CSE 5194.01 Autumn 2020
Lab 1 Report

TF_Horovod: Chen-Chun Chen, Wei Liu, Curtis Isaacson, Lang Xu

1. Introduction

In this lab, we have run several DNN experiments on the OSC cluster Owens with Tensorflow and Horovod framework. First, we trained a neural network model on Fashion MNIST and CIFAR-100 dataset using a single node (CPU/GPU). Then we selected the IMDB-Wiki dataset and trained the Inception model and ResNet model on the dataset. With the larger dataset, we trained the models on a single node (CPU/GPU) with different parameters. We'll analyze our experiment results (accuracy, loss, and training time) in the following sections.

2. What are the performance trends for Fashion MNIST and CIFAR100 for CPU and GPU?

We conducted experiments of different datasets in both CPU and GPU environments. For the CPU environment, we focused on scalability and hybrid performance. For the GPU environment, we observed how hyperparameters, such as batch size, influenced the throughput.

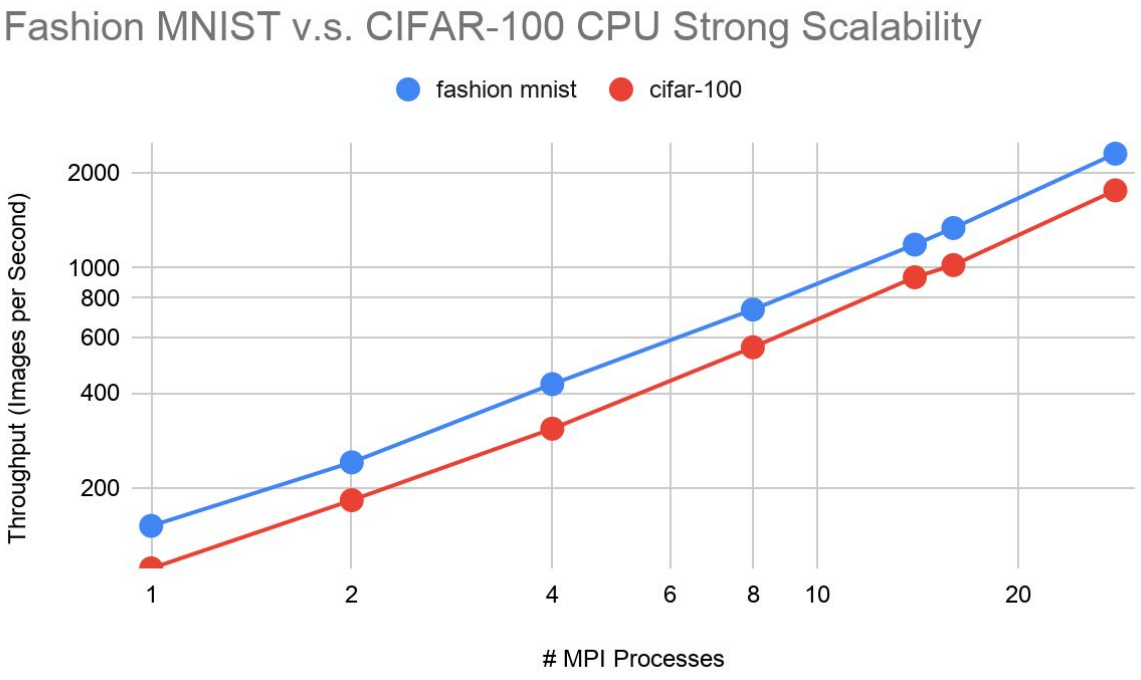

Figure 1

In the CPU experiments, we fixed the batch size to 128, and for both datasets, the models were the same except for the input and output layer due to different dimensions.

Figure 1 shows the performance of the strong scalability. As we increase the number of processes, the training time decreases, which means the throughput increases, for both datasets. It is of note that the throughput of Fashion MNIST is higher than CIFAR-100. This could be attributed to the nature of the CIFAR-100 data because it is 32x32 3-channel images, which makes each data point about 3 times the size of a Fashion MNIST data point.

