Infer-HiRes: Accelerating Inference for High-Resolution Images with Quantization and Distributed Deep Learning

Radha Gulhane
The Ohio State University
Columbus, Ohio, USA
gulhane.2@osu.edu

Quentin Anthony
The Ohio State University
Columbus, Ohio, USA
anthony.301@osu.edu

Aamir Shafi
The Ohio State University
Columbus, Ohio, USA
shafi.16@osu.edu

Hari Subramoni
The Ohio State University
Columbus, Ohio, USA
subramon@cse.ohio-state.edu

Dhabaleswar K. Panda
The Ohio State University
Columbus, Ohio, USA
panda@cse.ohio-state.edu

ABSTRACT

High-Resolution Images are being used in various applications, including Medical Imaging, Satellite Imagery, and Surveillance. Due to the evolution of Deep Learning (DL) and its widespread usage, it has also become a prominent choice for high-resolution image applications. But, large image sizes and denser convolutional neural networks pose limitations over computation and memory requirements. To overcome these challenges, several studies have discussed efficient approaches to accelerate training, but the inference of high-resolution images with deep learning and quantization techniques remains unexplored. In this paper, we propose accelerated and memory efficient inference techniques leveraging quantization techniques to reduce the memory and computation requirements while maintaining accuracy. Furthermore, we utilize different parallelism for Distributed DL to enable inference for high-resolution images on out-of-core models. We demonstrate an average 6.5× speedup and 4.55× memory reduction with a single GPU using INT8 quantization. By utilizing Distributed DL, we enabled inference for scaled images, achieving an average 1.58× speedup and 1.57× memory reduction using half-precision. To the best of our knowledge, this paper is the first in the literature to focus on high-resolution image inference using quantization with the support of Distributed DL.

CCS CONCEPTS

• **Computing methodologies** → **Object recognition**; **Neural networks**; **Distributed computing methodologies**.

KEYWORDS

Inference, Quantization, High-Resolution Images, Distributed Deep Learning

PEARC '24, July 21–25, 2024, Providence, RI, USA

ACM ISBN 979-8-4007-0419-2/24/07
https://doi.org/10.1145/3626203.3670548

ACM Reference Format:
Radha Gulhane, Quentin Anthony, Aamir Shafi, Hari Subramoni, and Dhabaleswar K. Panda. 2024. Infer-HiRes: Accelerating Inference for High-Resolution Images with Quantization and Distributed Deep Learning. In *Practice and Experience in Advanced Research Computing (PEARC '24), July 21–25, 2024, Providence, RI, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3626203.3670548

1 INTRODUCTION

The high-resolution images have a vast range of applications, including in sectors such as medical imaging, satellite imagery, and surveillance. Typically, images used in these applications range in gigapixels, with dimensions of 100,000×100,000 pixels and even above. As an example, the digital pathology dataset CAMELYON16 [2] consists of whole-slide images (WSI) with an approximate resolution of 100,000x200,000 pixels at its maximum 40× magnification.

With the evolution of Deep Learning (DL) and its proven efficiency in various sectors, it has also become a prominent choice for High-Resolution image applications to solve problems such as image classification and segmentation. Few popular DL models choices for such applications are ResNet [12], U-Net [26], and AmoebaNet [24], which consist of deep conventional layers. However, considering the large size of the image and several convolution layers, it provides challenges due to memory and computation limitations, as it cannot be accommodated in a single GPU memory.

Several studies [14][19][10] have adopted patch-based approach, where each whole slide image (WSI) is split into small patches with image size such as 256x256. This approach further requires pixel-wise annotation or classification mechanism to classify each patch to well-suited classes. But use of deep convolutions neural networks, restricts the patch size due to memory limitation. For instance, image size of 8192x8192 with ResNet101 model and batch size as one becomes out-of-core model on NVIDIA-A100-40 GB GPU. To facilitate scaled image sizes and improve performance, Hy-Fi [16] and GEMS [15] have made significant contributions by enabling training using Spatial Parallelism for image sizes up to 16384x16384. It further improved performance by integrating different parallelism techniques. While most studies have primarily focused on efficient deep learning training approaches for high-resolution images, optimizing inference in the context of high-resolution images remains unexplored.

In this paper, we propose and evaluate quantization approach to accelerate Deep Learning inference for high-resolution images to reduce memory and computation requirement while maintaining accuracy. Quantization is a technique where model parameters are converted to low-precision such as 16-bit floating point or 8-bit integer from 32-bit floating point. This results in reducing memory utilization and latency and its proven efficiency for DL inference has been evaluated in recent surveys [11][29]. We leverage the benefits of quantization to accelerate high-resolution image inference for deep learning models. Furthermore, to enable scaled image inference, further enhance acceleration, and harness the memory and compute-efficient benefits of different parallelism, we introduce quantization support for Spatial, Layer, and Pipeline parallelism in Distributed DL.

The source code is made available at https://github.com/OSU-Nowlab/Infer-HiRes.

1.1 Motivation

While research in high-resolution images with DL remains essential due to its applicability, several studies have been conducted for efficient training, whereas, very few have delved into the inference for high-resolution images. The studies focusing on inference with high-resolution images primarily involve a single-processing unit and are limited to small-scale images. The exploration of inference with quantization in the context of high-resolution images in Deep Learning and Distributed DL for scaled images is yet to be pursued.

