PERFORMANCE EVALUATION OF RESNET-50 INFERENCE ON NVIDIA V100 AND P100 GPUs

EECE5640- High Performance Computing

Prayag Sridhar; Venkatesh Arumugam

4/15/2025

Project Proposal

Objective:

To evaluate the performance of a ResNet-50 model on our P100 and A100 GPUs by measuring inference time, latency, and GPU utilization. We will first conduct a naïve run and then apply optimizations such as mixed precision and increased batch size, comparing the speedup achieved.
Optimization Techniques:

1) Mixed Precision
2) Increased Batch Size

Expected Outcomes:

We expect the V100 GPU to show a significant performance boost over the P100 when applying these optimization techniques. We will analyze the performance differences and explain the results from an architectural standpoint.

Suggested Grading Policy:

- A: Applied both optimizations and clearly reported on the performance variation across both GPUs.
- A-: Applied both optimizations, but the report did not clearly explain the reason for the performance variation.
- B+: Applied one of the optimizations and clearly reported on the performance variation across both GPUs.
- B-: Applied one of the optimizations, but the report did not clearly explain the reason for the variation

Extra Credit Opportunities:

- Achieve data parallelism with both GPUs. (According to the Explorer Quick Start guide, we need to submit a request and obtain approval for multi-GPU usage. Even though our model may not be computationally expensive enough to justify multi-GPU usage, we would like to explore this concept to gain hands-on experience with data parallelism, if time permits.

Introduction

The aim of this report is to evaluate the inference performance of the ResNet-50 model on two different GPUs: NVIDIA Tesla V100 and P100, using varying batch sizes. The V100 is a newer and more expensive GPU compared to the P100. Initially, we benchmark both GPUs without any optimization. Subsequently, we apply various GPU optimization techniques to identify which approaches yield better performance relative to the cost per host.

For this study, we are requesting access to the following GPUs from our computing cluster:

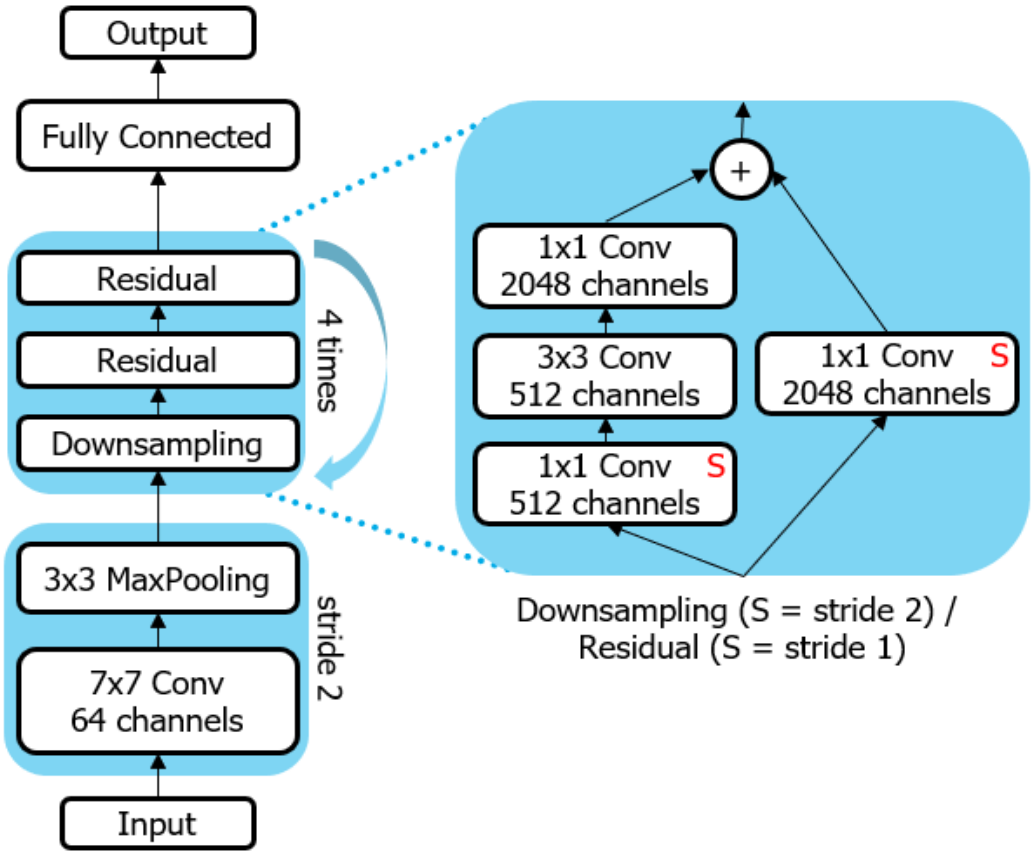
1. Tesla V100-SXM2-32GB
2. P100-PCIE-12GB

According to pricing information from Google Cloud, the V100 (32 GB) costs $4.96 per hour, while the P100 (12 GB) costs $1.46 per hour. In the following sections, we will provide details on the ResNet-50 model used in our experiments, as well as a comparison of the architectural features of the V100 and P100 GPUs.

