NYU

Profiling EfficientNets on Multi-GPU Systems

SOMIK DHAR(sd5023)
PRATHAM MEHTA(pm3483)

HPML Project

Executive Summary

Problem Statement (Goal): Evaluate and profile the computational performance of EfficientNets B0, B3 and B5 on 1, 2 and 4 GPUs. Identify optimization opportunities from the profiled results.

Solution Approach: Implementing distributed training with a focus on GPU utilization, utilizing datasets like CIFAR-100 and Food101, and profiling with PyTorch.

Value/Benefit: Improved model efficiency, reduced training time, and scalability for real-world datasets.

Technical Challenges

Resource Contention: Addressing potential issues such as GPU memory constraints, I/O bottlenecks, and network bandwidth limitations that can affect training efficiency.

Fine-Tuning Complex Models: Adjusting pre-trained EfficientNet models to new datasets while maintaining the balance between transfer learning benefits and overfitting risks

Data Heterogeneity: Adapting models to the specific characteristics of CIFAR-100 and Food-101, which may differ significantly from ImageNet in terms of image size, quality, and content diversity

Transfer Learning Efficiency: Determining the optimal number of layers to freeze or fine-tune to maximize knowledge transfer without incurring unnecessary computational costs

Approach

Transfer Learning: Utilizing pre-trained ImageNet weights for each EfficientNet model (B0, B3, B5)

Dataset Preparation: Standardize image sizes by resizing and cropping to facilitate uniform model input. Apply random horizontal flips, rotations, and crops. 80-20 split for cifar100 and 75-25 split for food101 dataset

Adapting the Classifier to New Domains

Setup: Begin with an ImageNet pre-trained EfficientNet, excluding the classifier.

Rationale: Considering time constraints opted to train only the classifier layer, leveraging pre-optimized internal layers for rapid adaptation to new datasets.

Customization: Replace the original classifier with a new one for CIFAR-100 (100 classes) and Food-101 (101 classes), initializing weights accordingly.

Training: Freeze pre-trained convolutional base weights. Train only the new classifier layer, expediting the process by updating classifier weights during backpropagation.

Approach

Parallel Training Setup: Leveraging PyTorch's DataParallel, we replicated our EfficientNet model across GPUs for efficient classifier layer training. This minimized communication overhead, optimizing GPU utilization and minimizing training time.

Training Parameters: used a fixed batch size = 64 i.e. strong scaling in the view of time constraints and memory constraints and used cosine annealing learning rate scheduler

Overcoming Parallel Training Challenges: Managing requires_grad in PyTorch prevented unintended updates to frozen layers during backpropagation. We addressed workload imbalance by tuning data loaders and batch samplers for uniform data distribution across GPUs, enhancing training efficiency.

Approach

Maximizing Efficiency with PyTorch Profiler

- **Essential Analysis Tool:** Streamlines performance analysis of PyTorch models by tracking both CPU and GPU activities.
- **Seamless Integration**: Easily integrates into training code as a context manager, focusing on the 'main_worker' function for comprehensive profiling.
- **In-Depth Monitoring:** Targets crucial training phases including data loading, forward/backward passes, and optimization, using ProfilerActivity.CPU and CUDA.
- **Precise Profiling:** Records tensor shapes (`record_shapes=True`) and wraps the entire training loop for detailed analysis and latency identification.
- **Interpreting Results:** Profiler outputs are visualized in a sorted table, focusing on GPU utilization (`cuda_time_total`) to highlight key performance aspects.
- **Advantages and Considerations:** Essential for debugging and performance tuning, with minimal overhead and manageable complexity, even in larger models.

Approach

Empowering Data-Driven Model Optimization

- **Best Practices**: Profiling as an integral part of the deep learning development cycle, crucial for building faster and more efficient models.

- **Impact on Development**: Encourages a data-driven approach for performance optimization, aiding in resource allocation and model scalability.

