|  |
| --- |
| PARALLEL DISTRIBUTED COMPUTING |
| ASSIGNMENT 4 |
| OMAR ISAMIL SP23-BCS-110 |

**PART I**

**CPU vs GPU Model Training**

* Using MNIST TK10 dataset with images and labels.
* Used PyTorch to create a small CNN model.

**CPU Train:**

**Using device: cpu**

Loading dataset...

Dataset loaded and split into 8000 training samples and 2000 testing samples.

Data loaders created.

Starting training...

Epoch [1/5], Step [100/125], Loss: 0.1797

**--- Time for Epoch 1: 2.88 seconds ---**

Epoch [2/5], Step [100/125], Loss: 0.1551

**--- Time for Epoch 2: 2.90 seconds ---**

Epoch [3/5], Step [100/125], Loss: 0.0442

**--- Time for Epoch 3: 2.87 seconds ---**

Epoch [4/5], Step [100/125], Loss: 0.0332

**--- Time for Epoch 4: 4.92 seconds ---**

Epoch [5/5], Step [100/125], Loss: 0.0853

**--- Time for Epoch 5: 2.88 seconds ---**

Training finished.

**Total training time: 16.46 seconds (0.27 minutes).**

**GPU Train:**

**Using device: cuda**

Loading dataset...

Dataset loaded and split into 8000 training samples and 2000 testing samples.

Data loaders created.

Starting training...

Epoch [1/5], Step [100/125], Loss: 0.3601

**--- Time for Epoch 1: 0.31 seconds ---**

Epoch [2/5], Step [100/125], Loss: 0.1377

**--- Time for Epoch 2: 0.31 seconds ---**

Epoch [3/5], Step [100/125], Loss: 0.1063

**--- Time for Epoch 3: 0.31 seconds ---**

Epoch [4/5], Step [100/125], Loss: 0.1600

**--- Time for Epoch 4: 0.32 seconds ---**

Epoch [5/5], Step [100/125], Loss: 0.1314

**--- Time for Epoch 5: 0.31 seconds ---**

Training finished.

**Total training time: 1.55 seconds (0.03 minutes).**

**Speedup ratio:**

= CPU time / GPU time

= 16.46 / 1.55

**= 10.6 x GPU speedup**

**GPU utilization:**

|  |  |
| --- | --- |
| **CPU** | **GPU** |
| **Allocated memory on gpu device: 0** | **Allocated memory on gpu device: 20561408** |

**Speedup Reason:**

The data is large enough so that the usage of GPU is a positive rather than a negative. GPU performs more due to the availability of significantly more cores. The model is also not a really small one but has some complex tasks such as image processing, feature extraction that can’t be efficiently performed by CPU.

**PART II**

**Effect of Batch Size**

**16 Batch Size:**

* **Epoch Time:**
  + Time for Epoch 1: 1.12 seconds
  + Time for Epoch 2: 1.04 seconds
  + Time for Epoch 3: 1.04 seconds
  + Time for Epoch 4: 1.05 seconds
  + Time for Epoch 5: 1.04 seconds
* **Training time:**

Total training time: 5.28 seconds

* **GPU Memory Utilization:**

Allocated memory on gpu device: 20409344

* **Accuracy:**

Accuracy of the network on the 2000 test images: 97.75 %

**64 Batch Size:**

* **Epoch Time:**
  + Time for Epoch 1: 0.31 seconds
  + Time for Epoch 2: 0.31 seconds
  + Time for Epoch 3: 0.31 seconds
  + Time for Epoch 4: 0.32 seconds
  + Time for Epoch 5: 0.31 seconds
* **Training time:**

Total training time: 1.55 seconds

* **GPU Memory Utilization:**

Allocated memory on gpu device: 20561408

* **Accuracy:**

Accuracy of the network on the 2000 test images: 97.45 %

**256 Batch Size:**

* **Epoch Time:**
  + Time for Epoch 1: 0.25 seconds
  + Time for Epoch 2: 0.16 seconds
  + Time for Epoch 3: 0.12 seconds
  + Time for Epoch 4: 0.12 seconds
  + Time for Epoch 5: 0.12 seconds
* **Training time:**