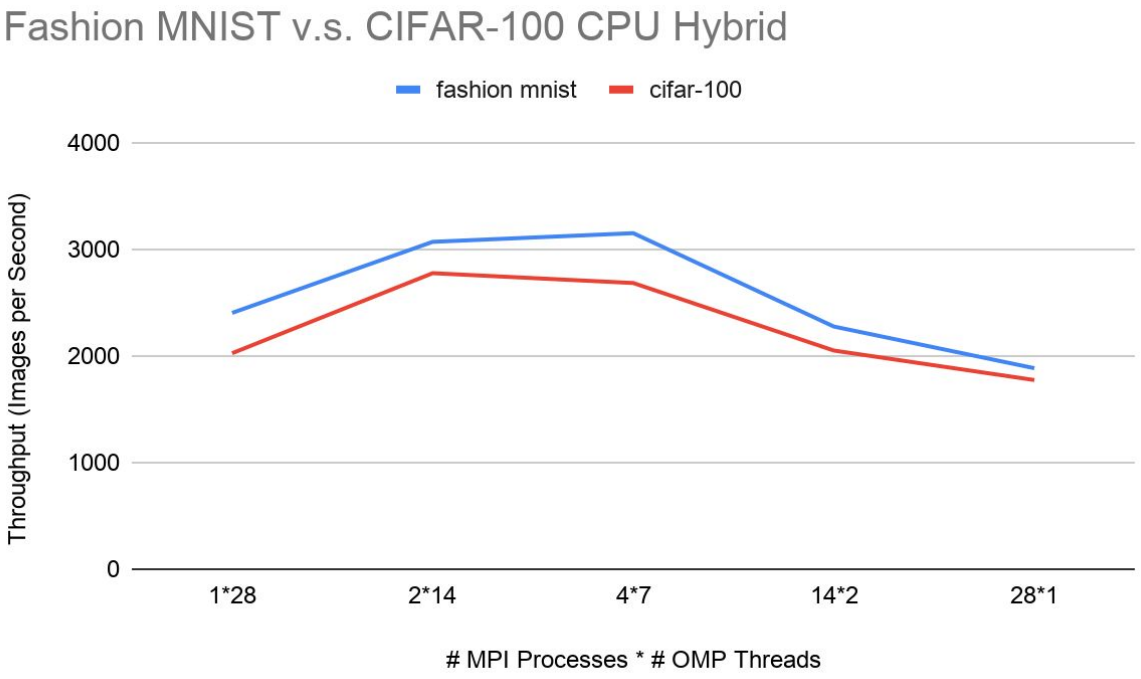

Figure 2

In another CPU experiment, we focused on hybrid performance. We fixed the number of resources and tested different combinations of MPI processes and OMP threads. Figure 2 shows that the best configuration for Fashion MNIST is 2 MPI processes and each process has 14 OMP threads; and for CIFAR-100 is 4 MPI processes and each process has 7 OMP threads. These configurations could be tricky because too many MPI processes may introduce too much communication overhead, which slows down the overall performance. On the other hand, if we just use 1 MPI process, other factors, such as inter-socket overhead, may also reduce the throughput.

The best throughput for Fashion MNIST in CPU is about 3150 images per second, and for CIFAR-100 in CPU is about 2770 images per second.

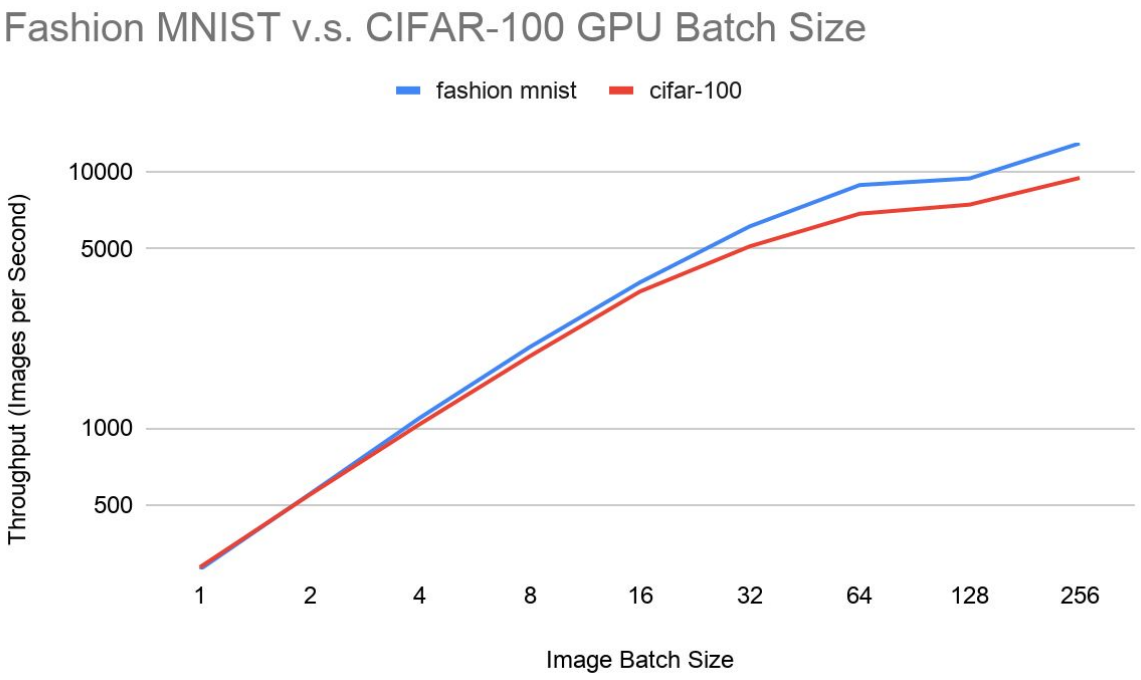

Figure 3

For the GPU environment, because we only had 1 GPU and 1 node, we fixed the number of resources but modified the batch size. Figure 3 shows that the throughput increases as the batch size increases. It is to say, if the accuracy won't drop after using a larger batch size, we intend to train the model with a higher batch size to accelerate the training. This trend is slowing down after batch size 64.

The best throughput for Fashion MNIST and CIFAR-100 in the GPU environment is about 9440 and 7450 images per second, if we set the batch size to 128.

3. What DNN model works best on CPU vs. GPU and why?

When comparing the throughput of the two environments overall, we set the criteria based on the throughput (images per second) that the system was able to process through for different configurations. It is of note that the GPU images per second were much higher at each experiment's respective batch size/number of processes.