Precision	Memory Utilization (GB)	Memory Reduction	Throughput (Img/Sec)	Speedup
FP32	12.48	Baseline	145.22	Baseline
FP16	7.64	**1.63**×	224.85	**1.55**×
BFLOAT16	5.28	**2.36**×	226.12	**1.56**×
INT8	2.39	**5.23**×	903.35	**6.22**×

Table 1: Memory and Throughput evaluation with different precision quantization using ResNet101 for 256x256 image size and 64 Batch size on NVIDIA A100-40 GB

We evaluated the quantization effects on latency and memory footprint for an image size of 256x256 using ResNet101. Table 1 provides the memory and speedup evaluation by comparing quantization with baseline full-precision (FP32), half-precision (FP16, BFLOAT16), and integer-only precision (INT8). Results show a significant reduction in latency and memory utilization, with the best performance observed for INT8, reducing memory requirements by 5.23× while improving speedup by 6.22×. Further, as ResNet101 can not scale beyond 2048×2048 or 4096×4096 image size on single GPU, to support larger images and slide level inference, we studied different parallelism implemented in Hy-fi and GEMS to enable image-sizes such as 8192×8192 and 16384×16384 and support quantization.

Consider the real-world application of digital pathology images, where inference for one whole slide image (WSI) contains an average of 500 patches, each of size 256x256. The inference time on a CPU [4][17] can take several minutes, while on a GPU, it reduces to seconds. Utilizing GPU-enabled quantization further minimizes this time to just 1-2 seconds.

1.2 Proposed Solution

We propose efficient inference for high-resolution images using DL on a single GPU, as well as in Distributed DL settings, leveraging post training quantization. We exploit the quantization precision range of 16-bit floating point, with both FP16 and BFLOAT16 datatypes, and 8-bit integer. As of today, 8-bit integer is the lowest precision for GPUs supported through PyTorch.

We provide quantization support for single GPU inference, specifically to facilitate patch-based inference, a widely used approach where patch sizes are small-scale images. Furthermore, to enable scaled images, slide-level inference, and improve performance, we enable quantization for Distributed DL. We utilized Spatial, Layer, and Pipeline Parallelism for Distributed DL from the Hy-Fi [16] implementation.

We implement our solution in PyTorch [21] and provide an inference pipeline for high-resolution images, supporting different precision quantization and Distributed DL. We evaluated our work with respect to computation, memory utilization, and accuracy, as discussed in detail in Section 5.

1.3 Contributions

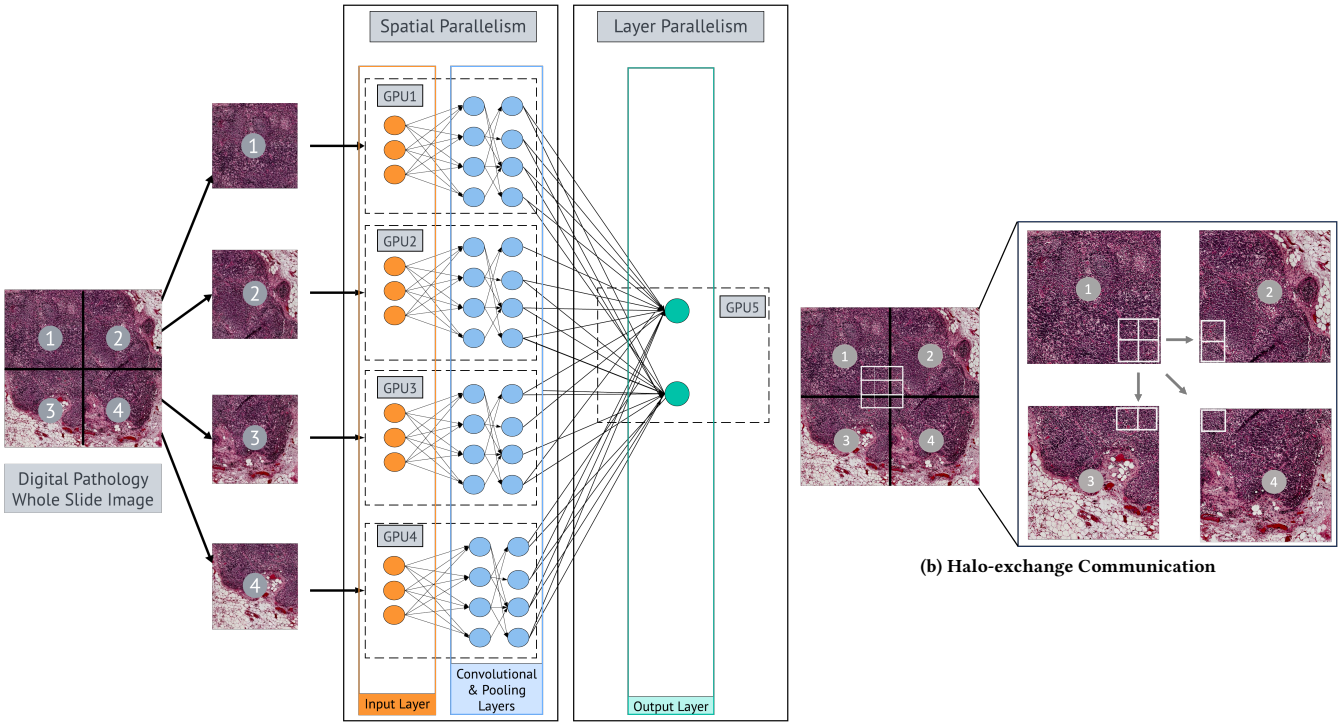
We list our contributions as follows:

(1) Implemented GPU-enabled Post Training quantization for Distributed DL to enable inference for high-resolution images with less computational resources (specifically, the parallelism techniques used are Spatial and Layer/Pipeline Parallelism) (Section 4)
(2) Provided thorough evaluation of quantization for single GPU and multi-GPU using Distributed DL with respective to throughput, memory utilization, and accuracy on different datasets, including CAMELYON16 [2], digital pathology dataset (Section 5)
(3) Achieved the average speedup of 1.58× and memory reduction of 1.57× with FP16 Distributed DL compared with baseline FP32 ResNet101 model (Section 5.3)
(4) With a Single GPU, we achieved speedup by an average of 6.5× while reducing memory utilization by 4.55× with INT8 ResNet101 quantized model when compared with baseline FP32 (Section 5.2)

