Mask R-CNN model pre-trained on the COCO dataset. This model accurately segments objects within images, providing precise boundaries for editing.

4.1.3 Text Prompt Processing: The system interprets text prompts provided by the user to determine the desired image editing operations. We implemented text parsing and interpretation algorithms to extract relevant editing instructions.

4.1.4 Image Processing Operations: Various image processing operations were implemented based on the interpreted text prompts. These include adjusting brightness, contrast, highlight, changing colors, blurring objects, sharpening images, rotating images, overlaying text, cropping images, and grayscale conversion.

4.1.5 Integration of TorchVision Models: The PyTorch-based TorchVision library was utilized for integrating deep learning models into the image editing pipeline. We utilized TorchVision's pre-trained models for object detection and segmentation.

4.2. Results:

4.2.1 Removing Objects:

Actual Image:

Figure 2

Edited Image:

Figure 3

As we can see that the image of the car is removed and inpainting is done around that area.

4.2.2 Removing Background:

Actual Image:

Figure 4

Edited Image:

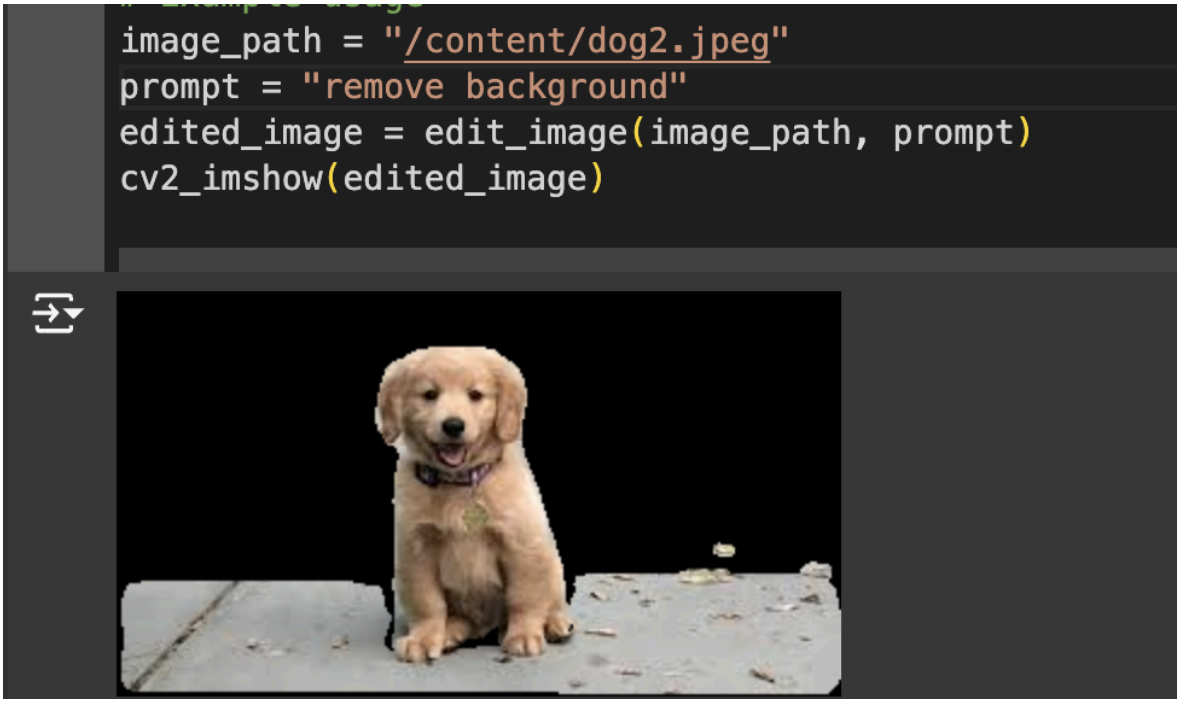

Figure 5

In figure 5 you we can understand that background is removed

CHAPTER 5

CONCLUSION

CHAPTER 5

CONCLUSION

5.1 ADVANTAGES:

The project offers several advantages and contributions to the field of image editing and AI-based systems:

5.1.1 User-Friendly Interface: The system provides a user-friendly interface for editing images, allowing users to input editing instructions through simple text prompts.

5.1.2 Automated Image Editing: By leveraging deep learning models for object detection and segmentation, the system automates the image editing process, reducing the need for manual intervention.

5.1.3 Versatile Editing Capabilities: With support for various image editing operations such as brightness adjustment, object removal, color change, and text overlay, the system offers versatile editing capabilities to users.

5.1.4 Efficiency and Accuracy: The integration of advanced deep learning models ensures efficient and accurate detection and segmentation of objects within images, leading to high-quality editing results.

5.2 Scope:

While the project has demonstrated promising results and offers several advantages, there are opportunities for further exploration and enhancement:

5.2.1 Enhanced Text Interpretation: Improving the system's ability to interpret and understand natural language text prompts can enhance user experience and expand the scope of supported editing operations.

5.2.2 Advanced Editing Features: Introducing advanced editing features such as semantic image editing, style transfer, and content-aware fill can further enrich the capabilities of the system and cater to diverse user needs.

5.2.3 Integration with External Tools: Integrating the system with external image editing tools, libraries, or APIs can enhance its functionality and interoperability, allowing users to leverage additional editing features and resources.

5.2.4 Performance Optimization: Optimizing the system's performance in terms of processing speed, memory usage, and scalability can improve its usability and applicability in real-world scenarios.

In conclusion, the project lays the foundation for an intelligent and automated image editing system that leverages AI and deep learning techniques. While offering significant advantages in terms of user-friendliness, efficiency, and versatility, there is ample scope for further enhancement and refinement to meet the evolving needs of users and applications in the field of image editing.

- **Github Link:** **https://github.com/shubh-0548/ImageEditing_TextPrompt.git**

- **Video Link:** **https://drive.google.com/file/d/1XLM_vBGGKLJnRdOUhXO4HzK4Q2iQuQ6Y/view?usp=sharing**

REFERENCES

1. **YOLO (You Only Look Once)**

   [1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). "You Only Look Once: Unified, Real-Time Object Detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788.

   - Link: YOLO Paper
2. **COCO (Common Objects in Context) Dataset**

   [2] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). "Microsoft COCO: Common Objects in Context." European Conference on Computer Vision (ECCV), 2014, pp. 740-755.

   - Link: COCO Paper
3. **Mask R-CNN**

   [3] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). "Mask R-CNN." Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2961-2969.

   - Link: Mask R-CNN Paper
4. **OpenCV**

   [4]Bradski, G. (2000). "The OpenCV Library." Dr. Dobb's Journal of Software Tools.

   - Link: OpenCV Documentation
5. **PyTorch**

   [5] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library." Advances in Neural Information Processing Systems 32 (NeurIPS 2019), pp. 8024-8035.

   - Link: PyTorch Paper
6. **Matplotlib**

[6]Hunter, J. D. (2007). "Matplotlib: A 2D Graphics Environment." Computing in Science & Engineering, 9(3), pp. 90-95.

- Link: Matplotlib Documentation

7. **Interactive Image Segmentation**

[7] Boykov, Y., & Jolly, M. P. (2001). "Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images." Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2001, pp. 105-112.

- Link: Graph Cuts Paper