Diffusion Model for Image Inpainting

CSE499A – Section 15 **Group 4 Update- 2**

Name	**Student ID**
Md. Minhajul Islam	2211022042
Nur Ibne Kawsar Zitu	2212021042
Kazi Tazrian Mon	2132798642

Weekly Update**: Md.Minhajul Islam**

To refine the diffusion-based inpainting workflow developed in Week 1 by improving mask precision, visualization, and quantitative evaluation. The focus was to create a controlled, measurable, and visually verifiable restoration process capable of replacing or regenerating masked regions based on prompts.

Work Done:

1. **Mask Refinement:**
   Implemented smoother binary masks using morphological operations and adjustable ratio-based masking. This improved the accuracy of edited regions and eliminated hard edges.
2. **Visualization Pipeline:**
   Built a 3-panel display (Original | Mask | Inpainted) with automatic saving to Google Drive for consistent result logging and presentation.
3. **Prompt & Parameter Experiments:**
   Tested multiple text prompts such as *"restore background naturally"* and *"replace with flowers"* while varying inference steps (30–50) and guidance scales (5.0–10.0). Higher steps and moderate guidance produced visually cleaner and more coherent results.
4. **Quantitative Evaluation:**
   Added **SSIM** and **PSNR** metrics for both full and masked regions to measure image quality. Stored results in CSV for later comparison across configurations.
5. **Semantic Replacement Test:**
   Attempted to replace a dog on a bench with a vase of yellow flowers using the prompt-based inpainting pipeline. The system executed successfully but preserved the dog, showing correct technical operation but limited semantic control. This highlights the need for stronger guidance or cross-image conditioning.

I will extend the system for cross-image inpainting, inserting content from a reference image into masked regions for controllable editing.

Update: **Nur Ibne Kawsar Zitu**

I focused on building the foundation for training. First, I selected and downloaded standard high-resolution image datasets, Places2 (for general scenes) and CelebA-HQ (for faces), to match my target applications. Next, I preprocessed all images by resizing them to a uniform 256x256 resolution and normalizing their pixel values. Finally, I implemented a mask generation strategy, creating both simple rectangular masks and more complex, irregular "free-form" masks to simulate realistic missing regions for the model to learn to fill.

Update : **Kazi Tazrian Mon**

This week, I focused on setting up the development environment and preparing the data pipeline for the inpainting task. The goal was to ensure everything is ready for model training in the coming weeks.

Work Done :

1. Environment Setup :

I successfully set up the necessary development environment, installing essential libraries such as PyTorch, diffusers, and transformers. These libraries are crucial for working with diffusion models and generating images based on input masks.

2. Data Preparation:

For this phase, I used a dataset (like CIFAR-10) for initial testing. I then resized the images to 64x64 pixels and normalized them to fit the needs of the model. The pre-processing was smooth and ready for inpainting tasks. I also implemented a mask generation process, where random square masks of varying sizes are applied to the images. This allows the model to learn how to fill in missing regions.

3. Masking a Hen's Image :

As part of validating the setup, I implemented the masking logic for training the inpainting model. A random square mask was generated and applied to a hen's image. The masked region (set to black) simulates missing content for the model to inpaint.

- **Mask Generation**: Random square masks with varying sizes were applied to the image.
- **Masked Image**: The mask was applied to the hen's image, creating the input for the model.
- **Visualization**: I displayed the original image, the generated mask, and the masked image to ensure the process worked as expected.

The masking logic is now fully functional and ready for model training.

4. Data Loader and Testing

I created a DataLoader to manage batches of images and their corresponding masks. After testing the setup with a sample batch, I confirmed that the system was correctly processing the data. The shapes of the images and masks matched expectations, ensuring everything is ready for the next steps.

With the environment set up and the mask generation working properly, the next step will be to begin training the inpainting model using the prepared data. The goal is to fine-tune the model and start evaluating its ability to fill in missing regions accurately in the following week.