2 BACKGROUND

2.1 Distributed DL for High Resolution Images

2.1.1 Layer and Pipeline Parallelism. Large Deep Neural Networks are memory and computation-intensive, thus it restricts to the smaller image size and batch size on a single GPU. As we scale with image size, it cannot be trained or inferred with a single GPU as memory requirements to accommodate model parameters exceed the available GPU memory. For such scenarios, Layer Parallelism (LP) [16][15] is employed. LP distributes one or more layers of DL model on different GPUs that can fit into GPU memory. However, distributing layers serialize the computation of GPUs, as input to a



(b) Halo-exchange Communication

(a) Implementation of Spatial and Layer parallelism with spatial partition factor is 4 and model split factor is 2. The first model partition contains convolution and pooling layers, while the second and last model partitions contain the output layer

Figure 1: Overview of Spatial and Layer parallelism

layer present on one GPU will depend on output of previous layer present on different GPU. Consequently, at any given instance, only one GPU will be performing computations, while the rest remain idle. To improve computation and memory efficiency and address the limitation of LP, Pipeline Parallelism (PP) [13] is utilized. In this approach, each layer is distributed similarly to LP, but the input batch size is divided into micro-batches, and each micro-batch size is executed in a pipeline manner.

However, the significantly large memory requirement for model parameters at each layer in Layer and Pipeline Parallelism restricts the batch size to 1 or 2, causing increased latency. Furthermore, if image sizes are scaled further to 4098×4098 and 8192×8192, even a single layer cannot fit into a single GPU memory. Therefore, LP or PP still has limitations when it comes to scaling image sizes and batch size.

2.1.2 Spatial Parallelism. Spatial Parallelism (SP) [16][27] overcomes the limitation of LP and PP by enabling training or inference for larger images and higher batch sizes. In Spatial Parallelism, the whole image is partitioned into smaller non-overlapping spatial parts and distributed across different GPUs. Further, convolution and pooling layers of the DL model are replicated on GPUs containing spatial parts and lastly, output layer will be replicated on single GPU undergoing LP. Figure 1(a) shows the overview of Spatial and Layer Parallelism. The digital pathology image is partitioned into 4 spatial parts, and the model is split into 2 parts. The first model split consists compute and memory-intensive convolution and pooling layers, while the second and final model partitions contain the output layer. Each spatial part performs convolution and pooling operations given by first model split, and finally, the outputs are aggregated by second model split.

Halo-Exchange Communication Convolution and pooling operations require information about adjacent pixels. For pixels located on the boundaries of the spatial segment, their adjacent pixels will be on different GPUs. Figure 1(b) illustrates the halo-exchange required by the first spatial part with different GPUs. Therefore, when using SP, each GPU needs to communicate with different GPUs to obtain adjacent pixels. We refer such communication as halo-exchange.

2.2 Quantization

Quantization is a technique use to reduce number of bits to represent a value, thereby reducing memory usage and latency significantly for a given problem. In context of Deep Learning, quantization is applied to model weights and activations by converting it to low-bit precision such as half-precision (16-bit floating point) or integer precision (8-bit or 4-bit integer) from default 32-bit floating point. Quantization is being widely used in training and inference in deep learning requires additional design efforts to carefully quantize gradients and activations in order to minimize accuracy errors

[30][8]. Recent studies have tended to be more inclined towards inference and have shown successful results [3][11].

2.2.1 Comparison: Integer-Only vs. Floating-Point Quantization. Floating-point conversion, i.e., converting 32-bit floating point to half-precision floating point is relatively simple, as both are floating point data types and follow same scheme representation. On contrast, converting 32-bit floating point to 8-bit integer significantly reduces value range to 256 values and requires to use new scheme representation to map 32-bit float value to integer [29]. This new representation scheme uses the range of floating-point values ([α, β] from Figure 2) in its representation to represent 32-bit floating point to integer values. Figure 2 shows mapping of floating-point range to b-bit integer values range. Further, Equation 3 provides conversion calculation to represent b-bit integer value relative to floating-point, where x_q is b-bit quantized value of floating-point value x.

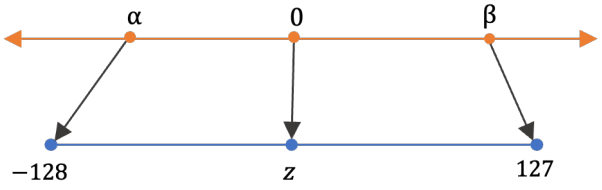


Figure 2: Mapping of floating-point values to 8-bit values[29]

$$s = \frac{2^b - 1}{\beta - \alpha} \tag{1}$$

$$z = -round(\alpha \cdot s - 2^{b-1}) \tag{2}$$

$$x_q = clip(round(x \cdot s + z), -2^{b-1}, 2^{b-1} - 1) \tag{3}$$

Equations 1 and 2 provides quantization parameters required in equation 3 scale factor (s) and zero-point value (z) respectively. Scale factor is floating-point value and zero-point is b-bit integer value corresponding to the zero value in the floating-point representation. $clip()$ maps values outside range to nearest integer representable value. To determine floating-point value range i.e. [α, β] calibration step is used which is done by performing forward pass with few given samples for particular model.

2.2.2 Post-Training Quantization (PTQ). Post-Training Quantization converts the weights and activations of a pre-trained unquantized model to low-bit precision, thereby reducing memory and computation requirements for inference. PTQ is categorized into two different modes, namely dynamic quantization and static quantization. Dynamic Quantization converts weights into low-precision values beforehand but converts activation dynamically at runtime depending observed data range. On other hand, in static quantization, the weights and activation are both quantize into low-precision values and requires calibration step to determine these values. However, for GPUs, PTQ with PyTorch is limited to Static Post-Training Quantization mode via TensorRT, and for our work, we have used Static Post-Training Quantization. Figure 3(b) provides the overview of Post-Training Quantization Inference Pipeline.