Total training time: 0.78 seconds

* **GPU Memory Utilization:**

Allocated memory on gpu device: 20561408

* **Accuracy:**

Accuracy of the network on the 2000 test images: 95.20 %

**1024 Batch Size:**

* **Epoch Time:**
  + Time for Epoch 1: 0.16 seconds
  + Time for Epoch 2: 0.12 seconds
  + Time for Epoch 3: 0.10 seconds
  + Time for Epoch 4: 0.11 seconds
  + Time for Epoch 5: 0.11 seconds
* **Training time:**

Total training time: 0.60 seconds

* **GPU Memory Utilization:**

Allocated memory on gpu device: 23006720

* **Accuracy:**

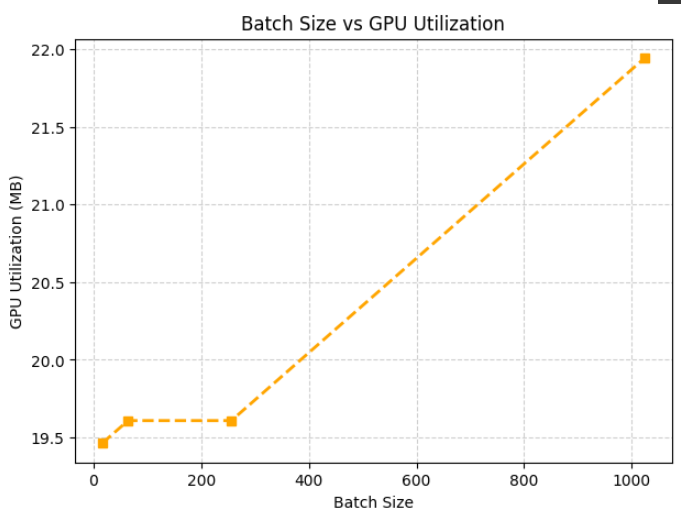
Accuracy of the network on the 2000 test images: 85.95 %

**PLOTS**

**Batch Size vs Training Time:**

**A graph with a line

AI-generated content may be incorrect.**

**Batch Size vs GPU Utilization:**

**Why does increasing batch size improve GPU efficiency up to a point?**

When the batch size is small like 16, the GPU is under utilized and it results in less efficiency. When we increase the batch size, the efficiency goes up until the GPU memory becomes saturated and then increasing the batch size does nothing on performance. Sometimes the performance decreases due to memory access and synchronization.

**Why does accuracy sometimes drop for very large batches?**

With large batches, the model updates its weights less often (since each update uses more data). With large batches, the learning rate also needs to be adjusted so that it does not drop the accuracy. Losing training accuracy is not a bad thing as it can help the model generalize better but too much loss on training accuracy can cause the model to generalize poorly.

**PART III**

**Model Complexity and GPU Utilization**

**Simple 2 Layer Model:**

* Total training time: 1.19 seconds
* Allocated GPU memory: 17501696 bytes
* Accuracy on 2000 test images: 91.80%
* A blue screen with white text

  AI-generated content may be incorrect.GPU Utilization:

**Medium 5 Layer Model:**

* Total training time: 1.71 seconds (0.03 minutes).
* Allocated memory on GPU device: 24069632
* Accuracy of the network on the 2000 test images: 97.70 %
* **A blue screen with white text

  AI-generated content may be incorrect.**GPU Utilization:

**Large CNN Model (9 layers):**

* Total training time: 1.57 seconds (0.03 minutes).
* Allocated memory on GPU device: 20561408
* Accuracy of the network on the 2000 test images: 97.50 %
* GPU Utilization:

A blue screen with white text

AI-generated content may be incorrect.

