Image Inpainting with Stable Diffusion and CLIPSeg

Overview

This project utilizes the Stable Diffusion model and CLIPSeg to perform image inpainting and editing. The script processes an uploaded image, identifies specific regions based on user-defined prompts, and modifies those regions using the inpainting capabilities of Stable Diffusion.

Process Flow

Step 1: Image Upload and Preparation

1. **Upload Image: The user uploads an image using the Google Colab file upload feature.**

NOTE: When you run the program, it will ask you to upload an image. Otherwise, the program will be waiting for you to upload an image.

2. **Resize Image: The uploaded image is resized to a target width and height of 512x512 pixels using the LANCZOS filter.**
3. **Transform Image: The resized image is transformed into a tensor and normalized to be used by the CLIPSeg model.**

Step 2: Prompt-Based Region Identification

1. **Define Prompts: User-defined prompts specify the regions of interest in the image (e.g., 'a hat', 'a dark skirt', 'shoes', 'a white shirt').**
2. **Generate Predictions: The CLIPSeg model generates segmentation masks for each prompt, identifying the specified regions in the image.**
3. **Display Masks: The identified regions are displayed in a grid format, showing the original image alongside the generated masks.**

Step 3: Image Inpainting

1. **Select Mask: The user selects one of the generated masks for inpainting.**
2. **Process Mask: The selected mask is processed to be used with the Stable Diffusion model.**
3. **Generate Inpainting Prompts: User-defined inpainting prompts specify the desired changes for the identified region (e.g., 'a skirt full of text', 'blue flowers', 'white flowers', 'a zebra skirt').**

4. **Perform Inpainting: The Stable Diffusion model generates new images based on the inpainting prompts and the selected mask.**
5. **Display Inpainted Images: The inpainted images are displayed in a grid format alongside the original image.**

How to Run the Program / EXAMPLE USAGE

1. Inside the code, define prompts such as 'a hat', 'a dark skirt', 'shoes', 'a white shirt' to identify these regions in the image **which you want to change.**

```
################### DO PROMPT ENGINEERING WHAT YOU WANT TO CHANGE STARTS ###################
prompts = ['a hat', 'a dark skirt', 'shoes', 'a white shirt']
################### DO PROMPT ENGINEERING WHAT YOU WANT TO CHANGE ENDS ######################
```

2. Inside the code, define prompts **for which you want your detected object to change/replace with**

```
inpainting_prompts = ["a skirt full of text",  "blue flowers", "white flowers", "a zebra skirt"]
```

3. Once you're done changing, run the code and wait until it gets / installs all dependencies and is done with importing libraries.
4. When it is running, it will ask you to upload an image which you want to perform operations on, **otherwise it will wait forever.**

By following these steps, users can efficiently edit specific regions of an image using natural language prompts and advanced inpainting techniques.