With the growing configuration of batch size up to 256, training across GPU reached a throughput of around 12950 and 9480 images per second for Fashion MNIST and CIFAR-100.

However, compared to GPU experiments, when the batch size is fixed to 128, CPU throughput can hardly catch up with GPU throughput. With max resources set up to 28 cores, CPU training could only maximize throughput up to around 3150 and 2770 images per second for Fashion MNIST and CIFAR-100.

One statistic that stands out: The GPU training on a batch size of 8, which is the 4th smallest tested, processed through the same amount of images per second as CPU training with 28 cores, which was the highest number of resources.

For the experiment, we were using a CNN model that is introduced with a Horovod example. The results suggested that the throughput from GPU is always better than CPU throughput in this model. That's because DNN training consists of lots of computation, such as matrix multiplication, and GPU can process these operations parallelly and faster. And the throughput of Fashion MNIST is slightly better than CIFAR-100, and it's because CIFAR-100 is more complex, which has higher input and output dimensions.

It is important to note that, for standard machine learning models where the number of parameters is not as high as deep learning models, CPUs should still be considered as more effective and cost-efficient. Also, there exist methods to optimize CPU performance such as MKLDNN and NNPACK. However, similar methods also exist for GPUs such as TensorRT. We also found that the performance on GPU clusters was far more consistent than the CPU.

4. Mention the scale and size of your dataset and explain why you chose it?

We used the IMDB-Wiki (face only) as the dataset. The size of IMDB is about 7GB and it contains about 460723 images; the size of the Wiki dataset is about 1GB and it contains about 62328 images. We preprocessed these 2 datasets and labeled them by gender.

Considering the limitation of disk quota and file number in Owens, we choose IMDB-Wiki (face only) as our dataset. The size and the number of images are big enough for large-scale training and are able to fit the quota.

Moreover, considering the computing power for only 1 node in this lab, we choose gender, which is a 2 categories classification problem, to make the training result more obvious in the earlier training stage.

5. What is the best DNN architecture for your chosen dataset in terms of "accuracy" and in terms of "training time"? Describe if you find a suitable tradeoff between these two metrics.