RESNET-50

In 2015, Kaiming He and his colleagues proposed the ResNet-50 deep convolutional neural network architecture as part of the Residual Network (ResNet) family. During this period, there was a considerable trend toward constructing deeper convolutional neural networks, as greater depth generally allowed for more complex feature representation and thus a boost in performance. However, growth was limited due to training difficulties, particularly with regard to the vanishing gradient problem. While some of the conventional methods, like careful weight initialization and normalization of mini-batches helped to some extent, they were far from effective in solving the problem.

The ResNet concept changed the game, since skip connections, or shortcuts, aided in the more direct passage of gradients back to earlier layers.This rendered incredibly deep networks not only feasible but also trainable within acceptable durations using conventional optimization methods such as stochastic gradient descent (SGD). A significant advancement of ResNet is the incorporation of residual blocks, which mitigates the vanishing gradient issue by enabling the network to learn residual mappings. Consequently, these mappings render the training of deeper networks viable by circumventing the necessity to acquire direct, unreferenced mapping. ResNet-50, with 50 layers, is considerably deeper than preceding CNNs while typically being more resource-efficient than its bigger counterparts, such as ResNet-101 or ResNet-152. It is often organized in the following manner:



The initial Convolution and Pooling: The network starts off with a 7×7 convolutional layer that maps the input image onto a higher-dimensional feature space and then proceeds with a max-pooling layer that reduces the spatial dimensions.
Residual blocks: The fundamental building block of ResNet are bottle-neck residual blocks made up of three convolutions (1×1, 3×3, and 1×1) and all the connections through which the original input to the block gets forwarded to the output. This "shortcut" is very instrumental in reducing the vanishing gradient problem.

- Global Average Pooling: Following the traversal of the stacked residual blocks, the network executes a global average pooling operation that condenses the spatial dimensions to 1×1, resulting in a feature vector.
- Fully Connected Layer: The fully connected layer ultimately transforms the retrieved feature vector into the specified number of output classes.

Due to its architecture, ResNet-50 is applicable to other tasks beyond basic classification, such as object identification, semantic segmentation, and picture captioning, frequently utilizing a transfer learning methodology.

To an extent, ResNet-50 needs to be very resource-consuming on any GPU in a standard training pipeline on account of its depth. Mixed-precision training techniques can, however, speed this up considerably on certain types of GPUs with dedicated hardware such as Tensor Cores. Most

practitioners adopt simple optimization methods such as stochastic gradient descent (SGD)-based techniques with momentum, or the newer methods like Adam and RMSProp for training, but the original research work by He et al. advocated very careful calibrations of SGD schedule. Early stopping, data augmentation, i.e., random cropping and horizontal flipping, and regularization strategies like weight decay and dropout in fully connected layers and label smoothing augment, multiply, improve excellent generalization. Good architectural design and precise procedures of training can quite often make ResNet-50 deliver outstanding results even in complex picture datasets like ImageNet..

CIFAR-10

CIFAR-10 is a small but popular dataset in computer vision and is used widely as a common benchmark for testing the performance of different neural network architectures. The dataset consists of 60,000 color images with a resolution of 32×32 pixels, organized into 10 diverse classes: airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The dataset comprises 50,000 photos for training and 10,000 for testing, enabling academics and practitioners to reliably compare outcomes. CIFAR-10 is often selected for assessing the performance of ResNet-50 due to its balanced and manageable nature, which emphasizes a model's ability to learn both low-level patterns, such as edges and basic textures, and more abstract characteristics, like object shapes and contextual signals.

Furthermore, the diminutive dimensions of each image facilitate expedited data loading and preprocessing, while also rendering training cycles more manageable, so allowing researchers to rapidly iterate on hyper parameter selections, innovative optimization techniques, or alternative network alterations. CIFAR-10 possesses a significant heritage within the computer vision domain, resulting in a comprehensive repository of set baselines for direct comparison. As a result, model innovation and optimization can be assessed by the investigators, with an idea about whether any improvements are considerable or mere noise. Even though the limited dataset may fail to capture most complexities of real-world situations seen from larger datasets like ImageNet, high accuracy on CIFAR-10 does indicate strong underlying pattern identification capability within the model.

It's better to have a very large dataset representing most of the complexities of real-world situations rather than just a couple of instances such as this dataset, which achieves quite high accuracy on CIFAR-10.

NVIDIA Tesla P100

The Tesla P100 is constructed on Pascal architecture, presented substantial innovations at its first launch, featuring High-Bandwidth Memory (HBM2) and a considerable quantity of CUDA cores (up to 3,584). The incorporation of second-generation High Bandwidth Memory into the P100

vastly improved the data transfer rates between memory and computing cores. This enhanced throughput was paramount for applications with considerable data requirements, such as large simulations within meteorological modeling, molecular dynamics, and computational fluid dynamics, and for substantial mini-batches in deep learning.

