Northeastern University
INFO 7105 34274 - High Performance Parallel Machine Learning & AI, Fall 2023.
Weather Classification in Images

Project Topic

Parallelization Techniques in Deep Learning for Weather Classification in Images

Instructor: Dr. Handan Liu
Team Number: Team 9
Member Names: Dushyant Mahajan, Soeb Hussain

Title: Parallelization Techniques in Deep Learning for Weather Classification in Images

Abstract:

This report explores the implementation of parallel computing techniques in deep learning for weather image classification. It highlights the challenges posed by the intricate task of classifying various weather phenomena and the potential of deep learning models, specifically convolutional neural networks (CNNs), to address these challenges. The study emphasizes the need for improved computational efficiency in model training and preprocessing, facilitated by parallelization methods such as multiprocessing, Dask, and multi-GPU strategies.

Introduction:

Background:

Weather image classification presents a unique set of challenges, necessitating the differentiation of various weather phenomena. These phenomena range from clear skies with beautiful rainbows to the tumultuousness of thunderstorms. The application of deep learning models has demonstrated significant potential in effectively handling this complex task. However, the computational resources required for training and preprocessing these models are substantial.

Motivation:

The motivation behind enhancing weather image classification models is driven by the increasing demand for accuracy and efficiency in various pivotal applications. These include enhanced weather forecasting, climate change analysis, and advancements in computer vision. Incorporating parallelization into these models is essential as it addresses the computational challenges posed by deep learning techniques, enabling faster processing and more scalable solutions for real-time applications.

Goals:

The project aims to engineer a deep learning model that operates in parallel to classify weather images. The objectives include crafting a deep learning architecture tailored for weather image classification, applying various parallel computing techniques to bolster the model's training process, and conducting a thorough evaluation to interpret the efficacy of each parallelization strategy.

Methodology:

The study employs various techniques learned in academic settings to train the model. Sequential processing serves as the baseline, with multiprocessing, Dask, and multi-GPU approaches providing a comparative performance landscape.

Sequential Running

In the sequential approach, the study highlights the limitations imposed by Python's Global Interpreter Lock (GIL), which restricts execution to a single operation at a time, thus creating a bottleneck in multi-threaded applications. The GIL is an essential mechanism for CPython to ensure thread safety, yet it significantly hinders the potential for concurrent operations, which is a critical aspect of optimizing deep learning models.

Multi-Processing

Python's multiprocessing[1] module is a standard library that supports spawning processes using an API similar to the threading module. It bypasses the Global Interpreter Lock (GIL) by creating separate processes instead of threads, thereby enabling true parallelism.

The study leverages Python's multiprocessing library to bypass the GIL limitation by creating separate processes instead of threads. Each process executes in its memory space, and the communication between processes is managed through inter-process communication (IPC). The 'Pool' object's 'map' function is instrumental in distributing tasks across processors, showcasing marked improvements in CPU-bound tasks by harnessing the capabilities of multiple CPU cores.

Dask

Dask [2]is an open-source library for parallel computing in Python that seamlessly integrates with existing Python libraries like NumPy, pandas, and scikit-learn. It's designed to scale from single machines to large clusters, providing a flexible and efficient way to manage parallel computations.

Dask offers advanced parallelization, well-suited for both CPU-bound and memory-bound tasks. It utilizes lazy operations and task scheduling to manage and execute concurrent computations efficiently, particularly advantageous for large datasets [2].

Data Parallel (DP) Torch

Data Parallelism in PyTorch [3] (DP Torch) involves splitting the data across different GPUs. In this approach, a model is replicated across multiple GPUs, where each GPU processes a portion of the input data. After processing, the results are aggregated back to form the final output.

