**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 have emerged as a powerful class of generative AI models that synthesize data (such as images) by iteratively denoising random noise. They often surpass earlier generative approaches (e.g. GANs and VAEs) in stability and output fidelity. However, these benefits come at the cost of extremely high computational intensity. Each generated image requires performing tens or hundreds of sequential denoising steps through a deep network, leading to heavy reuse of models. As a result, diffusion models demand significant processing power and are much slower to train or sample from compared to simpler models. In modern deep learning, parallel computing is indispensable to handle such demands. Graphics Processing Units and multi-core processors enable many operations to run concurrently, dramatically accelerating matrix computations and model. Leveraging parallel computing for diffusion-based image generation is therefore crucial to make these models practical, enabling faster training times and more efficient inference. This project proposes to combine diffusion generative modeling with parallel computing techniques to achieve high-performance image generation.

## 2. Motivation

Training and sampling from diffusion models on standard single-hardware setups poses serious challenges. On a single CPU, generation is practically infeasible due to the sequential nature of diffusion steps. Even on a single GPU, training a diffusion model can take days, and generating a high-quality image may take several seconds or more per image. These delays stem from the need to iterate over the neural network hundreds of times for one sample, as well as processing large datasets for training. Moreover, data pipeline overhead (loading and preprocessing images) can become a bottleneck if not managed in parallel. Without parallelization, the GPU may sit idle waiting for data, or multiple GPUs may be underutilized. By introducing parallel computing at various stages, we can tackle these issues. For example, parallel data loading can feed the model efficiently despite limited disk I/O or CPU speed, and multi-GPU training can distribute the workload of gradient computations. There are clear potential benefits to parallelizing different components: speeding up *data loading* via multi-threading, accelerating *model training* through distributed computing, and boosting *inference* throughput by generating multiple images in parallel. Faster training means researchers can iterate on models more quickly, and faster image generation opens the door to real-time or interactive generative applications. In summary, parallelization promises to alleviate the long runtimes and hardware strain associated with diffusion models, making diffusion-based image generation more scalable and accessible.

## 3. Goal

The primary goal of this research is to implement and benchmark parallel computing strategies that improve the speed of both training and image generation in diffusion models. We aim to achieve significant reductions in runtime without sacrificing model performance or output quality. Concretely, the project will identify key components of the diffusion model pipeline that benefit from parallelization (such as data preprocessing, neural network training steps, and sampling procedure) and apply appropriate parallel computing techniques to each. We will then quantitatively evaluate which parallelization strategies are most effective for which parts of the workflow. Success will be measured by achieving notable speed-ups in training (e.g. reduced time per epoch) and inference (e.g. more images generated per unit time) when using parallel methods, compared to a baseline serial implementation. An equally important goal is to maintain or improve the image quality of the generated outputs – ensuring that acceleration techniques (like larger batch sizes or distributed training) do not degrade metrics like fidelity or diversity. By the end of the project, we expect to have a clear understanding of the most effective parallel computing approaches for diffusion-based image generation, and guidelines for how to scale such models efficiently on modern parallel hardware.

## 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>
* Salimans, T., & Ho, J. (2022). *Progressive distillation for fast sampling of diffusion models*. arXiv preprint arXiv:2202.00512. <https://arxiv.org/abs/2202.00512>
* 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>
* Zhang, H., Saxena, S., Johnson, J., Shih, K. J., Zhang, Z., & Goyal, P. (2022). Training compute-optimal large language models (arXiv:2212.09748). arXiv. <https://doi.org/10.48550/arXiv.2212.09748>
* Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Henighan, T., Hennigan, T., … & Irving, G. (2022). Training compute-optimal large language models (arXiv:2105.05233). arXiv. <https://doi.org/10.48550/arXiv.2105.05233>