This combination facilitated accelerated data throughput, rendering the P100 highly appropriate for computationally intensive activities such as deep learning training, large-scale simulation, and data analytics. The P100 lacks specialized Tensor Cores designed for enhancing matrix operations; however, it accommodates half-precision (FP16), single-precision (FP32), and double-precision (FP64) computations, making it adaptable for scientific computing and AI applications. Numerous academic laboratories and data centers persist in utilizing the P100 because to its strong equilibrium of computational capacity, memory bandwidth, and cost efficiency.

NVIDIA Tesla V100

The overwhelming desire to launch the V100 graphics processing unit (GPU) was inside the architecture of the Volta, which would show large numbers of important improvements in the field of deep learning. It features high-bandwidth HBM2 memory that is available in 16 GB or

32 GB configurations, and it has thousands of CUDA cores (5,120 in most models). Its highlighting point is that it features the Tensor Cores, which are specialized units designed to enhance matrix multiplication, hence to be used for neural network operations in mixed-precision training.

The Tensor Core uses multiplications in half-precision and accumulations in single-precision to enhance most workloads of deep learning applications by doubling or tripling their effective throughput over previous designs when used optimally. The increase enables the best training for large networks such as image classifiers, natural language processing, and simulations depending on very large datasets. As a result, the Tesla V100 quickly became recognized as the ideal option in the most advanced research laboratories and corporate data centers designed to speed up AI workloads. Extensive parallelization, high memory bandwidth, and specialized hardware components for deep learning development mark the Forward Evolution of GPU Design to Meet the Requirements for Machine Learning Today and High-End Performance Computing Applications.

TensorFlow, PyTorch Software Frameworks: The V100 switched to use mixed-precision. Researchers will seamlessly take advantage of Tensor Cores by having to change a line or two of code or by selecting specific modes of training. Thus, V100 clusters can enjoy significant benefits, be they from developing advanced AI models or running physics simulations for weather forecasting, genomics, or astrophysics.

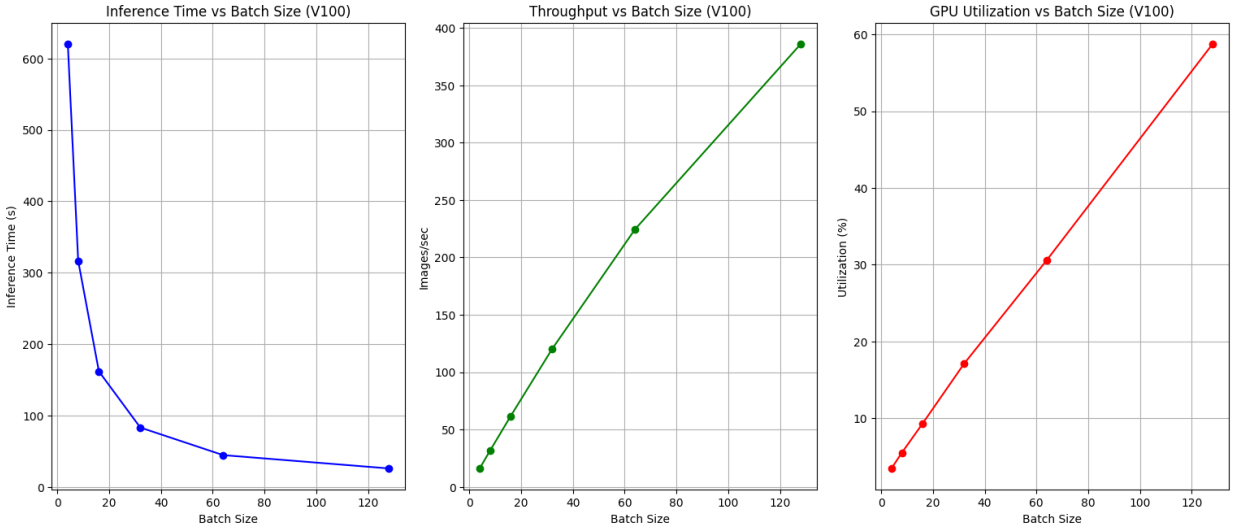
It has also become a common staple for data centers and cloud facilities around the world for its technological advantages. Countless providers offer V100 instances in their platforms for users to deploy and grow powerful GPU-accelerated environments whenever necessary. This immense availability proves much for the card because, while things get broader in deep learning, analytics, and high-performance computing, the V100 stays important in large-scale computational processes within academia and industry. High-perfor-mane computing applications are linked together with the emerging world of AI-ent-touched workloads through Tesla V100. These are found in advanced AI research labs working in areas of image recognition, language comprehension, and recommendation systems. It also can be found in national supercomputing centers conducting detailed simulations used to break through scientific obstacles.