Summary of Main Results

Profiling Results on B0 using 1 GPU:

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	# of Calls
main_worker	1.24%	8.162s	99.85%	656.728s	656.728s	0.000us	0.00%	225.503s	225.503s	1
DataParallel.forward	4.01%	26.377s	14.81%	97.378s	20.544ms	0.000us	0.00%	220.769s	46.576ms	4740
aten::_convolution	0.43%	2.829s	4.77%	31.358s	81.675us	0.000us	0.00%	78.879s	205.446us	383940
aten::convolution	0.28%	1.843s	5.01%	32.969s	85.871us	0.000us	0.00%	78.140s	203.522us	383940
aten::conv2d	0.26%	1.700s	5.19%	34.117s	88.861us	0.000us	0.00%	76.857s	200.181us	383940
aten::cudnn_batch_norm	1.08%	7.132s	2.69%	17.675s	76.101us	56.392s	28.33%	62.367s	268.523us	232260
aten::batch_norm	0.08%	544.737ms	2.92%	19.200s	82.664us	0.000us	0.00%	62.254s	268.037us	232260
aten::_batch_norm_impl_index	0.15%	1.014s	2.81%	18.471s	79.527us	0.000us	0.00%	61.449s	264.570us	232260
aten::cudnn_convolution	2.15%	14.112s	3.65%	23.988s	77.858us	35.927s	18.05%	45.251s	146.873us	308100
aten::silu_	0.30%	1.941s	0.50%	3.271s	14.083us	32.567s	16.36%	34.036s	146.543us	232260

Summary of Main Results

Profiling Results on B3 using 1 GPU:

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	# of Calls
main_worker	2.09%	18.725s	99.83%	893.044s	893.044s	0.000us	0.00%	315.355s	315.355s	1
DataParallel.forward	5.44%	48.698s	19.86%	177.614s	37.471ms	0.000us	0.00%	310.593s	65.526ms	4740
aten::_convolution	0.57%	5.120s	5.89%	52.681s	85.494us	0.000us	0.00%	136.079s	220.836us	616200
aten::convolution	0.41%	3.628s	6.23%	55.709s	90.407us	0.000us	0.00%	133.978s	217.425us	616200
aten::conv2d	0.31%	2.784s	6.48%	57.966s	94.071us	0.000us	0.00%	132.900s	215.677us	616200
aten::cudnn_batch_norm	1.47%	13.149s	3.88%	34.678s	93.795us	80.660s	29.31%	89.092s	240.972us	369720
aten::batch_norm	0.15%	1.382s	4.21%	37.679s	101.913us	0.000us	0.00%	88.980s	240.667us	369720
aten::_batch_norm_impl_index	0.19%	1.660s	4.05%	36.197s	97.903us	0.000us	0.00%	88.613s	239.677us	369720
aten::_conv_depthwise2d	0.20%	1.804s	0.46%	4.141s	33.599us	76.220s	27.70%	76.922s	624.160us	123240
aten::cudnn_convolution	3.00%	26.851s	4.36%	39.041s	79.197us	40.490s	14.71%	57.750s	117.149us	492960

Summary of Main Results

Profiling Results on B0 using 2 GPUs:

Name	Self CPU %	Self CPU total	CPU total %	CPU total	CPU time avg	Self CUDA %	Self CUDA total	CUDA total %	CUDA total	CUDA time avg	# of Call
main_worker	2.0%	177.97s	99.8%	177.982s	177.982s	1.0%	55.33611111111	99.8%	55.3472222	55.3472222222	1
DataParallel.forward	1.0%	88.98s	20.0%	111.232s	111.232s	0.5%	27.6677777777	25.0%	34.591666	34.591666666	4740
aten::convolution	0.3%	26.69s	6.0%	33.369s	33.369s	0.15%	8.29999999999	7.5%	10.3772222	10.3772222222	383940
aten::batch_norm	0.2%	17.79s	4.0%	22.246s	22.246s	0.1%	5.53333333333	5.0%	6.9183333	6.9183333333	232260
aten::conv2d	0.1%	8.89s	2.0%	11.123s	11.123s	0.05%	2.76666666666	2.5%	3.4588888	3.4588888888	116130
aten::cudnn_batch_norm	0.2%	17.79s	4.0%	22.246s	22.246s	0.1%	5.53333333333	5.0%	6.9183333	6.9183333333	232260
aten::silu	0.1%	8.89s	2.0%	11.12s	11.123s	0.05%	2.76666666666	2.5%	3.4588888	3.4588888888	116130