3 RELATED WORK

3.1 Deep Learning for High-Resolution Images

Due to the challenges posed by large image size of high-resolution images, several research studies have discussed efficient methodologies for improving accuracy using deep learning for training [18][28][10]. However, training these images considerably increases the training time to several hours. To accelerate training, [15][16] propose a Distributed DL approach that reduces training time from several hours to few minutes. For inference, recent work [19][22][25] has been done to make deep learning models accessible for high-resolution images inference. These works use deep learning models with a single processing unit, restricting to smaller image sizes, while Distributed DL for scaled image sizes remains unexplored for inference.

3.2 Quantization in Deep Learning

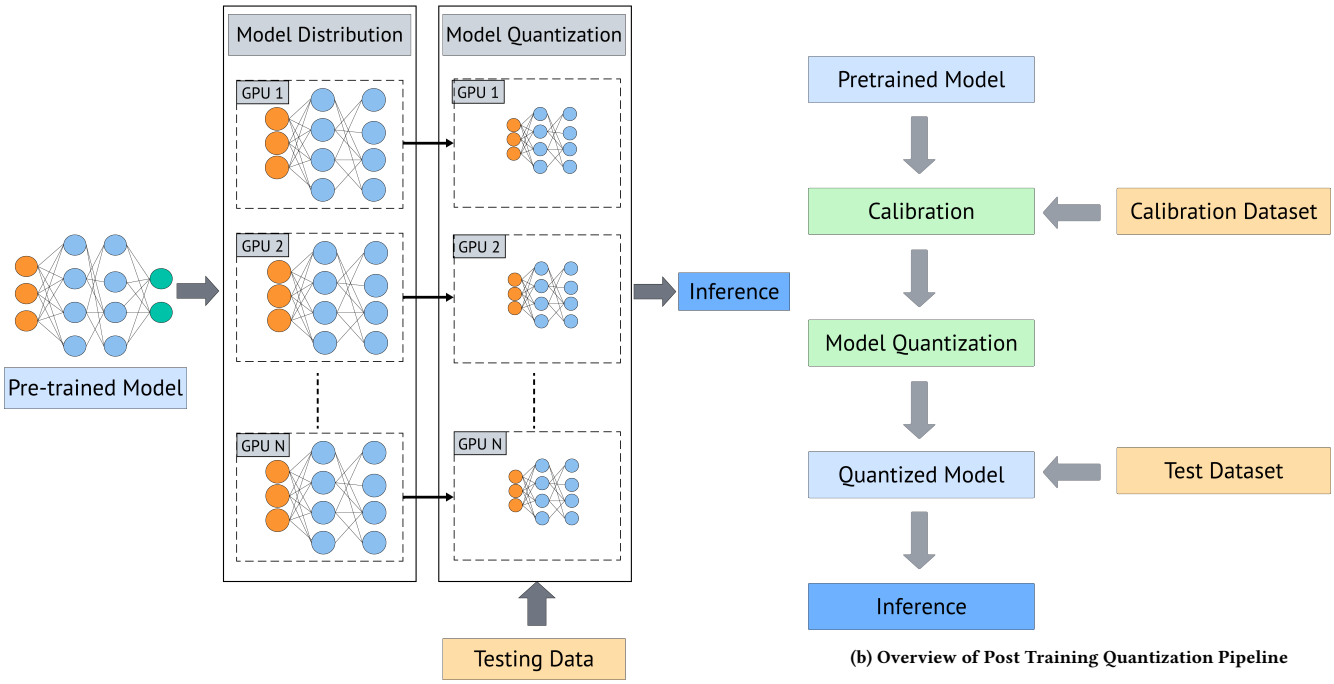
Quantization is utilized in deep learning to minimize memory and computation expenses. It has been widely adopted for inference tasks to achieve less accuracy degradation while lowering memory and computation resource requirements [3][11][20]. For compute and memory intensive large deep models, recent work [23][31] also shows quantization applicability with multi-GPU inference for transformer-based models. However, quantization for for scaled images requiring multi-GPU Distributed DL hasn't been evaluated.

Our proposed work leverages quantization for high-resolution image inference with Deep Learning. We further accelerate and scale image sizes with Distributed DL, utilizing different parallelism techniques for multi-GPU inference.

4 QUANTIZATION IN DISTRIBUTED DEEP LEARNING

Figure 3(b) illustrates a general quantization pipeline, and we follow the same pipeline when dealing with Distributed DL models. However, it's important to note that model quantization is performed separately on each GPU. In Distributed DL, model is distributed among different parallelism strategy and it can be big enough to not fit into memory. Thus, we perform model quantization for each distributed part of model. In terms of Spatial Parallelism, for each spatial part, after initialization of model with given weights, we perform model quantization independently at each GPU device. Similar is the case with Layer parallelism. Figure 3(a) shows implementation pipeline for quantization in Distributed DL.

Spatial parallelism requires to perform halo-exchange, as shown in Figure 1(b), where each convolutional and pooling operation, it perform point-to-point communication as part of the forward pass. In PyTorch, for GPUs, Integer-Only quantization is done via TensorRT. TensorRT takes the Deep Learning (DL) model defined in PyTorch and compiles it to support integer quantization specifically for NVIDIA GPUs. This compilation supports DL layers, such as convolutional, normalization, pooling, etc., but it does not cover collective communication function calls. Since spatial parallelism requires point-to-point communication for halo-exchange in the forward pass, TensorRT cannot resolve such communication calls during compilation, limiting INT8 quantization supported for spatial parallelism.



(a) Inference pipeline for quantization in Distributed DL

(b) Overview of Post Training Quantization Pipeline

Figure 3: Quantization in Deep Learning

5 EVALUATION

We conducted our experiments on NVIDIA A100-40 GB GPUs (2 GPUs per node) with AMD EPYC 7713 64-Core Processor. We used PyTorch v1.13.1 [21] as a Deep Learning framework and TensorRT [7] through Torch-TensorRT API for Integer-Only quantization. For collective communication in Distributed DL, we used NCCL (NVIDIA Collective Communications Library) [6] communication backend.