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
FP64
INT
FP32
TENSOR CORE
LD/ ST
SFU
128KB L1 Data Cache / Shared Memory
Tex

Benchmarking V100:

For inference with a V100 GPU, increasing the batch size greatly decreases total inference time and increases throughput resulting in much better GPU utilization. While memory consumption is very stable at smaller batch sizes, memory consumption can ramp up drastically at larger batch sizes and exceed hardware limits. Therefore, for high-throughput applications, larger batch sizes are much more cost-effective. So, using small batch sizes on a powerful GPU is costly because we'll not fully utilize the resources.

Batch Size	Inference Time (s)	Throughput (img/sec)	GPU Utilization (%)	Avg Memory Used (MB)	Total Memory (MB)
4	620.01	16.13	3.48	16787.00	32768.00
8	316.63	31.58	5.53	16787.00	32768.00
16	161.81	61.80	9.32	16787.00	32768.00
32	83.48	119.80	17.25	16787.00	32768.00
32 (run 2)	82.89	120.64	16.95	16787.00	32768.00
64	44.56	224.43	30.61	16787.00	32768.00
128	25.92	385.81	58.73	31481.00	32768.00



Benchmarking on P100 GPU – Constraints and Considerations

Due to the limited memory capacity of the P100 GPU (12 GB), we encountered a RESOURCE_EXHAUSTED error while attempting to benchmark the ResNet-50 model with larger batch sizes under the naive (unoptimized) configuration. This prevented the model from allocating sufficient memory during execution:

RESOURCE_EXHAUSTED: failed to allocate memory

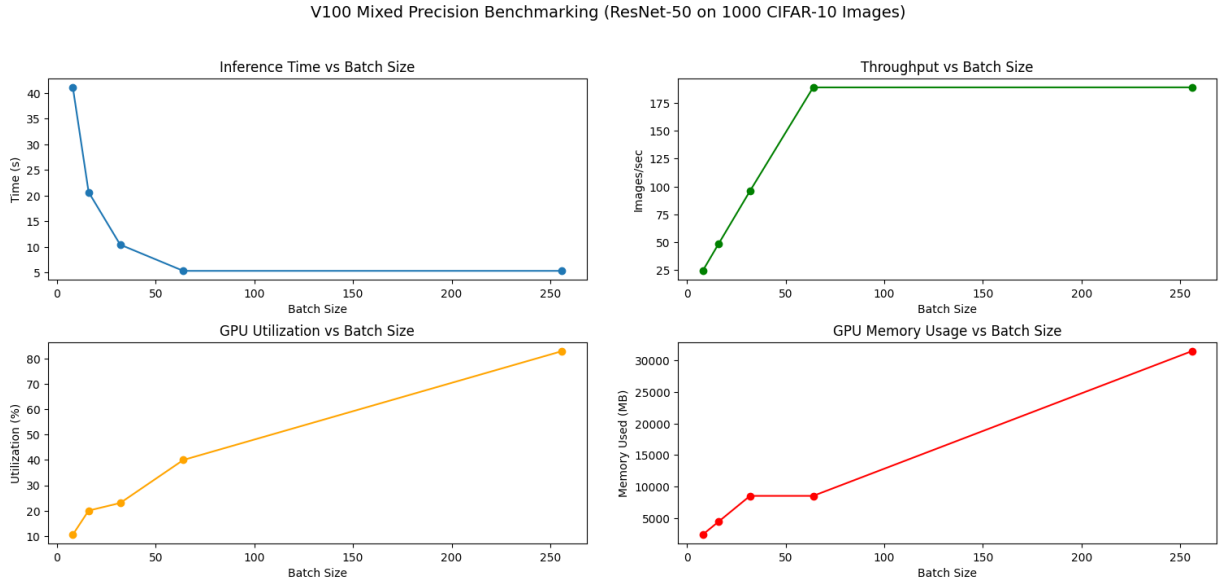
As a result, we restricted our P100 evaluation to only the optimized version of the model, where GPU memory usage was more efficient. This allowed us to successfully run benchmarks across multiple batch sizes without exceeding memory limits. Consequently, P100 results presented in this report reflect optimized inference performance only, and are not directly comparable to the naive runs performed on V100. However, they still provide valuable insights into how optimization techniques impact inference throughput on lower-memory GPUs.

Benchmarking on V100 GPU – Optimizations using mixed precision

Batch Size	Inference Time (s)	Throughput (images/s)	GPU Utilization (%)	Memory Used (MB)
8	41.14	24.31	10.5	2383.0
16	20.67	48.39	20.0	4431.0
32	10.41	96.04	23.0	8527.0
64	5.29	188.90	40.0	8527.0
256	5.29	188.95	83.0	31469.0

We evaluated the performance of ResNet-50 on the V100 GPU using mixed precision for various batch sizes (8 to 256). As expected, we observed a clear improvement in throughput with increasing batch size, peaking at 188.95 images/sec for batch sizes 64 and 256. The GPU utilization rose from 10.5% at batch size 8 to 83.0% at batch size 256, indicating efficient hardware usage at higher loads.

