Context Diffusion: In-Context Aware Image Generation

1 Problem Statement

The objective of this project is to implement a **context-aware image generation framework** based on diffusion models. Inspired by recent advances in **latent diffusion models (LDM)** and **few-shot learning**, the model aims to generate images that combine:

- **Structure/layout** derived from a *query image*.
- **Style and texture** drawn from multiple *context images*.
- **Semantic guidance** from an optional *text prompt*.

This approach addresses scenarios where generated images need to adhere to a specified layout while adopting the stylistic elements of provided reference images, a process known as *context-aware image synthesis*.

2 Project Overview

The solution is implemented using:

- **CLIP Encoders** to embed both text prompts and context images.
- A **Latent Diffusion Model (LDM)** from Hugging Face's `diffusers` library for image generation.
- **Dynamic Normalization**: Adaptive normalization values are calculated for each input image, enhancing flexibility in the preprocessing pipeline.

3 How the Code Works

3.1 Key Components

1. **Image and Text Encoding**: The project uses CLIP's pre-trained model to extract embeddings from both text prompts and context images. Text prompts are tokenized and encoded using the CLIP text model, while context images are resized, dynamically normalized, and encoded through the CLIP vision model.
2. **Latent Diffusion Model (LDM)**: The `StableDiffusionPipeline` from `diffusers` is used as the main generative model. This model takes conditioning inputs derived from the query image, context images, and text prompt to produce a final generated image.
3. **Dynamic Image Normalization**: Each context image is normalized based on its own mean and standard deviation, which is dynamically calculated. This approach allows the model to adapt to different image styles without fixed normalization values.

3.2 Code Flow

1. **Preprocessing the Query and Context Images**: Images are resized to 224x224 pixels. Each image undergoes dynamic normalization using its own mean and standard deviation, ensuring consistency in visual appearance without hardcoded values.
2. **Combining Contextual Embeddings**: The CLIP model generates separate embeddings for the text prompt and context images. Context embeddings from multiple images are averaged to create a single style representation, which is then combined with text embeddings to guide the diffusion model.
3. **Image Generation**: The combined embeddings (style from context images, layout from query image, and semantics from text) are fed into the Stable Diffusion pipeline. The model generates an output image that reflects the layout, style, and optional text guidance.

4 Usage Instructions

1. **Install Dependencies**:

```
pip install torch torchvision diffusers transformers
```

2. **Load and Run the Model**: Place your query and context images in the specified paths. Customize the text prompt as desired. Run the code to generate an image.
3. **Example Output**: The generated image will incorporate the layout from the query image and the style from context images, aligned with the optional text prompt.

5 Example Code Snippet

```
query_image = load_image_as_tensor("path/to/query_image.jpg")
context_images = [
    load_image_as_tensor("path/to/context_image1.jpg"),
    load_image_as_tensor("path/to/context_image2.jpg"),
    load_image_as_tensor("path/to/context_image3.jpg")
]
prompt = "A stylized forest with misty mountains."
generated_image = model(query_image, context_images, prompt)

# Display the generated image
plt.imshow(generated_image)
plt.axis('off')
plt.show()
```

6 Future Improvements

- **Training on New Data**: Fine-tuning on specific datasets to enhance style transfer for unique scenarios.
- **Experimentation with Alternative Embeddings**: Testing different embedding models to capture diverse styles and structures.