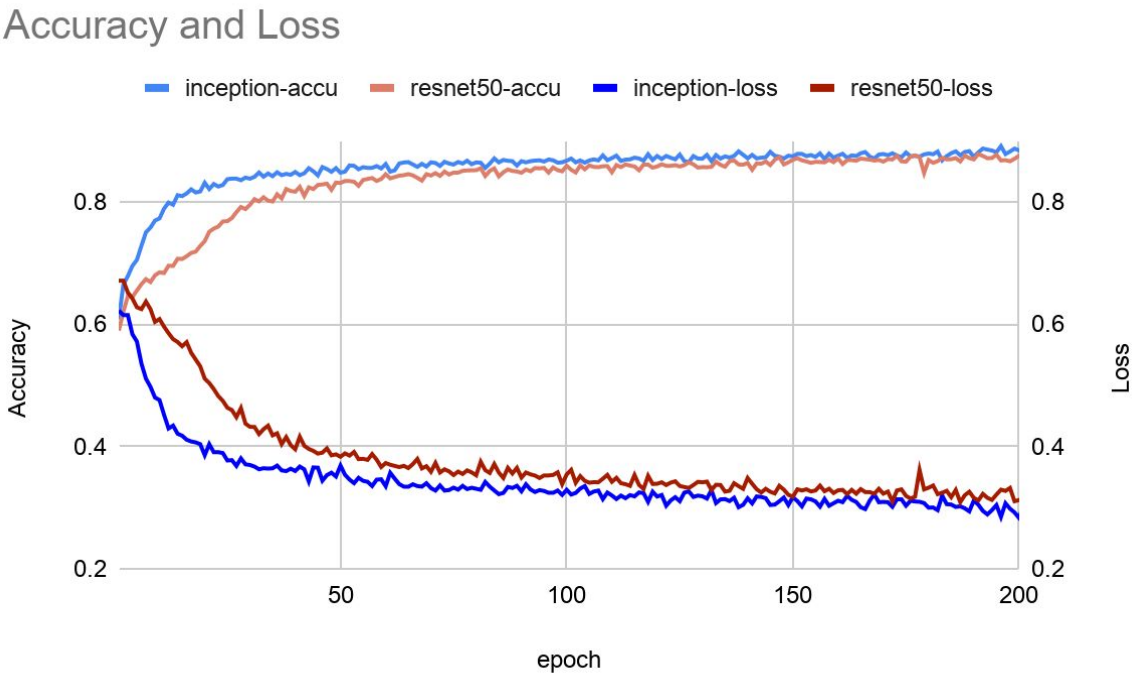

Figure 4

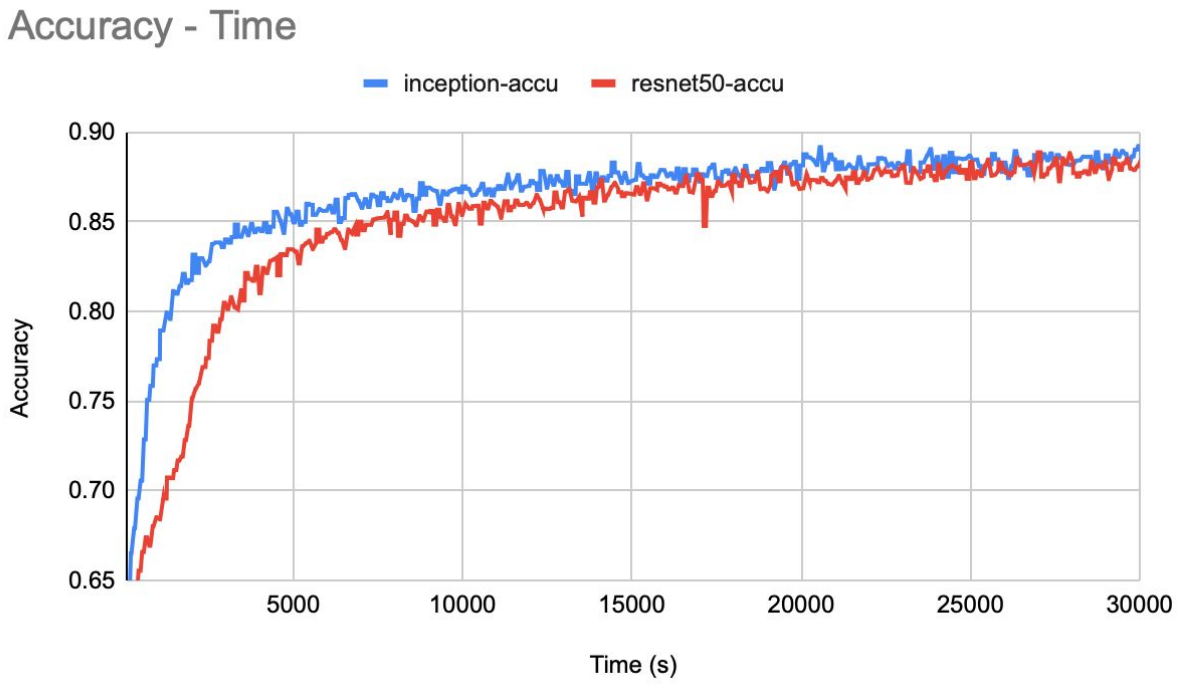

Figure 5

From figure 4, we could find that the Inception model is the better architecture for our chosen dataset in terms of accuracy. With the Inception model, our experiment on the IMDB-Wiki dataset got higher accuracy compared to the ResNet model.

Model	Training Time Per Epoch
Inception	105s
ResNet	96s

Table 1. Training Speed of Different Models

From the table 1 above, we could see that the Inception model is more accurate but it will take more time for training. However the time deviation is very small. So, from figure 5 we could find that for each time unit, the accuracy gain is larger than using the Inception model.

From the above results, we could see that the Inception model is better than the ResNet model for the dataset we selected with tradeoff between accuracy and training time. If we want to get a reasonable result in limited training time using TensorFlow Horovod, the Inception model is the better architecture for the IMDB-Wiki dataset we selected for the lab.

6. Is the GPU always better than the CPU for training with a larger dataset? Explain your observation.

Looking at the data we have collected, we must consider what is technically ‘better’. For the purpose of this question we will look at the time to train per epoch and the throughput as the measurements to determine what is ‘better’.

Figure 6 shows the training time for GPU in different batch sizes ranging from 1 to 128, we can see a max time of 108 seconds of the training time of the Inception model and a minimum time of 60 seconds to train on the ResNet model when comparing the two. This is important because, when looking at the range of times between GPU and CPU we can see just how much better GPU does.

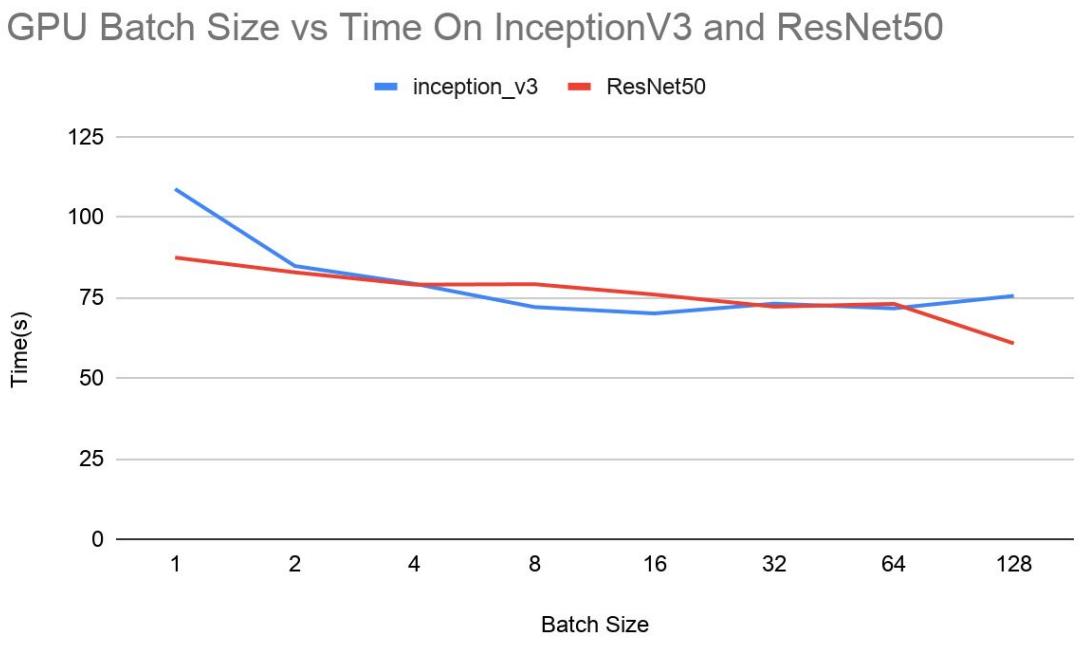

Figure 6

Looking at the range of CPU times, we see a max of roughly 1700 seconds on Inception and a minimum of 231 seconds on ResNet. This range is over much larger times compared to GPU training times. Of course, in both of these graphs, lower is better. Note that we cannot use more resources due to memory limitations on the CPU.

