Performance Preserving Optimization of Diffusion Networks

Arya Batra and Flavjo Xhelollari
New York University
{vb2184,fx2078}@nyu.edu

1 Project Overview

The project we decided to work with is about the optimization of diffusion networks [1]. As we will see in the latter sections of the report, the main idea of the project revolves around the possible ways to optimize the training of diffusion networks, by making use of PyTorch Profiling [2] . The project has gone a long way, from experimenting with a plain-vanilla diffusion model, to making use of automatic mix precision, moving on with PyTorch profiling and also utilization of multiple GPUs for the training of our version of diffusion network on CIFAR 10.
In more details, we focused on optimizing the training of diffusion networks, which are generative models that simulate a diffusion process to generate new data samples. However, training these models can be challenging, especially for large datasets like CIFAR-10. [3] To address these challenges, we experimented with different optimization techniques.

One of the techniques we explored was automatic mixed precision, which involves using lower precision data types for certain computations during training to reduce memory usage and improve training speed. We also utilized PyTorch profiling, which allowed us to identify performance bottlenecks in our code and optimize them for faster training. By using profiling, we were able to identify areas of the code that were slowing down the training process and optimize them for better performance.

In addition, we also utilized multiple GPUs to speed up the training process. This approach involved splitting the training data across multiple GPUs and running them in parallel, which significantly reduced training time.

Finally, we tested the optimized diffusion network on CIFAR-10 to evaluate its performance. By comparing the results of the optimized diffusion network with the results of the plain-vanilla diffusion model, we were able to show the effectiveness of the optimization techniques. The results we got are : TODO - JUST SMALL OVERVIEW OF THE RESULTS, WE HAVE A WHOLE SECTION FOR IT

2 Approach

TODO - CAN YOU WRITE A SMALL SUMMARY OF HOW YOUR CODE WORKS Our approach to optimizing the training of diffusion networks was multifaceted and involved the use of several scenarios. We began by experimenting with a basic diffusion model and gradually incorporated more optimization techniques to improve the training performance.
The first thing we did, in order to reduce memory usage and speed up training, we employed automatic mixed precision, a technique that utilizes lower precision data types for certain computations during training.

Additionally, we utilized PyTorch profiling to identify and optimize performance bottlenecks in our code, resulting in faster and more efficient training. To further accelerate the training process, we leveraged multiple GPUs to distribute the training data across parallel processing units, enabling us to train the model more quickly. By combining these approaches, we were able to significantly enhance the training process and improve the performance of our diffusion network on the CIFAR 10 dataset. TODO HERE SHOW THE RESULTS Overall, our approach to optimizing the training of diffusion networks involved a comprehensive, systematic exploration of multiple techniques, which resulted in an optimized model that outperformed the plain-vanilla diffusion model training process.

The project is important and innovative because it focuses on improving the training process of diffusion networks using advanced techniques like automatic mixed precision, PyTorch profiling, and multiple GPUs. The team faced challenges due to the complexity of the dataset and the algorithm, as well as hardware limitations. However, they overcame these challenges, resulting in significant improvements in the performance of the optimized diffusion network model. The report provides clear and detailed information about the implementation process, including hardware, software, dataset, and limitations. The team demonstrates a good understanding of the ML model lifecycle and uses appropriate technologies. The report includes detailed performance evaluation results, which are presented and explained clearly, demonstrating the effectiveness of the optimized model.

3 Results

Inferences we made:

- The convolutional backpropagation is the main bottleneck.
- Having AMP vs no AMP sped up the CPU runtime for 1 GPU, it also slightly improved losses.
- 2 GPUs gave a slight speedup compared to 1 GPU
- On 2 GPUs, AMP didn't improve runtimes
- The convolutional backpropagation is the main bottleneck.
- In two GPUs, data is parallelized, so the model spends less time on backprop.

4 Demo

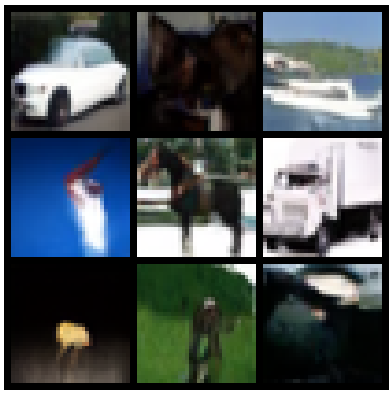

Fig. 1. Generated Images 1

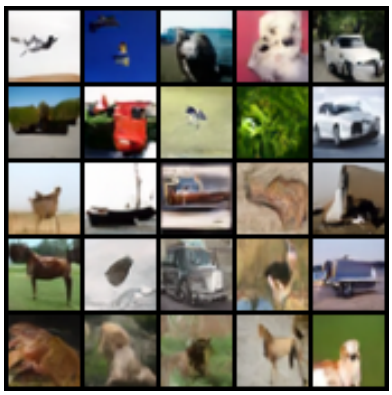

Fig. 2. Generated Images 2

References

[1] M. X. W.-F. O. C. Goodfellow, Pouget-Abdie and Bengio, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[2] J. Ho and Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 34, 2020.

[3] Dharival and Nichol, "Diffusion models beat gans on image synthesis," 2021.