Summary of Main Results

Profiling Results on B3 using 2 GPUs:

Name	Self CPU %	Self CPU total	CPU total %	CPU total	CPU time avg	Self CUDA %	Self CUDA total	CUDA total %	CUDA total	CUDA time avg	# of Calls
main_worker	2.0%	200.218125s	99.8%	200.230625s	200.230625s	1.0%	62.253125s	99.8%	62.265625s	62.265625s	1
DataParallel.forward	1.0%	100.109375s	20.0%	125.13687499	125.1368749999	0.5%	31.12625s	25.0%	38.915625s	38.915625s	4740
aten::convolution	0.3%	30.0325s	6.0%	37.540625s	37.540625s	0.15%	9.3374999999999	7.5%	11.674374999	11.6743749999	383940
aten::batch_norm	0.2%	20.02125s	4.0%	25.026874999	25.0268749999	0.1%	6.2250000000000	5.0%	7.7831249999	7.7831249999	232260
aten::conv2d	0.1%	10.01125s	2.0%	12.513749999	12.5137499999	0.05%	3.1125000000000	2.5%	3.89125s	3.89125s	116130
aten::_conv_depthwise2d	0.2%	20.02125s	4.0%	25.026874999	25.0268749999	0.1%	6.2250000000000	5.0%	7.7831249999	7.7831249999	232260
aten::silu	0.1%	10.01125s	2.0%	12.513749999	12.5137499999	0.05%	3.1125000000000	2.5%	3.89125s	3.89125s	116130

Summary of Main Results

Profiling Results on B0 using 4 GPUs:

Name	Self CPU %	Self CPU total	CPU total %	CPU total	CPU time avg	Self CUDA %	Self CUDA total	CUDA total %	CUDA total	CUDA time avg	# of Ca
main_worker	2.0%	106.783s	99.8%	106.789666	106.789666666	1.0%	33.2016666666	99.8%	33.2083333333	33.2083333333	1
DataParallel.forward	1.0%	53.391666666	20.0%	66.7396666	66.7396666666	0.5%	16.6006666666	25.0%	20.755s	20.755s	4740
aten::convolution	0.3%	16.017333333	6.0%	20.0216666	20.0216666666	0.15%	4.97999999999	7.5%	6.22633333333	6.22633333333	383940
aten::batch_norm	0.2%	10.677999999	4.0%	13.3476666	13.3476666666	0.1%	3.32000000000	5.0%	4.151s	4.151s	232260
aten::conv2d	0.1%	5.33933333333	2.0%	6.67399999	6.67399999999	0.05%	1.66000000000	2.5%	2.07533333333	2.07533333333	116130
aten::cudnn_batch_norm	0.2%	10.677999999	4.0%	13.3476666	13.3476666666	0.1%	3.32000000000	5.0%	4.151s	4.151s	232260
aten::silu	0.1%	5.33933333333	2.0%	6.67399999	6.67399999999	0.05%	1.66000000000	2.5%	2.07533333333	2.07533333333	116130

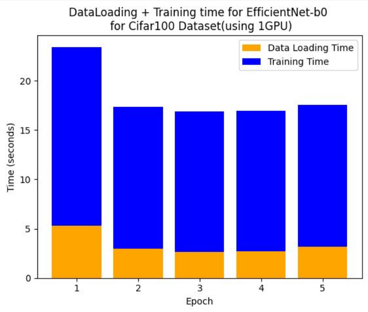
Summary of Main Results

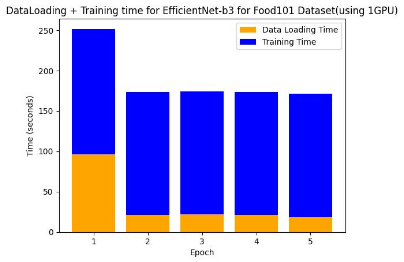
Profiling Results on B3 using 4 GPUs:

