CS 7389D
Scalable Deep Learning with Distributed Training

By: Sai Sri Ganesh Putsala, Savyasachi Thati, Bojanapatti Neerajakshulu Puneeth, Lingareddy, Sai Mahalakshmi Reddy

1. Introduction

Deep learning models have revolutionized fields like computer vision, natural language processing, and generative models. Recent advances, such as DALL·E and ChatGPT, highlight the tremendous potential of large-scale AI systems. However, as models become more complex and datasets grow, training these models on a single GPU becomes inefficient due to memory and computational constraints. This project addresses these challenges by utilizing distributed training strategies to enhance scalability and efficiency in deep learning models. By focusing on text-to-image generation, we aim to implement and scale up a deep learning pipeline that integrates a Conditional Variational Autoencoder (CVAE) with a diffusion model, using parallel computing frameworks like PyTorch, CLIP, and Fully Sharded Data Parallelism (FSDP).

2. Project Goals

The main objective of this project was to implement a text-to-image generative model using the CVAE architecture. We aimed to generate high-quality images conditioned on textual input, leveraging the CLIP embeddings for semantic control. The project focused on evaluating the scalability of different distributed training methods: Data Parallelism (DP), Distributed Data Parallel (DDP), and Fully Sharded Data Parallel (FSDP). The performance of each strategy was measured in terms of GPU utilization, training time, memory usage, and power efficiency.
Key milestones include:

- Implementing the text-to-image generative model.
- Scaling the model's training process using distributed methods.
- Training the model on TACC's Frontera supercomputer to gain hands-on experience with high-performance computing (HPC) systems.

3. Methodology & Architecture

The methodology used for this project involves the following key components:

- **Model Architecture:**
  The architecture integrates a CVAE with a diffusion model for text-based image

generation. CLIP embeddings are used to condition both the encoder and decoder of the CVAE, providing semantic control for image generation.

- **Distributed Training Strategies:**
  We explored three primary parallelism techniques:
  - **Data Parallelism (DP):** This technique replicates the model across GPUs, with each GPU handling a subset of the data. It is simple to implement but suffers from high communication overhead.
  - **Distributed Data Parallelism (DDP):** The model is replicated across GPUs, and gradients are synchronized via all-reduce operations. This method improves scalability and GPU utilization.
  - **Fully Sharded Data Parallelism (FSDP):** This advanced technique shards the model's parameters, gradients, and optimizer states across GPUs, providing memory-efficient training, particularly useful for large models.

High-Performance Computing Infrastructure:
We utilized the TACC Frontera supercomputer, leveraging SLURM job scheduling to manage distributed training processes. NCCL was used for efficient GPU-to-GPU communication, enabling the parallel training of the model across multiple GPUs

4. Implementation & Training Strategy

The model was trained on the CIFAR-10 dataset, with images resized from 32x32 to 512x512 for higher resolution. The training loop utilized the following techniques for optimization:

- **Mixed Precision Training:** This reduces memory usage by employing float16 precision where safe, improving computational efficiency.
- **Checkpointing:** This technique helps manage memory efficiently by recomputing activations during the backward pass, allowing for the handling of larger images and reducing GPU memory usage.

Training involved iterating over epochs where random latent vectors, combined with text labels, were used to generate new samples. The loss function employed a combination of L1 loss (reconstruction error) and Kullback-Leibler divergence (KLD) to ensure high-quality image generation while regularizing the latent space.

5. Experimental Setup

The training process was carried out on multiple GPUs using the SLURM scheduler. For distributed training, each GPU was assigned a unique process and local rank. Parameters were synchronized across GPUs at the start of the training, and FSDP was used to shard model parameters and gradients. A mixed precision approach, combined with gradient checkpointing, allowed for efficient use of GPU memory, enabling the training of the model at high resolution (512x512).
Key experimental components include:

- **Dataset:** CIFAR-10 dataset, which was preprocessed to ensure consistency in image size and class labels.
- **Loss Function:** L1 loss for pixel-wise reconstruction and KLD for latent space regularization.
- **Training Loop:** Epoch-wise loss tracking, backpropagation, and optimizer updates.

6. Results & Evaluation

6.1 Training Loss:

The training loss steadily decreased throughout the epochs, indicating that the model was learning. Early epochs showed rapid improvement, while later epochs fine-tuned the model for better image quality. The combined loss function (L1 + KLD) showed a stable downward trend, suggesting effective learning.

6.2 Generated Sample Images:

Generated images improved over time, with better alignment to input text labels (e.g., "dog," "airplane"). Initially, images were blurry, but with further training, they became sharper and clearer. By the end, the model was consistently generating well-formed images with fine details.

6.3 GPU Memory and Training Time:

GPU memory was efficiently used, with training at 512x512 resolution occupying only 40-50% of memory, thanks to techniques like FSDP and mixed precision training. Training time per epoch stabilized at around 20 seconds after the initial setup.

6.4 Scalability and Power Efficiency:

The model showed stable scalability across multiple GPUs, with FSDP providing the best memory efficiency. The use of mixed precision and gradient checkpointing reduced both training time and power consumption, making the process more efficient overall.

7. Challenges & Limitations

Several challenges were encountered during the project:

- **FSDP Synchronization:** Proper synchronization of model parameters across GPUs was tricky, particularly when resuming from checkpoints or switching devices.
- **Large Image Resolution:** Training at 512x512 resolution caused significant memory stress, especially during gradient accumulation and optimizer updates.
- **Distributed Debugging:** Debugging distributed training processes, especially when errors occurred on individual ranks, proved to be time-consuming.
- Despite these challenges, the project successfully demonstrated the scalability of distributed training and the feasibility of training large models using FSDP.

8. Future Work

Future work will focus on:

- **Integrating a Diffusion Model:** To improve image quality, we plan to integrate a denoising diffusion module that will allow for higher fidelity in image generation.
- **Implementing DeepSpeed:** We will explore Microsoft's DeepSpeed library to further enhance scalability and efficiency, particularly for larger models.
- **Training on Larger Datasets:** Expanding the training to datasets like MS-COCO and ImageNet will allow for more meaningful text-to-image generation and improved generalization.

9. Conclusion

The project achieved the goal of generating high-resolution images conditioned on text labels using a distributed CVAE model. The training was efficiently scaled across multiple GPUs using FSDP, with memory usage optimized through mixed precision and gradient checkpointing. The hands-on experience gained in configuring distributed training pipelines and troubleshooting large-scale training issues has provided valuable insights into the practical challenges and solutions in scalable deep learning.