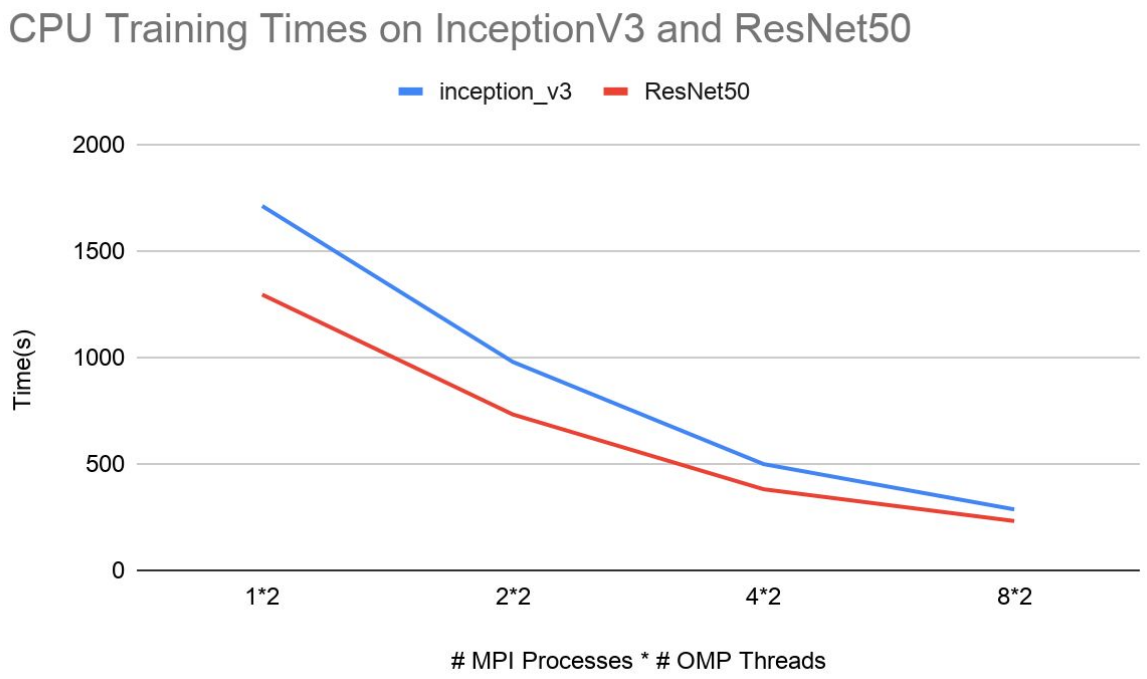

Figure 7

Overall, we can see that the range gives an accurate description of how much faster the GPU trained the model. Thus, if we are using training time as the metric, we could say yes, GPU does always train faster.

Similarly, we can also look at throughput on GPU and CPU training in figure 8 and 9. The lowest of the CPU is 1.19 (which is on inception) and the highest is 8.83 (on ResNet). This pales in comparison when looking at the GPU throughput which had the lowest of 18 and the highest of 33 on Inception and ResNet, respectively. It is of note that, in this case, higher is better. Thus, we can see that GPU trains faster, has a higher throughput and we can consider it “better”.