This section is divided into three parts. First, we understand the different precision quantization effects on accuracy in Section 5.1. Second, we evaluate quantization on a Single-GPU for small-scale images in Section 5.2. Finally, we discuss quantization with Spatial, layer, and Pipeline Parallelism, enabling large-scale images and higher batch sizes in Section 5.3.

5.1 Effect of quantization on accuracy for Inference

For accuracy evaluation, we used ResNet101 and performed model quantization with different precisions. We conducted our accuracy evaluation on various datasets and compared quantization results with the baseline inference accuracy using FP32 precision.

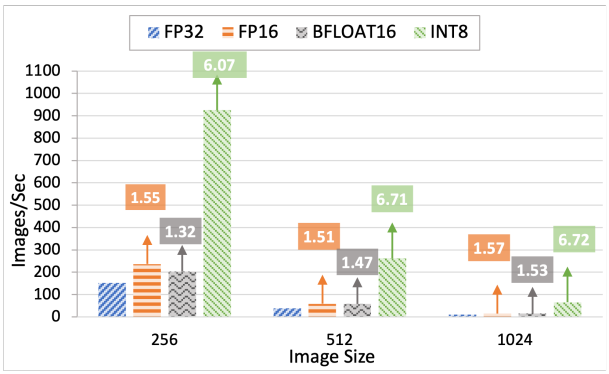
5.1.1 Dataset Description. We used following datasets: CAMELYON16 [2], ImageNet [5], CIFAR-10 [1], and Imagenette [9]. CAMELYON16 is real-world digital pathology dataset from competition held by International Symposium on Biomedical Imaging (ISBI) to detect metastatic breast cancer in whole slide images (WSI). It consists of 400 WSI images categorized into two classes: normal and tumor. ImageNet, CIFAR-10, and Imagenette are object detection datasets containing 1,431,167 images with 1000 object classes, 60,000 images with 10 classes, and 13,394 images with 10 classes, respectively.

5.1.2 Evaluation Methodology. For the CAMELYON16 Dataset, the total size is 300GB, and each image is around 5GB with an approximate image resolution of 100,000x200,000. Since these images cannot fit into memory, for accuracy evaluation, we used a patch-based approach. We extracted patches of size 256x256 containing the tissue region and labeled each patch based on the slide label.

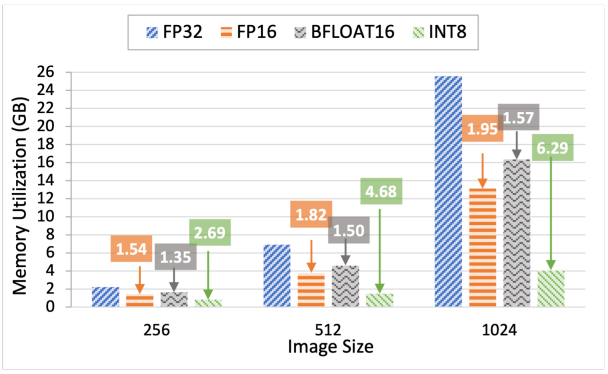
To evaluate the quantization effect on each dataset, we trained ResNet101 for a few epochs to achieve the desired training accuracy, applied PTQ to obtain a quantized model with various precision levels, and then tested the accuracy for inference on either the testing or validation dataset.

Dataset	Precision			
	FP32	FP16	BFLOAT16	INT8
CAMELYON16	70.27	70.26	70.32	70.26
ImageNet	77.62	77.57	78.41	76.85
CIFAR-10	86.02	86.05	86.04	85.99
Imagenette	75.87	75.87	75.90	75.13

Table 2: Inference Accuracy (%) with Different Precision Quantized Models on the NVIDIA A100-40 GB GPU



(a) Throughput Evaluation on Single GPU



(b) Memory Utilization Evaluation on Single GPU

Figure 4: Throughput and Memory Evaluation on a single GPU for the ResNet101 model with different image sizes and batch size 32. The speedup and memory reduction is shown in respective colored boxes for FP16, BFLOAT16, and INT8 when compared to baseline FP32

5.1.3 Result Evaluation. Table 2 shows accuracy evaluation with quantization on different datasets. We observed negligible variations in accuracy while using different precision and demonstrates the accuracy degradation less than **1%.**

5.2 Quantization with Single GPU

We evaluate quantization effects for throughput and memory utilization on different image sizes to understand the benefits of quantization on small-scale image size. Figures 4(a) and 4(b) shows perform evaluation on different image sizes, 256x256 , 512x512, and 1024x1024 with batch size of 32 on ResNet101 model, and we compare our results with baseline FP32 precision.

Figure 4(a) illustrates the throughput evaluation. FP16 improved performance by an average of 1.54×, BFLOAT16 by 1.45×, and INT8 by 6.5×. As shown in Figure 4(b), we achieved an average memory reduction of 1.77×, 1.47×, and 4.55× with FP16, BFLOAT16, and INT8 precision, respectively. Overall, INT8 quantization appears to be the optimal choice for small-scale images with a Single-GPU. Additionally, with an image size of 1024x1024, we observed a 6.29× memory reduction with a speedup improvement of 6.72×. It is important to note the overhead incurred due to calibration with INT8 quantization the very first time we perform model quantization. Figure 3(b) and Section 2.2.1 illustrate the need for the calibration step. To note, for any new model and dataset, it is necessary to perform calibration the first time to obtain an INT8 quantized model. The quantized model's weights can be stored and restored at a later stage. Figure 5 shows the overhead due to calibration. Although we observed a significant calibration overhead of an average of 2.89×, it still outperforms the unquantized FP32 precision model by average speedup of 1.54×.

