Performance Preserving Optimization of Diffusion Networks

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

PROJECT OVERVIEW

Optimization of Diffusion Networks

- The project is focused on optimizing the training process of diffusion networks
- The main objective is to identify possible ways to optimize the training of diffusion networks by using PyTorch Profiling
- The project has undergone significant changes, from experimenting with a plain-vanilla diffusion model to utilizing advanced techniques like automatic mix precision and multiple GPUs.

PROJECT OVERVIEW

Techniques Used for Optimizing Training of Diffusion Networks

- The project experimented with different optimization techniques to address challenges in training diffusion networks
- Automatic mixed precision was used to reduce memory usage and improve training speed
- PyTorch profiling was utilized to identify performance bottlenecks and optimize them for faster training
- Multiple GPUs were used to speed up the training process by running them in parallel
- The optimized diffusion network was tested on CIFAR-10 to evaluate its performance, which demonstrated the effectiveness of the optimization techniques.

APPROACH

Optimization of Diffusion Networks

- The project focuses on optimizing the training of diffusion networks using PyTorch profiling
- Started with a basic diffusion model and experimented with optimization techniques to improve training performance
- Utilized automatic mixed precision to reduce memory usage and speed up training
- Used PyTorch profiling to identify and optimize performance bottlenecks in the code
- Utilized multiple GPUs to distribute the training data across parallel processing units for faster training
- Optimized diffusion network tested on CIFAR 10 for performance evaluation

APPROACH

Multifaceted Approach to Optimizing Diffusion Network Training

- Approach to optimizing the training of diffusion networks involved multiple techniques:
- Started with a basic diffusion model and gradually incorporated more optimization techniques
- Automatic mixed precision employed to reduce memory usage and speed up training
- PyTorch profiling used to identify and optimize performance bottlenecks in the code
- Multiple GPUs (up to 2) utilized to distribute training data for faster training
- Combination of these techniques significantly enhanced the training process and improved diffusion network performance on CIFAR 10 dataset

Profiling Results - (1 GPU with AMP)

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	CPU Mem	Self CPU Mem	CUDA Mem	Self CUDA Mem	# of Calls
cudaLaunchKernel	15.78%	4.468s	15.78%	4.468s	5.913us	2.877s	11.27%	2.888s	3.822us	0 b	0 b	-262.66 Mb	-262.66 Mb	755631
aten::convolution_backward	9.54%	2.703s	14.06%	3.982s	337.466us	5.674s	22.22%	6.935s	587.672us	0 b	0 b	137.19 Gb	-255.01 Gb	11800
cudaMemcpyAsync	8.43%	2.386s	8.43%	2.386s	759.431us	9.403ms	0.04%	9.403ms	2.993us	0 b	0 b	0 b	0 b	3142
cudaStreamSynchronize	6.82%	1.933s	6.82%	1.933s	3.850ms	1.882ms	0.01%	1.882ms	3.749us	0 b	0 b	0 b	0 b	502
Optimizer.step#Adam.step	4.58%	1.297s	13.61%	3.853s	38.533ms	0.000us	0.00%	1.152s	11.515ms	524 b	11.11 Kb	80.84 Mb	-7.64 Gb	100
aten::cudnn_convolution	4.12%	1.166s	8.79%	2.490s	210.991us	2.333s	9.13%	2.713s	229.898us	0 b	0 b	136.67 Gb	-30.13 Gb	11800
enumerate(DataLoader)#_MultiProcessingDataLoaderIter...	2.96%	837.211ms	3.09%	874.431ms	4.350ms	0.000us	0.00%	1.775ms	8.831us	-804 b	-30.18 Kb	0 b	0 b	201
aten::add_	2.70%	764.770ms	5.31%	1.504s	11.957us	703.607ms	2.76%	1.077s	8.562us	-16.83 Kb	-113.85 Kb	-6.60 Mb	-6.60 Mb	125769
aten::copy_	2.65%	751.568ms	11.89%	3.369s	24.456us	3.707s	14.52%	4.146s	30.103us	0 b	0 b	0 b	0 b	137742
aten::_to_copy	2.49%	706.336ms	15.84%	4.486s	34.697us	0.000us	0.00%	3.574s	27.643us	90.23 Kb	5.65 Kb	596.31 Gb	-3.91 Mb	129302