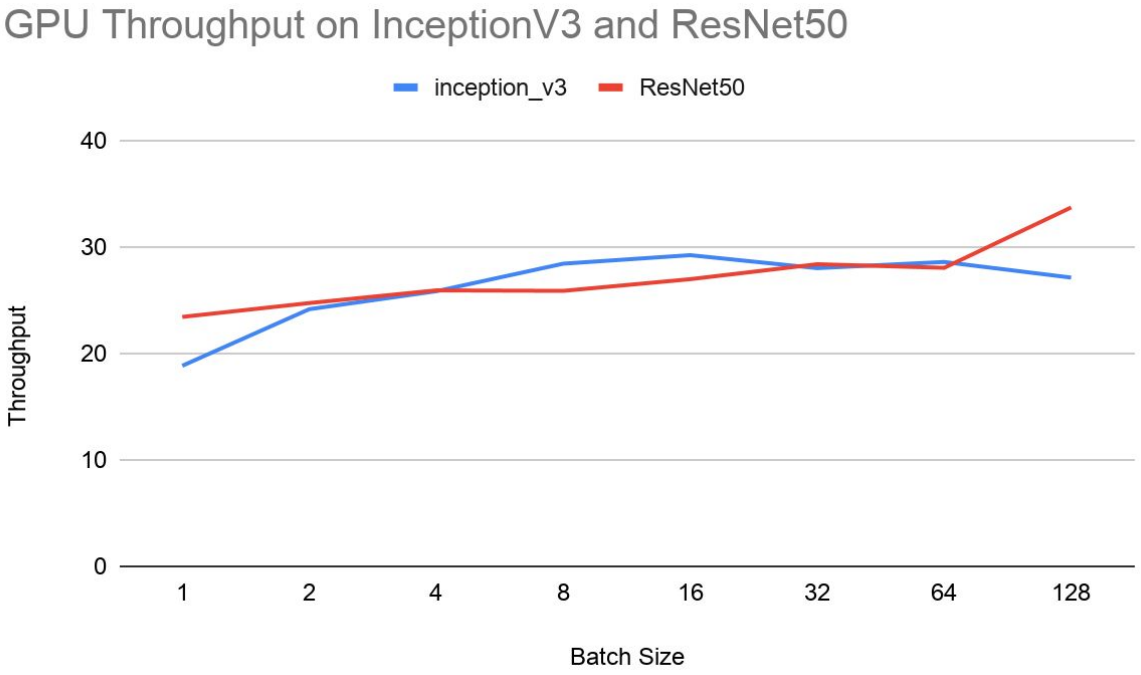

Figure 8

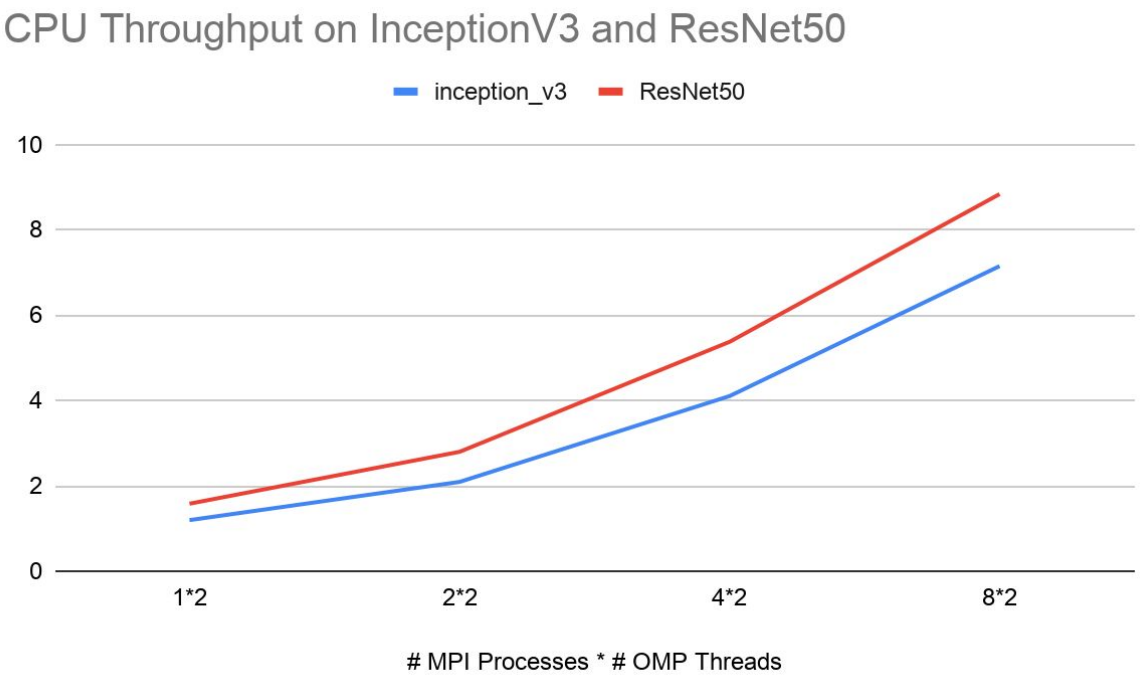

Figure 9

7. For the single-node CPU run, do you see any trend when varying the number of cores and number of threads in the application? Explain your observation

For single-node CPU run, we conducted the experiments with a varied number of MPI processes and the number of OMP threads and compared based on two types of output: Throughput, specifically, the processed images per second, and the training time in terms of the second. The experiments are performed under the same batch size of 128.

Based on the same dataset (IMDB-Wiki), we were able to run tests both with InceptionV3 and ResNet50. We first configure a different number of MPI processes each with the same two OMP threads.