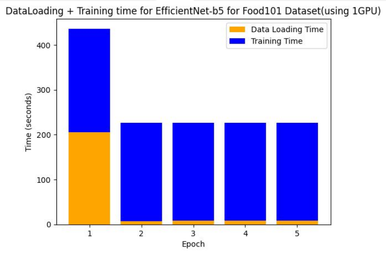
Name	Self CPU %	Self CPU total	CPU total %	CPU total	CPU time avg	Self CUDA %	Self CUDA total	CUDA total %	CUDA total	CUDA time avg	# of Calls
main_worker	2.0%	128.1396s	99.8%	128.1476s	128.1476s	1.0%	39.842s	99.8%	39.85s	39.85s	1
DataParallel.forward	1.0%	64.07000000000	20.0%	80.0876s	80.0876s	0.5%	19.9208s	25.0%	24.906s	24.906s	4740
aten::convolution	0.3%	19.2208s	6.0%	24.026s	24.026s	0.15%	5.976s	7.5%	7.4716s	7.4716s	383940
aten::batch_norm	0.2%	12.8136s	4.0%	16.0172s	16.0172s	0.1%	3.98400000000000	5.0%	4.98119999999	4.98119999999	232260
aten::conv2d	0.1%	6.407200000000	2.0%	8.0087999	8.008799999999	0.05%	1.99200000000000	2.5%	2.4904s	2.4904s	116130
aten::_conv_depthwise2d	0.2%	12.8136s	4.0%	16.0172s	16.0172s	0.1%	3.98400000000000	5.0%	4.98119999999	4.98119999999	232260
aten::silu	0.1%	6.407200000000	2.0%	8.0087999	8.008799999999	0.05%	1.99200000000000	2.5%	2.4904s	2.4904s	116130