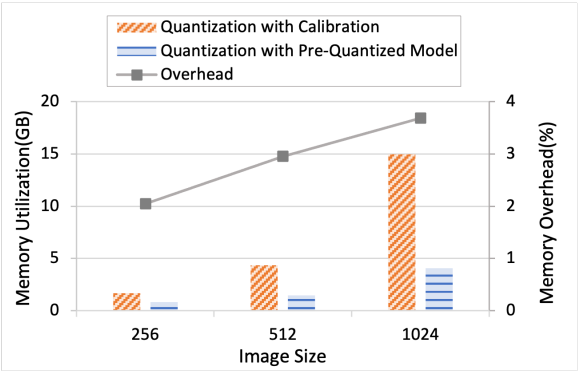


Figure 5: Calibration Overhead

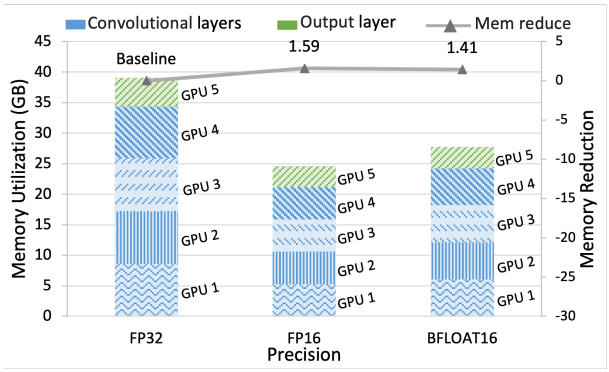
5.3 Distributed DL Quantization Performance Evaluation

In this section, first, we evaluate the performance benefits of using quantization in Distributed DL with respect to memory and throughput. Further, we evaluate benefits of spatial parallelism by enabling inference for very-high resolution images and accelerating performance. We will discuss each of these benefits in specific sections as outlined below.

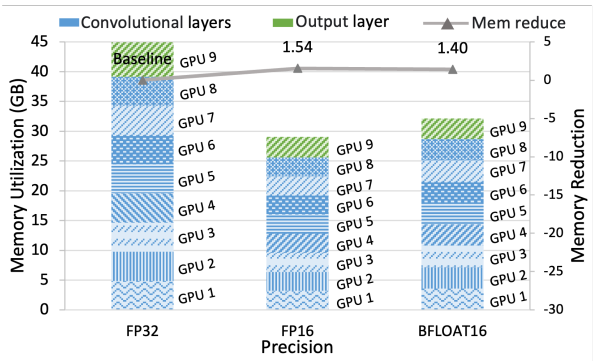
5.3.1 Memory Evaluation. We profiled memory footprints for an image size of 4096x4096 to analyze spatial parallelism with quantization.

Figure 6 illustrates the memory distribution on different GPUs. We perform an experiment configuration with 4 and 8 spatial parts, as shown in Figures 6(a) and 6(b), where one additional GPU is used for layer parallelism. Through quantization, we are able to reduce memory requirements on each GPU by half compared to the memory required with a full-precision FP32 model. Overall, we achieve a memory reduction of 1.57× with FP16 and 1.40× with BFLOAT16 when compared to the baseline FP32.

5.3.2 Throughput Evaluation. We experimented with image sizes image sizes of 2048x2048 and 4096x4096, employing Spatial and layer Parallelism. We scaled the experiment with the number of spatial parts, ranging from 2, 4, to 8, where each part was distributed on a different GPU. It is important to note that an additional GPU was utilized to perform layer Parallelism for the last output layer, as depicted in Figure 1(a). For this experiment, we chose the maximum

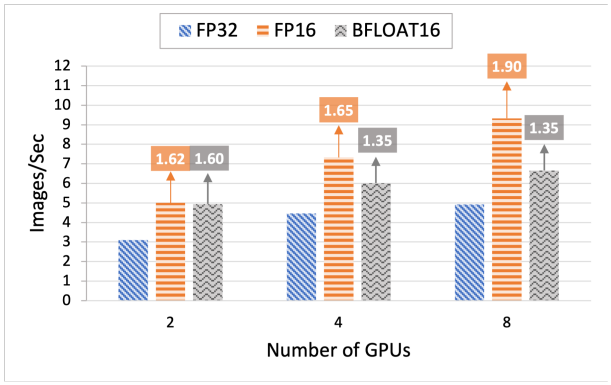


(a) Memory footprints on 5 GPUs for SP+LP

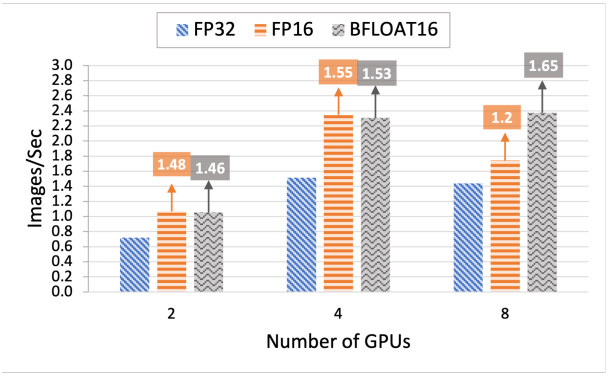


(b) Memory footprints on 9 GPUs for SP+LP

Figure 6: Overview of ResNet101 model and image size 4096×4096 distribution with respective memory and evaluation with SP+LP on multi-GPUs for the ResNet101 model. The evaluation is done by comparing memory utilization by FP16, BFLOAT16 quantization with FP32 as the baseline.