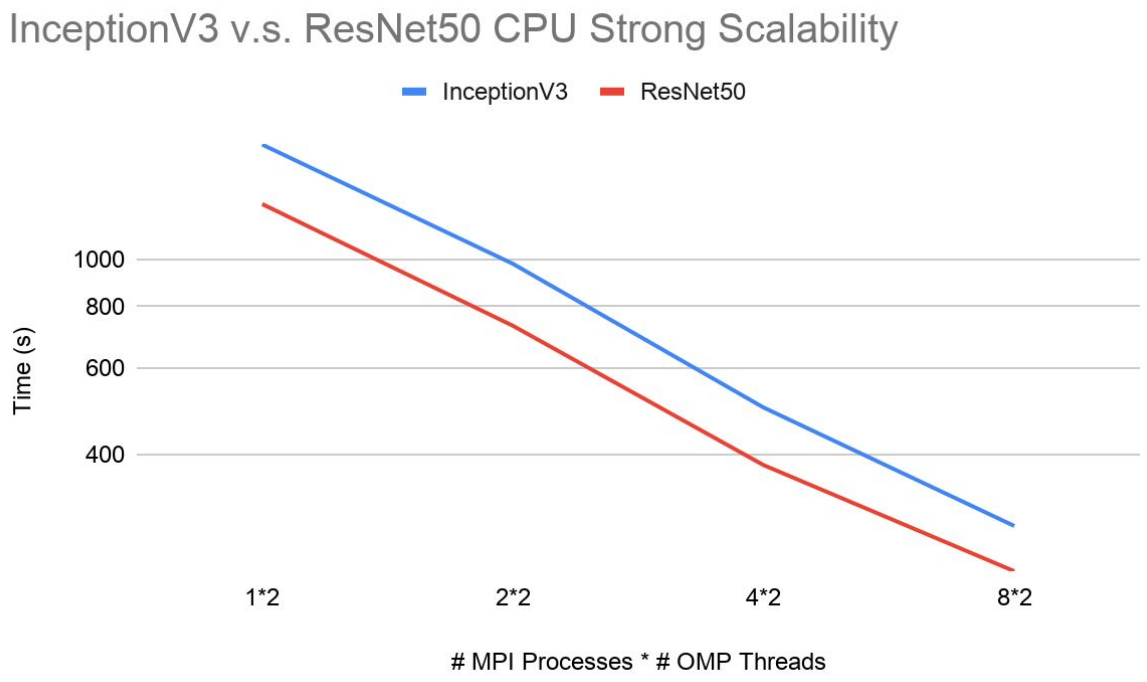

Figure 10

Figure 10 provides us a direct output of training time in terms of seconds. As the graph showed, we can see a significant decrease in training time when running with a higher number of MPI processes both with InceptionV3 and ResNet50.

In figure 11, to provide a more accurate sense of throughput of the system, we calculated the images per second processed for each of the different configurations and models.

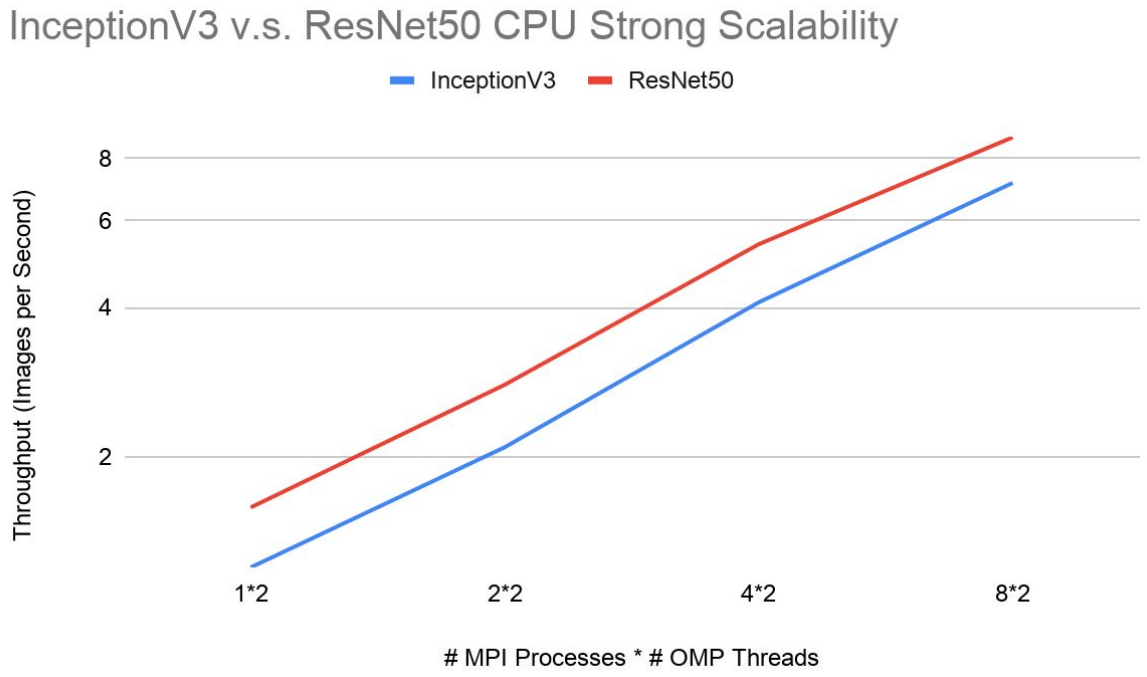

Figure 11

In terms of throughput (images per second as we mentioned before), we can see that as the number of MPI processes increases, both models experienced a significant increase in the number of images processed each second.

Next, we set the total number of resources available constant and set different numbers of threads allocated to some MPI processes.