Self CPU time total: 28.320s
Self CUDA time total: 25.535s

Total runtime was: 502.48211908340454

Profiling Results - (1 GPU without AMP)

training complete

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	CPU Mem	Self CPU Mem	CUDA Mem	Self CUDA Mem	# of Calls
cudaLaunchKernel	30.42%	12.902s	30.70%	13.020s	20.198us	4.710s	11.48%	4.724s	7.328us	-40 b	-40 b	-310.93 Mb	-310.93 Mb	644631
cudaStreamSynchronize	18.16%	7.701s	18.16%	7.701s	15.341ms	2.993ms	0.01%	2.993ms	5.962us	0 b	0 b	0 b	0 b	502
cudaMemcpyAsync	13.35%	5.664s	13.35%	5.664s	1.803ms	37.406ms	0.09%	37.406ms	11.905us	0 b	0 b	0 b	0 b	3142
aten::convolution_backward	5.02%	2.128s	10.49%	4.451s	377.215us	19.073s	46.50%	22.296s	1.889ms	0 b	0 b	273.99 Gb	-1350.05 Gb	11800
Optimizer.step#Adam.step	2.97%	1.259s	21.22%	9.002s	90.019ms	0.000us	0.00%	1.806s	18.064ms	524 b	10.86 Kb	80.21 Mb	-7.54 Gb	100
aten::cudnn_convolution	2.34%	994.281ms	5.37%	2.276s	192.884us	4.718s	11.50%	5.491s	465.321us	0 b	0 b	273.05 Gb	-113.71 Gb	11800
cudaMemsetAsync	2.06%	875.094ms	2.06%	875.094ms	43.317us	141.697ms	0.35%	141.697ms	7.014us	0 b	0 b	879.50 Kb	879.50 Kb	20202
aten::add_	1.74%	737.246ms	8.37%	3.551s	28.231us	1.027s	2.50%	1.840s	14.632us	-15.95 Kb	-109.78 Kb	-3.50 Kb	-3.50 Kb	125769
enumerate(DataLoader)#_MultiProcessingDataLoaderIter...	1.67%	706.254ms	1.74%	739.554ms	3.679ms	0.000us	0.00%	1.000ms	4.975us	-804 b	-31.46 Kb	0 b	0 b	201
aten::sum	1.25%	530.440ms	6.14%	2.604s	85.112us	1.222s	2.98%	1.489s	48.652us	0 b	0 b	427.89 Mb	427.08 Mb	30600

Self CPU time total: 42.417s
Self CUDA time total: 41.019s

Total runtime was: 428.0465748310089

2 GPU with AMP

training complete

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	CPU Mem	Self CPU Mem	CUDA Mem	Self CUDA Mem	# of Calls
cudaLaunchKernel	19.06%	7.576s	19.06%	7.576s	10.026us	2.872s	11.26%	2.879s	3.810us	0 b	0 b	-217.36 Mb	-217.36 Mb	755631
cudaFree	13.36%	5.310s	14.44%	5.738s	717.254ms	2.000us	0.00%	414.000us	51.750us	0 b	0 b	0 b	0 b	8
aten::convolution_backward	7.30%	2.902s	11.78%	4.682s	396.746us	5.667s	22.21%	6.889s	583.784us	0 b	0 b	137.19 Gb	-255.15 Gb	11800
cudaStreamSynchronize	6.74%	2.680s	6.74%	2.680s	5.339ms	1.455ms	0.01%	1.615ms	3.217us	0 b	0 b	0 b	0 b	502
aten::cudnn_convolution	6.17%	2.450s	29.92%	11.889s	1.008ms	2.327s	9.12%	2.718s	230.307us	0 b	0 b	136.67 Gb	-30.13 Gb	11800
Optimizer.step#Adam.step	4.36%	1.731s	11.82%	4.698s	46.980ms	0.000us	0.00%	1.136s	11.361ms	524 b	-22.66 Kb	80.84 Mb	-7.64 Gb	100
cudaMemcpyAsync	3.99%	1.586s	3.99%	1.586s	504.917us	12.612ms	0.05%	12.612ms	4.014us	0 b	0 b	0 b	0 b	3142
enumerate(DataLoader)#_MultiProcessingDataLoaderIter...	2.26%	900.012ms	2.42%	961.721ms	4.785ms	0.000us	0.00%	2.397ms	11.925us	-804 b	-40.49 Kb	0 b	0 b	201
aten::add_	2.17%	861.586ms	4.10%	1.629s	12.955us	702.945ms	2.76%	1.070s	8.511us	4.57 Kb	-88.10 Kb	-111.50 Kb	-111.50 Kb	125769
aten::_to_copy	2.07%	824.545ms	13.75%	5.465s	42.264us	0.000us	0.00%	3.558s	27.518us	90.23 Kb	30.02 Kb	596.31 Gb	-6.41 Mb	129302