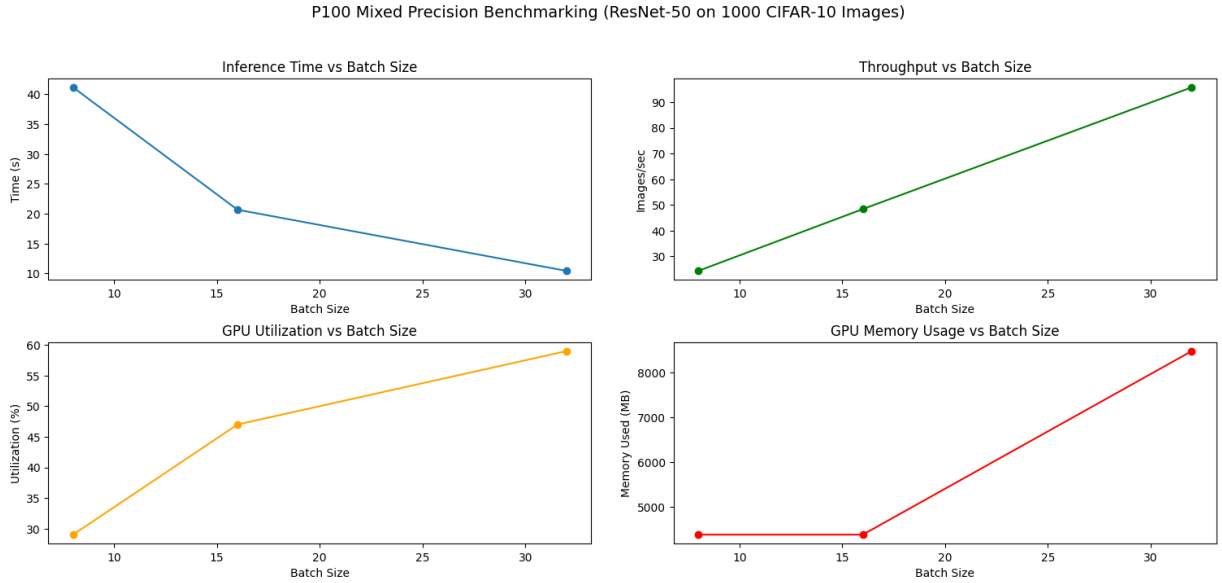
Interestingly, the inference time decreased significantly as batch size increased, demonstrating the benefits of batching and mixed precision. The memory usage also scaled with batch size, with the highest memory usage observed at batch size 256 (31.5 GB), nearly saturating the available GPU memory. Despite this, the V100 was able to maintain high throughput, validating its suitability for high-throughput, mixed precision workloads



Benchmarking on P100 GPU – Optimizations using mixed precision

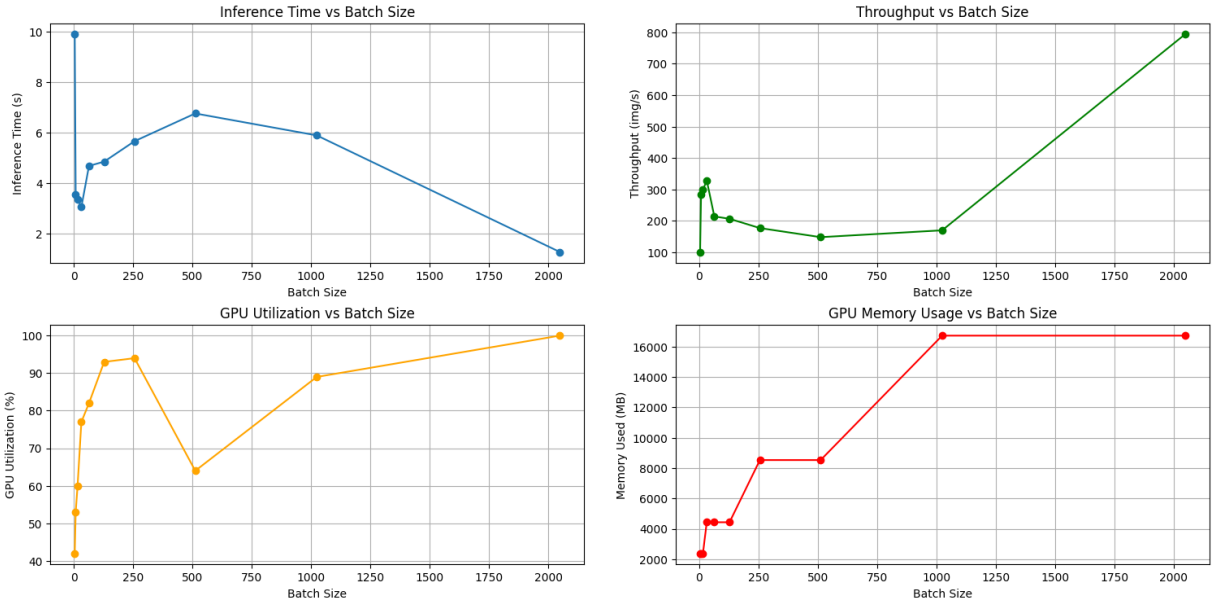
Batch Size	Inference Time (s)	Throughput (images/s)	GPU Utilization (%)	Memory Used (MB)
8	41.17	24.29	29.0	4378.0
16	20.68	48.35	47.0	4378.0
32	10.44	95.76	59.0	8474.0