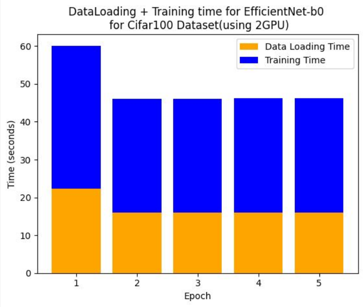
PART 12

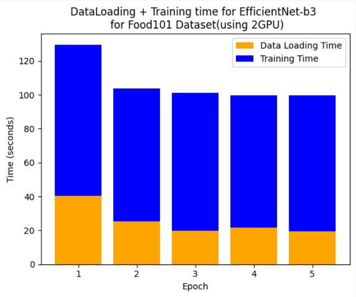
Experimental Evaluation

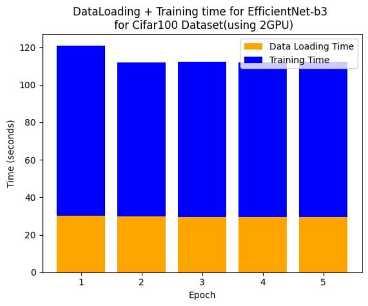


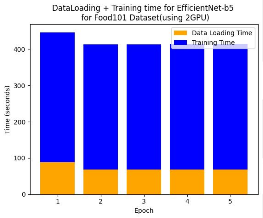


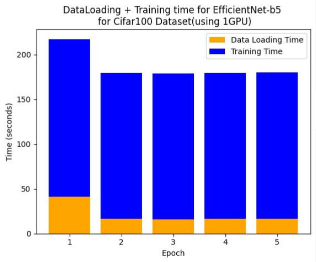






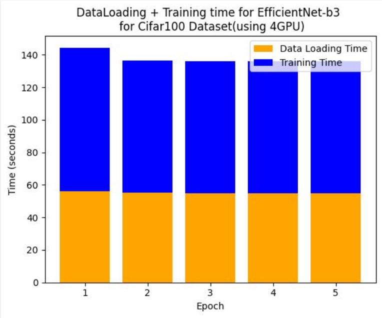


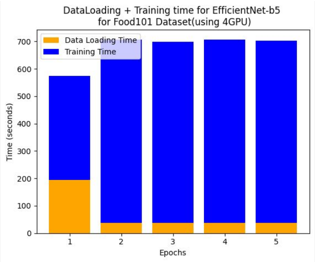


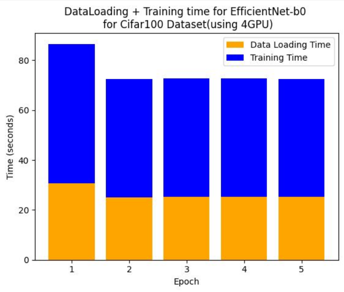


PART 13

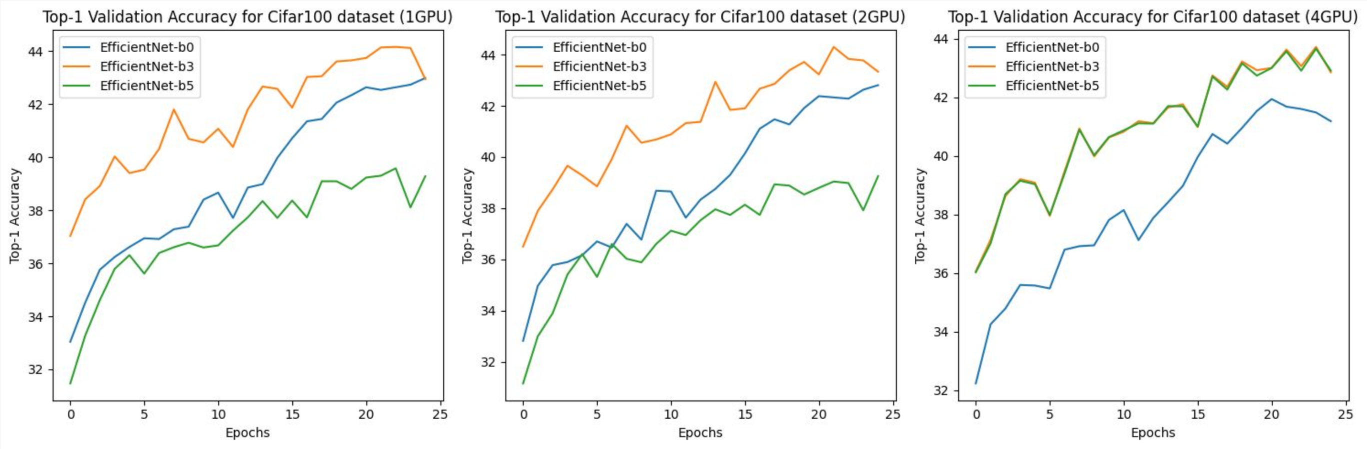
Experimental Evaluation

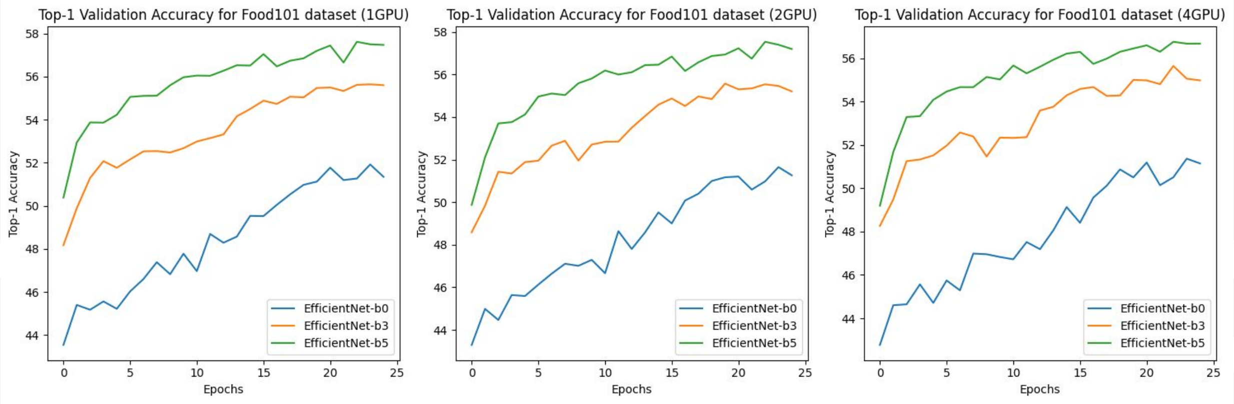






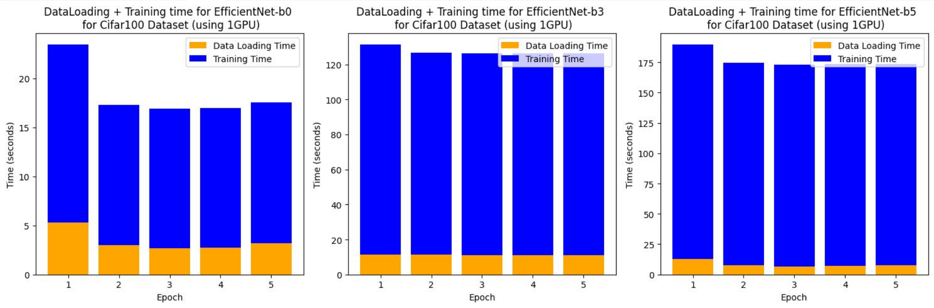
Experimental Evaluation

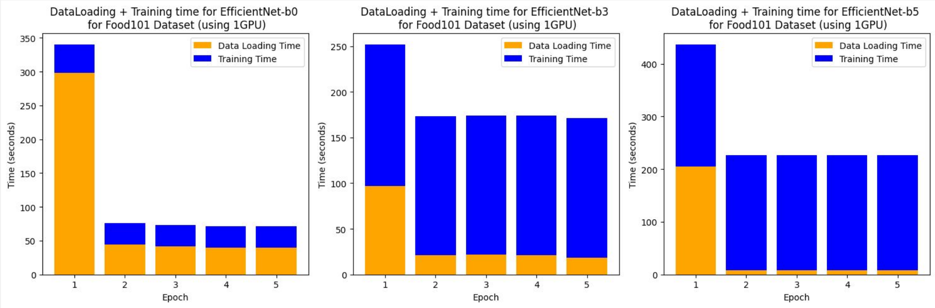


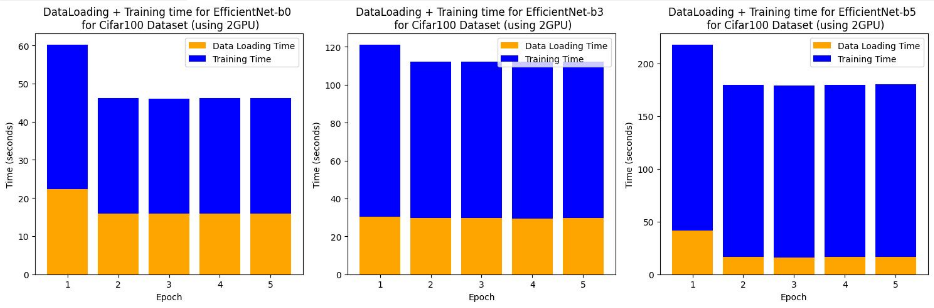


PART 15

Experimental Evaluation







Observations and Conclusion

Profiling Results Observations:

- EfficientNet B0 vs. B3 on 1 GPU:
  - EfficientNet B3 has a higher CPU and CUDA time compared to B0
  - B3 also has more CUDA kernel calls
- Scaling from 1 to 2 GPUs
  - reduction in both CPU and CUDA times for both B0 and B3 models
  - reduction in time is not exactly half
- Scaling from 2 to 4 GPUs
  - Further reduction in times is observed due to diminishing returns as more GPUs are added
  - percentage of self CPU and CUDA time relative to the total decreases

Observations and Conclusion

Bar Chart Observations:

- Data Loading Time:
  - data loading time is significant, especially for EfficientNet B0 on the Food101 dataset
  - the number of GPUs increases, the data loading time does not seem to decrease proportionally
- Training Time
  - Training time per epoch generally decreases with more GPUs
  - The training time for EfficientNet B5 on Food101 is notably higher

Conclusions:

- Model Complexity
- Scaling Efficiency
- Data Loading Constraints
- Hardware Utilization
- Dataset Impact

Github Link

https://github.com/Pratham-mehta/OptimizedEfficientNets