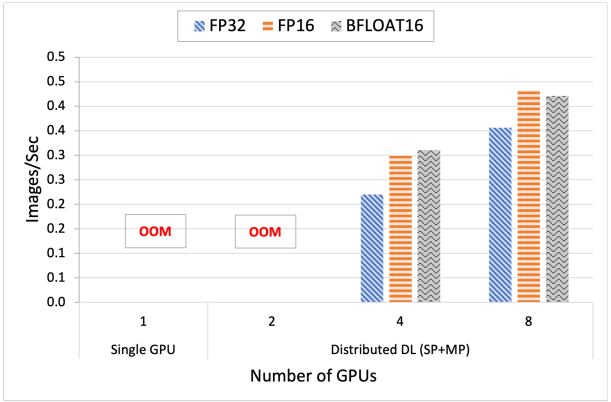


(a) Throughput Evaluation for 2048x2048 Image Size

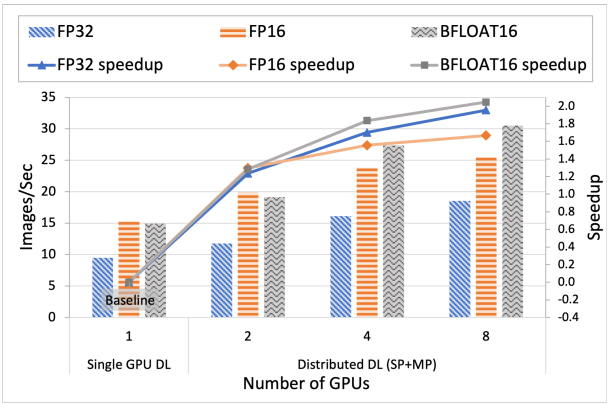


(b) Throughput Evaluation for 4096x4096 Image Size

Figure 7: Throughput Evaluation using SP+LP for ResNet101 model with image sizes of 2048x2048 and 4096x4096. The speedup is shown in respective colored boxes for FP16, BFLOAT16, and INT8 when compared to baseline FP32.



(a) Enabling inference for 8192×8192 with FP16



(b) Accelerating performance with SP

Figure 8: Enabling scaled images and accelerating performance using SP

batch size supported for different GPU counts to utilize memory to its maximum extent. Due to partitioning into a higher number of parts across different GPUs, we were able to enable a higher batch size. For example, with a resolution of 2048x2048, we could not scale beyond a batch size of 16 as it would become out-of-core on a single GPU, but the batch size can be increased when we partition images using SP.

Figure 7(a) shows throughput for 2048x2048 with batch sizes of 16, 32, and 64, on spatial parts 2, 4, and 8, respectively. Similarly, Figure 7(b) shows throughput for 4096x4096 with batch sizes of 16, 32, and 64, on spatial parts 4, 8, and 16, respectively. We compared the results with the baseline FP32. For 2048x2048, we achieved up to a 1.9x speedup with FP16 and 1.6× with BFLOAT16. For 4096x4096, we achieved up to a 1.55× speedup with FP16 and 1.65× with BFLOAT16.

5.3.3 Enabling very high-resolution images. The ResNet101 model, initialized with FP32 precision, requires approximately 87GB of memory for image size 8192x8192. Consequently, it becomes out-of-core even with the smallest batch size of 1 when running on a single GPU with 40GB of memory. It requires to split image into number of spatial parts to enable inference for 8192x8192. We enabled inference for an image size of 8192x8192 with 4 GPUs for spatial partitioning. Figure 8(a) shows the overview and performance for image size 8192x8192 with 4 and 8 GPUs. We further evaluate Spatial Parallelism to accelerate performance while scaling with respect to the number of GPUs. Figure 8(b) shows the performance comparison of SP on different GPUs (2, 4, and 8) with a baseline of a Single GPU. We achieve linear scaling, attaining up to a 1.8× and 2× speedup on 4 and 8 GPUs with BFLOAT16.

6 CONCLUSION

High-resolution images with Deep Learning come with their own set of challenges due to the large size of the image and deep networked DL models, making it compute and memory-intensive. However, research in high-resolution images in DL is crucial due to its applicability and efficiency, for instance, in digital pathology. Our efforts are focused on making trained DL models accessible for high-resolution image inference by reducing computation time and resource requirements.

We proposed accelerated inference for high-resolution images utilizing quantization technique while reducing memory and computation and without accuracy degradation. We provided support for single GPU as well as multi-GPU Distributed DL inference. We achieved overall 6.5× speedup and 4.55× memory reduction with single GPU with INT8 quantization. With Distributed DL, we enabled inference for scaled images. We achieved 1.58× speedup and 1.57× memory reduction using half-precision Distributed DL. We further accelerate performance by 2× using SP compared to single GPU.

We hope that our work will facilitate researchers in achieving accessibility and efficiency in Deep Learning inference while reducing computational costs for their innovative research in the field of high-resolution images.

ACKNOWLEDGMENTS

We thank Arpan Jain et al. for providing access to their Hy-fi work [16]. This research is supported by NSF grants #1818253, #1854828, #2007991, #2018627, #2311830, #2312927, and XRAC grant #NCR-130002, which were instrumental for this research.