**PART IV**

**Data Loading and Bottlenecks**

* **For Single-threaded:**

Data Loading Time: 0.05 seconds

* **For Multi-threaded:**
  + **Num\_workers = 2**

Data Loading Time: 0.17 – 0.19 seconds

* + **Num\_workers = 4**

Data Loading Time: 0.21 – 0.23 seconds

* + **Num\_workers = 8**

Data Loading Time: 0.26 – 0.29 seconds

**Why inefficient data pipelines cause low GPU utilization?**

Inefficient data pipelines cause low GPU utilization as when the GPU trains the model on a batch, it has to wait for the next batch to again perform training, this keeps the GPU idle that leads to lower GPU utilization and less efficiency.

**How overlapping CPU data loading and GPU training improves performance?**

By overlapping CPU and GPU, the GPU performs training on a certain batch and then the CPU keeps the next batch ready so that when the GPU is finished with one batch, it directly starts training for another batch, causing higher efficiency and more GPU utilization.

**PART V**

**Mixed Precision Training**

**With AMP:**

* **Per Epoch Time:**
  + Time for Epoch 1: 0.71 seconds
  + Time for Epoch 2: 0.42 seconds
  + Time for Epoch 3: 0.44 seconds
  + Time for Epoch 4: 0.49 seconds
  + Time for Epoch 5: 0.50 seconds
  + Total training time: 2.57 seconds
* F**inal Accuracy:**
  + Accuracy of the network on the 2000 test images: 96.15 %
* **GPU Memory Usage:**
  + Allocated memory on gpu device: 20561408

**Without AMP:**

* **Per Epoch Time:**
  + Time for Epoch 1: 0.31 seconds
  + Time for Epoch 2: 0.31 seconds
  + Time for Epoch 3: 0.31 seconds
  + Time for Epoch 4: 0.32 seconds
  + Time for Epoch 5: 0.31 seconds
  + Total training time: 1.55 seconds
* **Final Accuracy:**
  + Accuracy of the network on the 2000 test images: 97.45 %
* **GPU Memory Usage:**
  + Allocated memory on gpu device: 20561408

**How FP16 training improves throughput?**

FP16 (half precision) uses 16 bits per value instead of 32 bits (FP32). This halves the amount of data moved between GPU memory and compute units. On modern GPUs (especially with Tensor Cores — e.g., NVIDIA Turing, Ampere, Ada), this means faster memory access and higher arithmetic throughput.

**When it might cause numerical instability?**

Too high a learning rate or lack of normalization (BatchNorm, LayerNorm) can amplify FP16 instability. Training might diverge or produce NaN losses. FP16 can only represent numbers roughly between **6e−8 and 6e4**. If your gradients or activations are smaller/larger, they can **underflow to 0** or **overflow to inf**.

**DISCUSSION QUESTIONS:**

**1. What factors most affect GPU training performance (batch size, model size, precision, data pipeline)?**

Factors like batch size, data pipeline can significantly impact the training performance as batch size can tell how much data to train per epoch and a wrong number can hurt training performance as well as accuracy. Data pipeline can increase performance by not keeping theGPU idle.

**2. Why might small models not benefit much from GPU acceleration?**

Small models cannot benefit as they are not complex enough to benefit from GPU acceleration and due to CPU GPU communication overhead, they will suffer from using GPU acceleration in some instances.

**3. How can you minimize GPU idle time during training?**

By using appropriate number of CPU threads and batch size such that the batch size is not so much big that it is not processed by CPU in time when GPU finishes training.

**4. What are the trade-offs between higher batch size and model accuracy?**

With large batches, the model updates its weights less often (since each update uses more data). With large batches, the learning rate also needs to be adjusted so that it does not drop the accuracy. Losing training accuracy is not a bad thing as it can help the model generalize better but too much loss on training accuracy can cause the model to generalize poorly.

**5. Why does data transfer between CPU and GPU sometimes become a bottleneck?**

If the data to be transferred between CPU and GPU is so much bigger than the CPU can handle and give to the GPU in time for the next training batch, then the GPU becomes idle and it is less efficient. In this case the CPU becomes a bottleneck for the GPU.