- Inference time decreases significantly with increasing batch size, showing effective batching.
- Throughput improves almost linearly with batch size, highlighting efficient parallelism with mixed precision on the P100.
- GPU utilization increases with batch size, suggesting better hardware saturation at higher workloads.
- Memory usage grows with batch size, and at batch size 32, we are already using ~8474 MB out of 12288 MB (almost 70%).



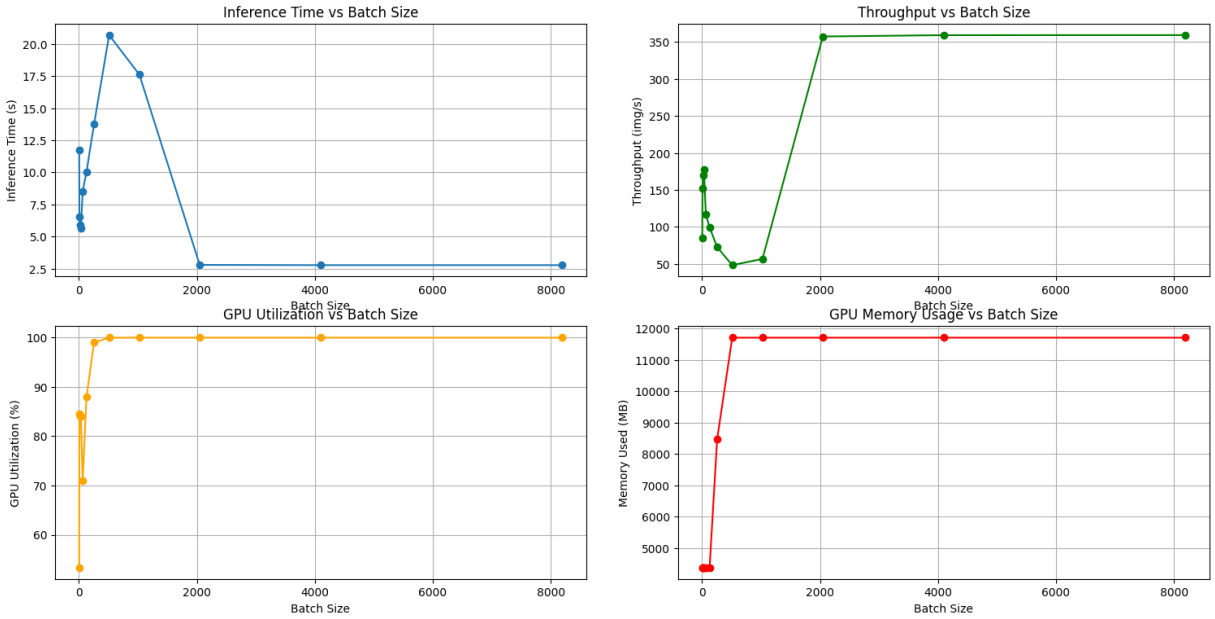
Benchmarking on V100 – Mixed Precision + Tensorflow graph

Batch Size	Inference Time (s)	Throughput (imgs/s)	GPU Utilization (%)	Memory Used (MB)
4	9.89	101.09	42.0	2383.0
8	3.53	283.11	53.0	2383.0
16	3.35	298.33	60.0	2383.0
32	3.04	328.96	77.0	4431.0
64	4.67	214.14	82.0	4431.0
128	4.84	206.65	93.0	4431.0
256	5.65	177.05	94.0	8527.0
512	6.75	148.19	64.0	8527.0
1024	5.88	169.94	89.0	16719.0
2048	1.26	793.95	100.0	16719.0



