Performance Preserving Optimization of Diffusion Networks

Arya Batra and Flavjo Xhelollari
New York University
vb2184 and fx2078

1 Goals and Objectives

Diffusion models [1] are a powerful deep learning tool used for data generation. They improve upon Generative Adversarial Networks [2] with better training stability as we don't have two models competing against each other. Instead, they are built based on Markov Chains that learn to convert images to noise; thus enabling the model to generate images from noise in the opposite direction. Images produced by diffusion models are also better in quality and diversity of images synthesisized [3].
However, diffusion models are built on U-Net [4] which use [5] ResNet blocks acting at multiple layers with an upsampling and downsampling process. As a result of implementing U-Net, our model is extremely computationally complex, requiring long training times and heavy memory utilization.
Current solutions to this problem either attempt to solve it through reducing run-time at potential cost to performance. The authors of [6] propose using half-precision floats instead to decrease compute time of floating point operations at the expense of precision, thus potentially affecting performance. Another approaches look to optimize code through scaling using parallization across multiple GPUs [7]
In this project, our goal is consider various techniques and even propose methods learned in class to achieve a performance-cost evaluation between resource utilization, compute times and overall quality of results.

2 Challenges

The biggest challenge we anticipate is the diffusion model's resource requirements. Due to the expensive nature of running a ResNet [5] based U-Net [4] model, the diffusion model requires a lot of computational resources and time to produce high quality results.
These challenges will make running experiments expensive. That is the biggest challenge to overcome. Due to this constraint, we will also use simpler tracing tools with minimal overhead to prevent a further increase in our runtimes. We will also be using CIFAR-10 as it is a smaller dataset we can use to prove the efficacy of our approach.

3 Approaches and Techniques

Our goal is to improve performance without compromising on model performance. To do so, we use the Frechet Inception Distance (FID) [8] metric as a measure of quality. FID is a metric used in image generation applications that measures the quality and diversity of synthetic images. We measure each of our models with FID to measure if our changes have a significant affect on the model's ability to produce high quality images. Our metrics include

FID scores, runtime and resource utilization; measured against the vanilla diffusion model as our benchmark. In the process of benchmarking and tracing, we will also measure the arithmetic intensity of our program to assess performance and bottlenecks.
For our purposes, we attempt the following implementations based on things we learning in HPML class along with existing solutions:

- Parallelizing U-Net across multiple GPUs.[7]
- Including 1x1 bottleneck layers as seen in InceptionNet [9] between our ResNet blocks to reduce number of computations to speed up processing.
- Reducing the number of sampling blocks during training as it is a bottleneck that negatively affect training times.
- Attempting Im2Col implementations instead of naive convolutions to reduce the computational intensity.
- Replacing the traditional implementation with half precision floating points to decrease the cost of each floating point computation.
- Measure performance across available GPUs to compare the effects of GPU on training times.

4 Implementation

All our code will be run on NYU HPC using GPUs. Profiling is done using cPython. Our code base is from [10] that runs code using PyTorch.
For our dataset, in order to keep runtimes in check, we train or model on CIFAR-10 which is reasonably sized and allows to run experiments in a timely fashion.

5 Demos

Using our code base, we implemented a diffusion model for CIFAR-10 to confirm the implement-ability of diffusion networks for their intended purpose on this project.

From our experiments, we can see that diffusion models are capable of producing high quality samples when trained and evaluated on a CIFAR-10 dataset. After initial experimentation, we will work to make these models more efficient and usable.

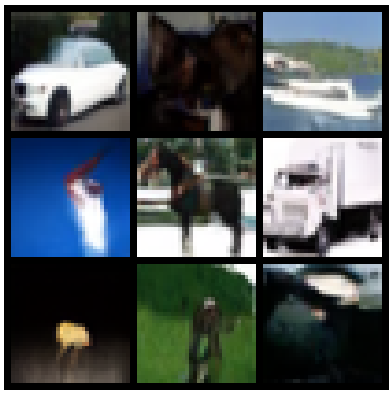

Fig. 1. Generated Images 1

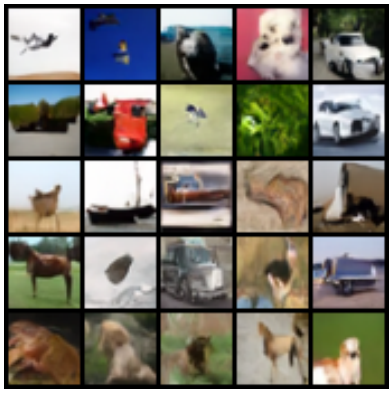

Fig. 2. Generated Images 2

References

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

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

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

[4] F. Ronnenberger and Brox, "U-net: Convolutional networks for biomedical image segmentation," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[5] R. He, Zhang and Sun, "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] Foong, "How to optimize stable diffusion for image generation," https://betterprogramming.pub/how-to-optimize-stable-diffusion-for-image-generation-d238dad644a3, 2023.

[7] Dave and Bowne-Anderson, "Parallelizing stable diffusion for production use cases," https://outerbounds.com/blog/parallelizing-stable-diffusion-production-use-cases/, 2022.

[8] U. N. Heusel, Ramsauer and Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[9] J. S. R. A.-E. V. Szegedy, Liu and Rabinovic, "Going deeper with convolutions," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[10] Lucidrains, "denoising-diffusion-pytorch," https://github.com/lucidrains/denoising-diffusion-pytorch/tree/main/denoising_diffusion_pytorch, 2023.