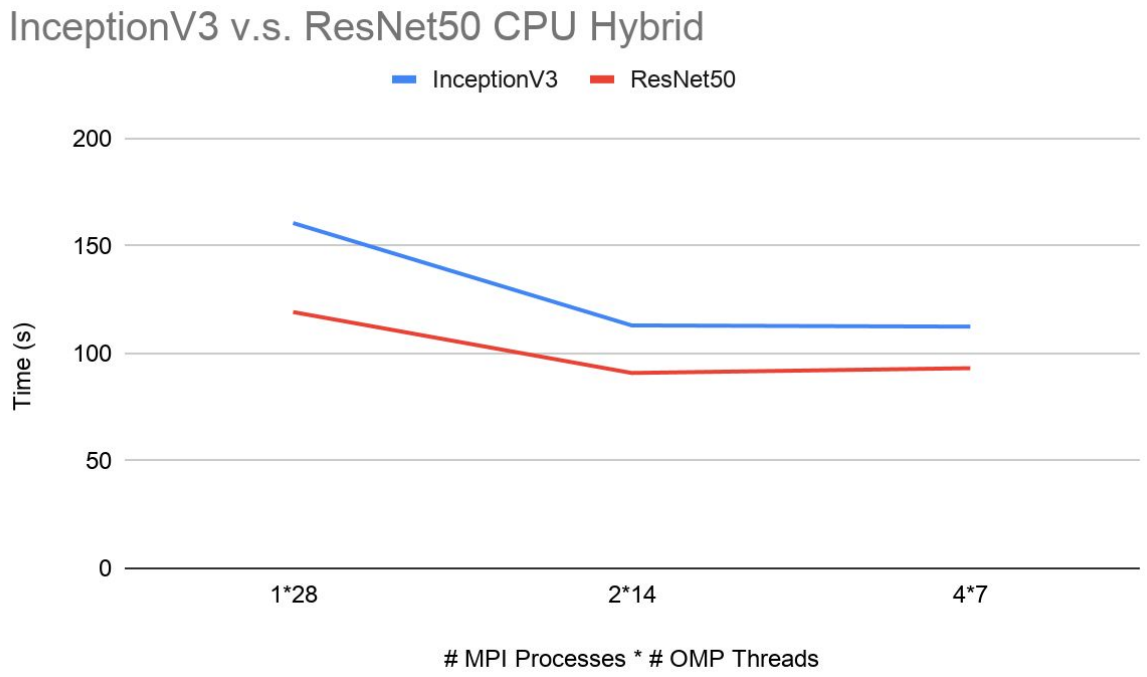

Figure 12

Figure 12 shows when we decreased the number of OMP threads allocated to each process, we first experienced a significant decrease in the training time when we went from 28 OMP threads to 14 OMP threads while we observed a rather stable trend after going from 14 OMP threads to 7 OMP threads.

Next we looked at the results regarding throughput (images per second).

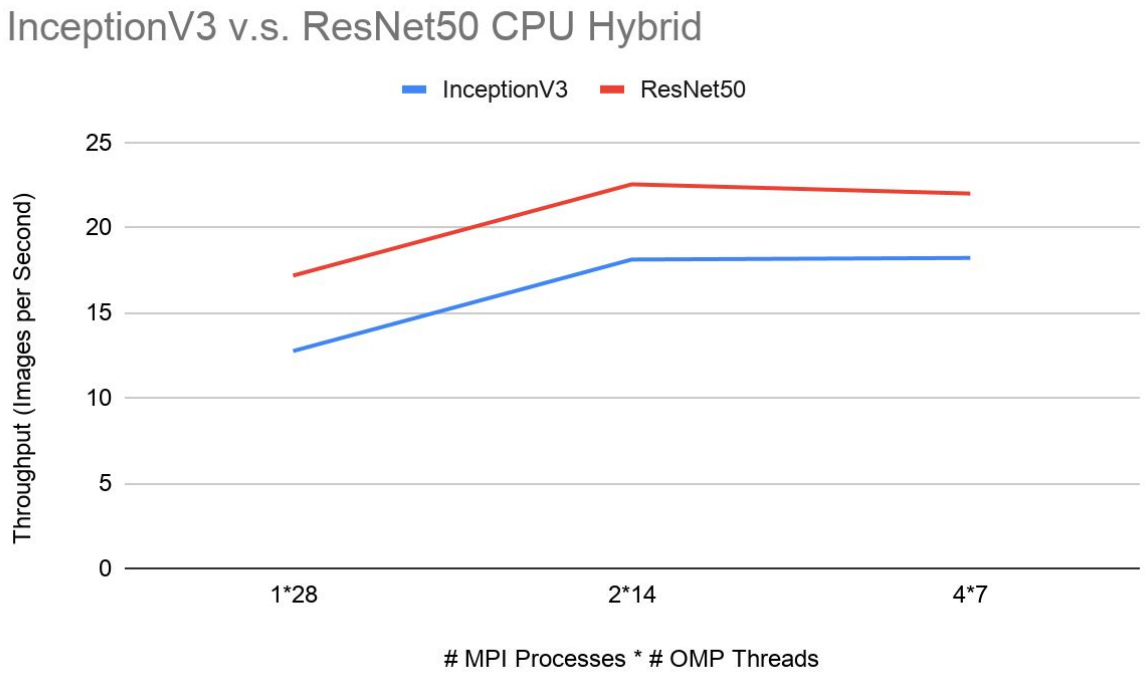

Figure 13

We observed a quite similar trend compared to training time, both models had a significant increase in number of images processed per second when going from 28 OMP threads to 14 OMP threads and a following stable trend from 14 OMP threads to 7 OMP threads.

Figure 13 can also explicitly show with a contrast that when our tests were being run under the different number of processes and number of threads, the variation of the number of processes will stand a significant portion in optimizing performance while changing the number of threads allocated to processes may have a positive impact on the performance, but a balance point should be determined to reach maximal optimization.