Benchmarking P100- Mixed Precision +Tensorflow graph

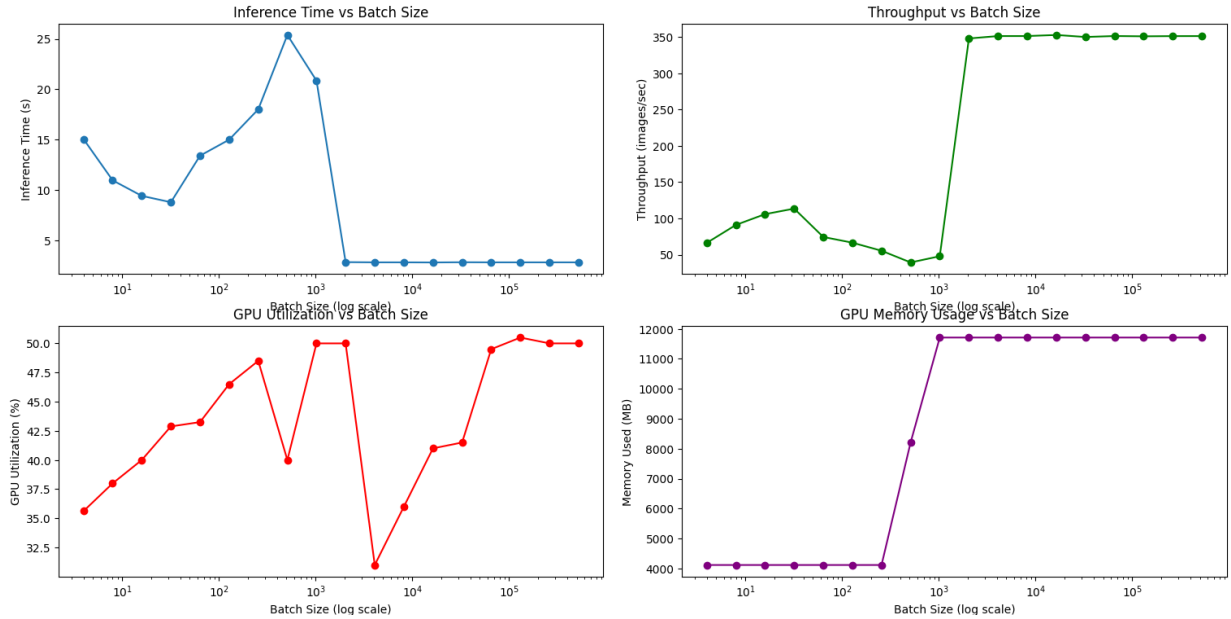
Batch Size	Inference Time (s)	Throughput (img/s)	GPU Utilization (%)	Memory Used (MB)
4	11.77	85.00	53.33	4378.0
8	6.56	152.46	84.5	4378.0
16	5.89	169.67	84.0	4378.0
32	5.63	177.52	84.0	4378.0
64	8.53	117.17	71.0	4378.0
128	10.05	99.49	88.0	4378.0
256	13.75	72.72	99.0	8474.0
512	20.69	48.34	100.0	11712.0
1024	17.62	56.75	100.0	11712.0
2048	2.80	357.63	100.0	11712.0
4096	2.78	359.38	100.0	11712.0
8192	2.78	359.48	100.0	11712.0



Benchmarking with two P100 (Extra Credit):

We tried evaluating the performance with getting two P100 GPU's to see if we can replicate the performance of 1 V100 GPU. For this we got 2-12GB P100 in the same node and repeated the experiment.

Batch Size	Total Inference time(s)	Throughput (images/sec)	GPU Utilization (%)
2	15.07	66.37	35.62
4	10.98	91.09	38.00
8	9.44	105.91	40.00
16	8.81	113.53	42.88
32	13.42	74.5	43.25
64	15.01	66.60	46.50
128	18.02	55.5	48.50
256	25.39	39.39	40.50
512	20.86	47.95	50.00
1024	2.87	347.91	50.00
2048	2.85	351.23	31.00
4096	2.85	352.71	36.00

