**CSYE7105 33290 HIGH PARALLEL MACHINE LEARNING & AI Proposal**

**Accelerating Diffusion-Based Image Using Parallel Computing**

Team 1

Ao Li, Yulun feng

## **1. Introduction**

Diffusion models generate data by iteratively denoising random noise, often surpassing GANs and VAEs in stability and output quality. However, they require substantial computational power, as each image generation involves tens or hundreds of sequential deep network passes, making them slower to train and sample from. Parallel computing, leveraging GPUs and multi-core processors, is essential for accelerating these operations. Efficient parallelization can significantly speed up training and inference, making diffusion models more practical. This project explores integrating diffusion generative modeling with parallel computing to enhance performance in high-speed image generation.

## **2. Motivation**

Training and sampling from diffusion models on a single hardware setup is highly inefficient. On a CPU, generation is infeasible, and even on a single GPU, training takes days while generating a single image may take seconds or more due to hundreds of neural network iterations. Data pipeline overhead can also bottleneck performance, leaving GPUs underutilized. Parallel computing addresses these issues by enabling efficient data loading, multi-GPU training, and concurrent image generation. This reduces runtimes, enhances scalability, and allows for faster model iteration and real-time generative applications.

## **3. Goal**

This research aims to implement and benchmark parallel computing strategies to accelerate training and image generation in diffusion models while preserving output quality. The project will identify key diffusion pipeline components benefiting from parallelization—such as data preprocessing, training steps, and sampling—and apply appropriate techniques. Effectiveness will be measured by speed-ups in training (e.g., reduced epoch time) and inference (e.g., higher image throughput) compared to a serial baseline. Ensuring acceleration does not compromise image fidelity or diversity is equally crucial. The outcome will provide insights into optimal parallelization strategies for scalable diffusion-based image generation.

## **4. Methodology**

1. **Model Architecture**
   * **U-Net–style Backbone with ResNet Blocks:** A U-Net structure equipped with ResNet blocks handles denoising at each time step, leveraging skip connections to preserve spatial detail.
   * **Attention Mechanism (Low Resolution):** Attention modules are introduced at lower-resolution layers to capture global relationships across regions of the image with reduced computational overhead. Low-resolution stages focus on overall structure, making them ideal for learning broader context.
   * **Potential Extension to Higher Resolution:** If computational resources permit, similar attention mechanisms can be applied at higher-resolution layers to further enhance detail modeling.
   * **Noise Scheduling:** A linear or cosine schedule gradually adds noise during the forward diffusion process, balancing model performance and training stability.
2. **Dataset Handling**
   * Using CIFAR-10 images dataset.
   * **Parallel Data Loading** using multi-threaded PyTorch DataLoader to ensure GPUs stay busy.
3. **Training Flow**
   * Sample a random diffusion step t, add noise to the image, then predict the noise with the model.
   * Optimize via MSE loss between predicted and actual noise.
   * Implemented in PyTorch, potentially with **DDP** (DistributedDataParallel) for multi-GPU runs.
4. **Parallelization Strategies**
   * **Multi-Threaded Data Loading:** Minimizes CPU bottlenecks.
   * **Multi-GPU Training:** Data parallel approach where each GPU handles a portion of the batch.
   * **Batched or Multi-GPU Inference:** Generate many images simultaneously.

## **5. Dataset Information**

* **CIFAR-10:** 
  + 60000 32x32 color images
  + 10 classes, 6000 images per class
  + 50000 training images and 10000 test images.
* **Size:** 162 MB
* **Data Source:** https://www.cs.toronto.edu/~kriz/cifar.html
* **Batches Size:** match GPU memory constraints, 128–256 images per batch.

## **6. Performance Evaluation**

1. **Computation Speedup**
   * Measure time per epoch/training iteration.
   * Compare single vs. multi-GPU training times, including multi-thread vs. single-thread data loading.
   * Compare speedup and efficiency with different numbers of GPUs.
2. **Image Quality**
   * **FID Score** and/or **Inception Score** on generated samples to confirm the model learns effectively.
   * Visual inspection of generated images to check for diversity and clarity.

## **7. Expected Challenges & Possible Future Work**

* **Synchronization Overhead:** Distributed training requires coordinating gradients, which may affect speed gains.
* **Sequential Denoising:** Each image generation has multiple steps; we can only parallelize across multiple images, not within a single image’s steps.
* **Possible Future Extensions:**
  + Try more advanced diffusion techniques (e.g., DDIM) for faster sampling.
  + Scale to larger image sizes or different datasets.
  + Further integrate with HPC cluster technologies (e.g., Slurm for resource management).

## **8. Conclusion**

By leveraging parallel computing on the NEU Explorer, we aim to reduce training and inference times for diffusion-based image generation. Successful outcomes will show improved throughput without compromising the quality of images generated.

## **9. Reference**

* Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems (NeurIPS), 33, 6840–6851. [https://arxiv.org/abs/2006.11239](https://arxiv.org/abs/2006.11239" \t "_new)
* Salimans, T., & Ho, J. (2022). Progressive distillation for fast sampling of diffusion models. arXiv preprint arXiv:2202.00512. [https://arxiv.org/abs/2202.00512](https://arxiv.org/abs/2202.00512" \t "_new)
* Tang, Z., Tang, J., Luo, H., Wang, F., & Chang, T.-H. (2024). Accelerating parallel sampling of diffusion models. arXiv preprint arXiv:2402.09970. [https://arxiv.org/abs/2402.09970](https://arxiv.org/abs/2402.09970" \t "_new)