REFERENCES

[1] 2014. The CIFAR-10 Dataset. https://www.cs.toronto.edu/~kriz/cifar.html. Accessed: 2024-01-31.
[2] 2016. Camelyon 2016. https://camelyon16.grand-challenge.org/. Accessed: 2024-01-31.
[3] Hyunho Ahn, Tian Chen, Nawras Alnaasan, Aamir Shafi, Mustafa Abduljabbar, Hari Subramoni, Dhabaleswar K., and Panda. 2023. Performance Characterization of using Quantization for DNN Inference on Edge Devices: Extended Version. arXiv:2303.05016 [cs.PF]
[4] Jon Braatz, Pranav Rajpurkar, Stephanie Zhang, Andrew Y. Ng, and Jeanne Shen. 2022. Deep Learning-Based Sparse Whole-Slide Image Analysis for the Diagnosis of Gastric Intestinal Metaplasia. arXiv:2201.01449 [eess.IV]
[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
[6] NVIDIA Developer. 2016. Nvidia Collective Communications Library (NCCL). https://developer.nvidia.com/nccl. Accessed: 2024-01-31.
[7] NVIDIA Developer. 2019. NVIDIA TensorRT. https://developer.nvidia.com/tensorrt/. Accessed: 2024-01-31.
[8] Yinpeng Dong, Renkun Ni, Jianguo Li, Yurong Chen, Jun Zhu, and Hang Su. 2017. Learning Accurate Low-Bit Deep Neural Networks with Stochastic Quantization. arXiv:1708.01001 [cs.CV]
[9] Fastai. [n. d.]. GitHub - fastai/imagenette: A smaller subset of 10 easily classified classes from Imagenet, and a little more French. https://github.com/fastai/imagenette
[10] Ruiwei Feng, Xuechen Liu, Jintai Chen, Danny Z. Chen, Honghao Gao, and Jian Wu. 2021. A Deep Learning Approach for Colonoscopy Pathology WSI Analysis: Accurate Segmentation and Classification. *IEEE Journal of Biomedical and Health Informatics* 25, 10 (2021), 3700–3708. https://doi.org/10.1109/JBHI.2020.3040269
[11] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. A Survey of Quantization Methods for Efficient Neural Network Inference. arXiv:2103.13630 [cs.CV]
[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs.CV]
[13] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. 2019. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. arXiv:1811.06965 [cs.CV]
[14] Osamu Iizuka, Fahdi Kanavati, Kei Kato, Michael Rambeau, Koji Arihiro, and Masayuki Tsuneki. 2020. Deep learning models for histopathological classification of gastric and colonic epithelial tumours. *Scientific reports* 10, 1 (2020), 1504.
[15] Arpan Jain, Ammar Ahmad Awan, Asmaa M. Aljuhani, Jahanzeb Maqbool Hashmi, Quentin G. Anthony, Hari Subramoni, Dhableswar K. Panda, Raghu Machiraju, and Anil Parwani. 2020. GEMS: GPU-Enabled Memory-Aware Model-Parallelism System for Distributed DNN Training. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–15. https://doi.org/10.1109/SC41405.2020.00049
[16] Arpan Jain, Aamir Shafi, Quentin Anthony, Pouya Kousha, Hari Subramoni, and Dhableswar K. Panda. 2022. Hy-Fi: Hybrid Five-Dimensional Parallel DNN Training on High-Performance GPU Clusters. In *High Performance Computing*, Ana-Lucia Varbanescu, Abhinav Bhatele, Piotr Luszczek, and Baboulin Marc (Eds.). Springer International Publishing, Cham, 109–130.
[17] Jakub R. Kaczmarzyk, Alan O'Callaghan, Fiona Inglis, Swarad Gat, Tahsin Kurc, Rajarsi Gupta, Erich Bremer, Peter Bankhead, and Joel H. Saltz. 2024. Open and reusable deep learning for pathology with WSInfer and QuPath. *npj Precision Oncology* 8, 1 (Jan. 2024). https://doi.org/10.1038/s41698-024-00499-9
[18] Mahendra Khened, Avinash Kori, Haran Rajkumar, Balaji Srinivasan, and Ganapathy Krishnamurthi. 2020. A Generalized Deep Learning Framework for Whole-Slide Image Segmentation and Analysis. arXiv:2001.00258 [eess.IV]
[19] Weizhe Li, Mike Mikailov, and Weijie Chen. 2023. Scaling the Inference of Digital Pathology Deep Learning Models Using CPU-Based High-Performance Computing. *IEEE Transactions on Artificial Intelligence* 4, 6 (2023), 1691–1704. https://doi.org/10.1109/TAI.2023.3246032
[20] Zhikai Li and Qingyi Gu. 2023. I-ViT: Integer-only Quantization for Efficient Vision Transformer Inference. arXiv:2207.01405 [cs.CV]
[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith

Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[22] André Pedersen, Marit Valla, Anna M. Bofin, Javier Pérez de Frutos, Ingerid Reinertsen, and Erik Smistad. 2020. FastPathology: An open-source platform for deep learning-based research and decision support in digital pathology. arXiv:2011.06033 [cs.LG]

[23] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems* 5 (2023).

[24] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized Evolution for Image Classifier Architecture Search. arXiv:1802.01548 [cs.NE]

[25] Ruichen Rong, Hudanyun Sheng, Kevin W Jin, Fangjiang Wu, Danni Luo, Zhuoyu Wen, Chen Tang, Donghan M Yang, Liwei Jia, Mohamed Amgad, et al. 2023. A Deep Learning Approach for Histology-Based Nucleus Segmentation and Tumor Microenvironment Characterization. *Modern Pathology* 36, 8 (2023), 100196.

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597 [cs.CV]

[27] Aristeidis Tsaris, Josh Romero, Thorsten Kurth, Jacob Hinkle, Hong-Jun Yoon, Feiyi Wang, Sajal Dash, and Georgia Tourassi. 2023. Scaling Resolution of Gigapixel Whole Slide Images Using Spatial Decomposition on Convolutional Neural Networks. In *Proceedings of the Platform for Advanced Scientific Computing Conference* (Davos, Switzerland) *(PASC '23)*. Association for Computing Machinery, New York, NY, USA, Article 2, 11 pages. https://doi.org/10.1145/3592979.3593401

[28] Jason Wei, Laura Tafe, Yevgeniy Linnik, Louis Vaickus, Naofumi Tomita, and Saeed Hassanpour. 2019. Pathologist-level classification of histologic patterns on resected lung adenocarcinoma slides with deep neural networks. *Scientific Reports* 9 (03 2019). https://doi.org/10.1038/s41598-019-40041-7

[29] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. 2020. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation. arXiv:2004.09602 [cs.LG]

[30] Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. 2018. Training and Inference with Integers in Deep Neural Networks. arXiv:1802.04680 [cs.LG]

[31] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. arXiv:2211.10438 [cs.CL]