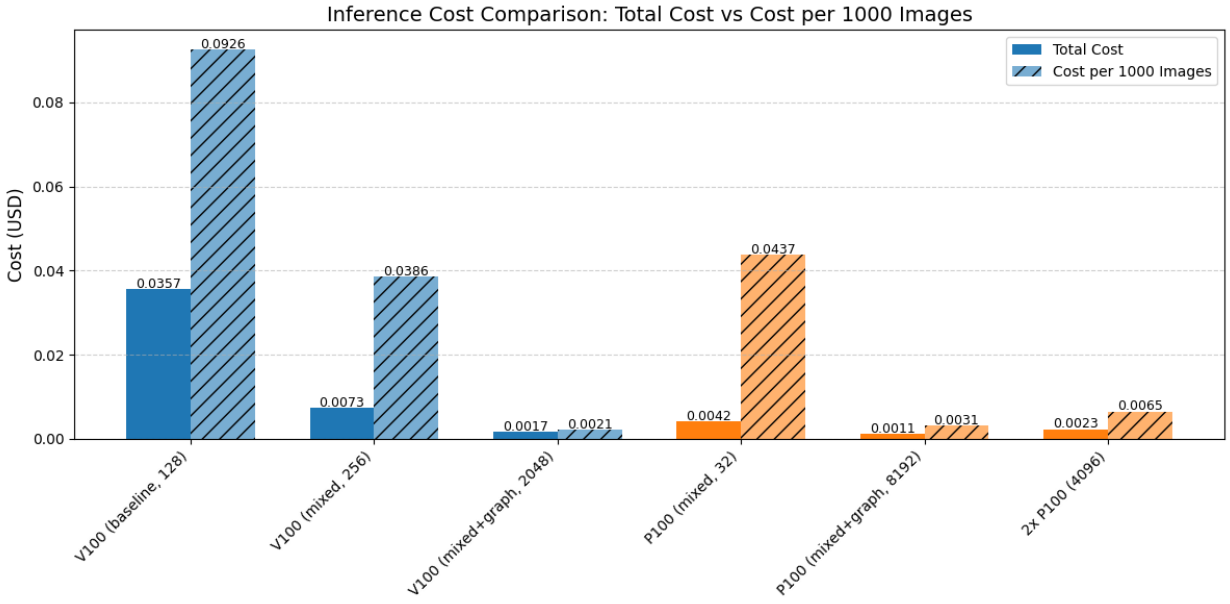


From the observed results, it is evident that after a certain batch size (around 2048), the throughput on a single GPU saturates at approximately 350 images per second, with GPU utilization hovering around 50% and memory usage plateauing at 11.7 GB. This indicates that the GPU's computational and memory capacities are being approached, and further increases in batch size do not lead to proportional improvements in throughput. Introducing a second GPU could be beneficial in such scenarios, particularly for larger batch sizes, as it would allow for data parallelism—splitting the workload between the GPUs—which can potentially double throughput while reducing the memory pressure on each GPU. This setup would also lead to better overall GPU utilization. However, the advantages of using two GPUs become significant primarily for medium to large batch sizes (2048 and above), where the workload is sufficient to offset the communication and synchronization overheads associated with multi-GPU execution. For smaller batch sizes, the added complexity may not yield meaningful performance gains.

Inference Cost Comparison across V100 and P100 GPU Setups

For this experiment we get the maximum utilization condition for each case and plot the cost per hour and cost per 1000 images.

Configuration	Time (S)	Throughput(image/s)	$Cost/hr	Total $Cost for inference	$Cost/1000 Images
V100 (baseline, batch 128)	25.92	385.81	4.96	0.0357	0.0926
V100 (mixed precision, batch 256)	5.29	188.95	4.96	0.0073	0.0386
V100 (mixed + graph, batch 2048)	1.26	793.95	4.96	0.0017	0.0021
P100 (mixed, batch 32)	10.44	95.76	1.46	0.0042	0.0437
P100 (mixed + graph, batch 8192)	2.78	359.48	1.46	0.0011	0.0031
Two P100 GPUs (batch 4096)	2.85	352.71	2.92	0.0023	0.0065



The cost comparisons across the various GPU configurations made for some interesting summaries. Using optimizations such as mixed precision along with TensorFlow graph configurations, we have reasonably improved both the total inference cost and also the cost-per-1000-images - as using the V100 GPU configuration with mixed precision and the graph

produced lowest cost of $0.0021, and least total inference cost of $0.0017. In fact the baseline V100 configuration produced the most amount total cost of $0.0357, and highest ratio of cost-per-1000-images at $0.0926, to exemplify how inefficient this use case is while running inference without any optimizations. The P100 GPU is older, and is therefore slower than the V100, but even including mixed precision, and TensorFlow graph, the P100 was shown to provide a very different configuration - resulting in total inference cost of down to $0.0011, and cost-per-1000-images down to $0.0031. When using two P100 GPUs provided throughput comparable to the optimized V100 configuration, but these had total inference costs $0.0023, and cost-per-1000-images $0.0065. In summary, the fully optimized V100 configuration provided the best balance of performance and cost effectiveness, obviously meaning that the V100 is much better suited to a high inference demand task

Conclusion

The V100 GPU is far superior to the P100, specifically due to its newer Volta architecture which has Tensor Cores that are going to optimize mixed precision performance, and features for faster and more memory efficient utilization. The V100 also has higher memory, and has a higher memory bandwidth, so the V100 can accommodate larger input sizes, and run much larger batch sizes compared to the P100. The P100, on the other hand, is based on the older Pascal architecture, and does not have native support for mixed precision, nor does it have Tensor Core support, which lead to less efficient memory usage as using mixed precision will then fall back to FP32, and may exhaust limited memory quickly when working with deep learning models that are large and were run on smaller resized 224×224 images dense models, like ResNet-50. Different architectures along with differences in hardware, allow the V100 to be greatly more optimized to handle high-performance inference of deep learning models.

References:

https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/

https://en.wikipedia.org/wiki/Volta_(microarchitecture)

https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

https://www.nvidia.com/en-us/data-center/pascal-gpu-architecture/

https://en.wikipedia.org/wiki/Pascal_(microarchitecture)

https://cvw.cac.cornell.edu/gpu-architecture/gpu-example-tesla-v100/volta_sm

https://www.xcelerit.com/computing-benchmarks/insights/benchmarks-deep-learning-nvidia-p100-vs-v100-gpu/

https://www.e2enetworks.com/blog/tesla-v100-vs-p100-do-you-know-the-differences

https://cloud.google.com/compute/gpus-pricing