Self CPU time total: 39.740s
Self CUDA time total: 25.513s

Total runtime was: 572.3351054191589

2 GPU no AMP

training complete

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	CPU Mem	Self CPU Mem	CUDA Mem	Self CUDA Mem	# of Calls
cudaLaunchKernel	19.90%	7.609s	19.90%	7.609s	10.069us	2.875s	11.28%	2.884s	3.817us	0 b	0 b	-196.68 Mb	-196.68 Mb	755631
cudaFree	14.02%	5.360s	15.27%	5.837s	729.640ms	2.000us	0.00%	766.000us	95.750us	0 b	0 b	0 b	0 b	8
aten::convolution_backward	7.77%	2.972s	12.66%	4.841s	410.285us	5.664s	22.23%	6.883s	583.267us	0 b	0 b	137.19 Gb	-255.16 Gb	11800
aten::cudnn_convolution	6.39%	2.445s	31.51%	12.043s	1.021ms	2.328s	9.14%	2.719s	230.410us	0 b	0 b	136.67 Gb	-30.13 Gb	11800
cudaStreamSynchronize	6.39%	2.443s	6.39%	2.443s	4.867ms	1.339ms	0.01%	1.451ms	2.890us	0 b	0 b	0 b	0 b	502
cudaMemcpyAsync	5.33%	2.036s	5.33%	2.036s	647.941us	12.635ms	0.05%	12.635ms	4.021us	0 b	0 b	0 b	0 b	3142
Optimizer.step#Adam.step	3.25%	1.242s	10.20%	3.898s	38.978ms	0.000us	0.00%	1.128s	11.280ms	524 b	2.79 Kb	80.84 Mb	-7.64 Gb	100
cudaHostAlloc	2.20%	842.557ms	2.20%	842.591ms	8.511ms	1.451ms	0.01%	1.451ms	14.657us	0 b	0 b	1.50 Mb	1.50 Mb	99
aten::add_	1.95%	744.077ms	3.91%	1.495s	11.889us	695.977ms	2.73%	1.071s	8.512us	-10.31 Kb	-101.45 Kb	-78.50 Kb	-78.50 Kb	125769
aten::copy_	1.93%	736.864ms	10.06%	3.846s	27.923us	3.701s	14.52%	4.148s	30.112us	0 b	0 b	0 b	0 b	137742

Self CPU time total: 38.226s
Self CUDA time total: 25.481s

Total runtime was: 559.5399775505066

RESULTS

Inferences we made:

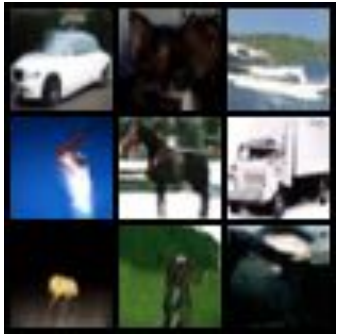
1. The convolutional backpropagation is the main bottleneck.
2. Having AMP vs no AMP sped up the CPU runtime for 1 GPU, it also slightly improved losses.
3. 2 GPUs gave a slight speedup compared to 1 GPU
4. On 2 GPUs, AMP didn’t improve runtimes
5. The convolutional backpropagation is the main bottleneck.
6. In two GPUs, data is parallelized, so the model spends less time on backprop.

Results

Thing measured	1GPU No AMP	1 GPU AMP	2 GPU no AMP	2GPU Amp
Convolution BP	19s	6.9s	6.883s	6.889
Synchronize	2.9ms	1.882ms	1.451ms	1.615ms

VISUALS

Fig. 1



GitHub Repo

Link:
https://github.com/VidushB/Performance-Preserving-Optimization-of-Diffusion-Networks/blob/main/README.md

THANK YOU!
****

Q&A