Assignment: Implementing Components of a DiT-based Diffusion Model in PyTorch

Objective:
To implement key functions and logic for a PyTorch-based diffusion model, focusing on a Diffusion Transformer (DiT) architecture. Follow principles from the DiT paper (https://arxiv.org/abs/2212.09748) and use the Landscape Pictures dataset from Kaggle (https://www.kaggle.com/datasets/arnaud58/landscape-pictures).

Instructions:

Implement the following functions, logic, and scripts in Python using PyTorch.

1. Forward Process and Input Preparation Components:

- Implement a Python function to add noise to images per a diffusion schedule.
- Implement or adapt a standard noise scheduler (e.g., linear, cosine), understanding its configuration.
- Implement a Time Embedding layer for DiT conditioning (e.g., sinusoidal positional embeddings).
- Implement a Patchify layer to convert images into sequences of flattened patches.
- Experiment with various patch sizes (e.g., 2×2, 4×4, 8×8) and note their impact.

2. DiT Model Implementation:

Implement a simple DiT model architecture for DDPM (refer to DiT paper for guidance and configurations).

3. Basic Training Loop for DiT Model:

- Write a Python script for a one-epoch training loop using your DiT model (from Task 2).
- Follow the DiT paper (Peebles & Xie, 2022; https://arxiv.org/abs/2212.09748) for methodology.
- Script essentials:
  - Dataset/DataLoader setup (Kaggle Landscape dataset or similar).
  - Initialize your DiT model (Task 2), noise scheduler (Task 1), and optimizer.
  - Core training steps: noise prediction, loss calculation, backpropagation, optimizer update.
  - Basic loss trend visualization.

4. DiT Model Experiments and Visualization:

- Experiment with the number of diffusion timesteps (e.g., T=100,500,1000, or other values)
- Experiment with the number of DiT blocks in your model and observe the impact on performance/samples. (4, 6, 8 or other values)
- Experiment with the number of attention heads in your model and obser the impact on performance. (1, 2, 4 or other values)
- Utilize a pre-trained DiT-based diffusion model (e.g., FLUX.1-dev or a similar DiT architecture) with a DDIM sampler. Visualize the model's evolving prediction of the clean image (100 images in a 10X10 grid) at various time steps during the reverse sampling process, culminating in the final image.(Make sure to visualise the final approximation at each step without the drift)

5. Classifier-Free Guidance (CFG) Logic and Analysis:

- Implement CFG logic for combining conditional/unconditional predictions. (You may adapt your DiT model (Task 2) or use a pre-trained model. For DiT-based CFG, consider models like FLUX.1-dev or SD with DiT architecture can also be used for demonstration, noting architectural differences).
- Your code should show final noise prediction from conditional/unconditional outputs and guidance scale w, given conditional_prediction and unconditional_prediction.
- Perform a sensitivity analysis for the guidance scale w (e.g., w from 0-10) and observe effects on sample quality/diversity.

Submission Requirements:

- Submit Python code/scripts for all tasks.
- For all analysis on performance use FID.
- Submit a detailed report including:
  - Description of your DiT architecture (Task 2), its components, and design choices.
  - For experiments present 100 images in a 10×10 grid showing parameter impacts.
  - Support visuals with brief quantitative or qualitative observations.
  - Clear code explanations for all tasks.
  - Discussion of significant design choices/assumptions.
  - Summary of challenges and solutions.

NOTE: You may use the provided video as a reference to help your understanding, but please ensure the code you submit is your own original work.