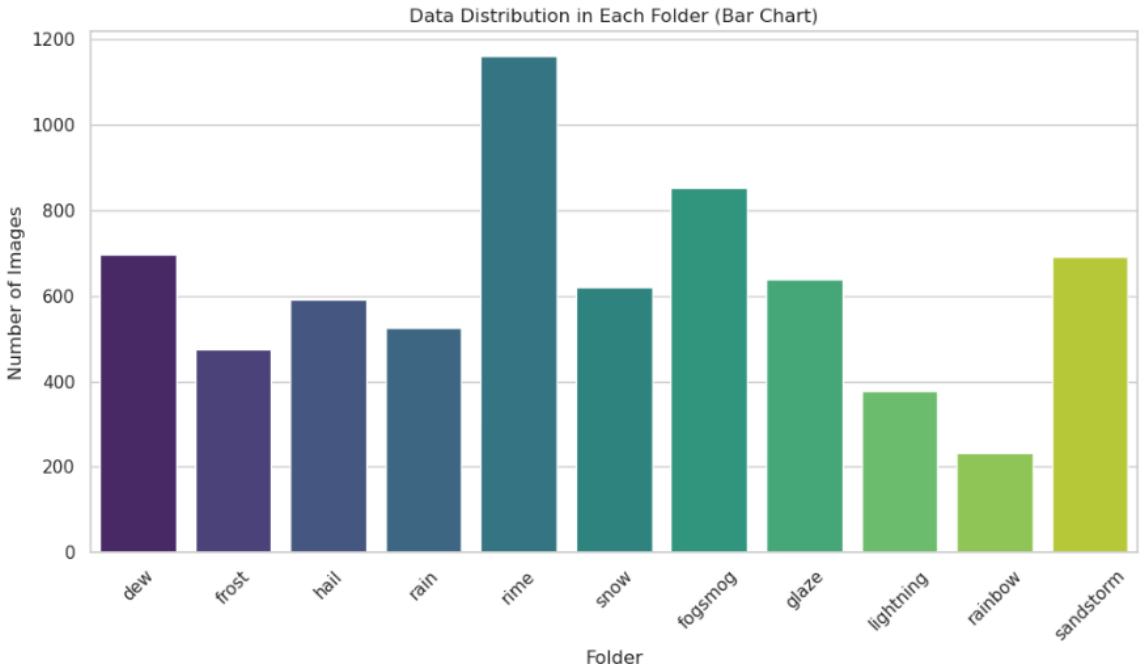
DP Torch was used to train deep learning models on large datasets of weather images[4] more rapidly. By distributing the data across multiple GPUs, the training process became significantly faster, allowing for quicker experimentation and model tuning.

Distributed Data Parallel (DDP) Torch

DDP[6] Torch is an advanced version of data parallelism in PyTorch [3]. It differs from DP Torch in that it reduces the inter-GPU communication overhead and balances the load more evenly across GPUs. This method is more efficient and scalable than standard data parallelism.

In this project, DDP Torch was essential for training models across multiple nodes in a distributed computing environment. It was particularly advantageous in scenarios where the dataset or model was too large to fit into the memory of a single GPU, thus necessitating distributed training.

Dataset



The Weather Phenomenon Database (WEAPD) serves as the foundation for this study. It contains 6,877 images across 11 different weather phenomena, providing a comprehensive dataset for training and testing the model.

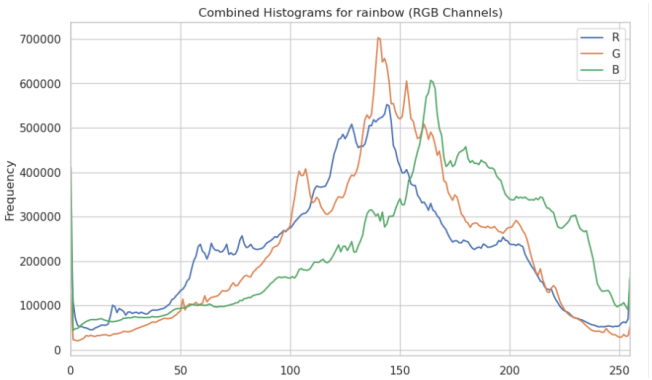
Experiment:

EDA and Data Loading

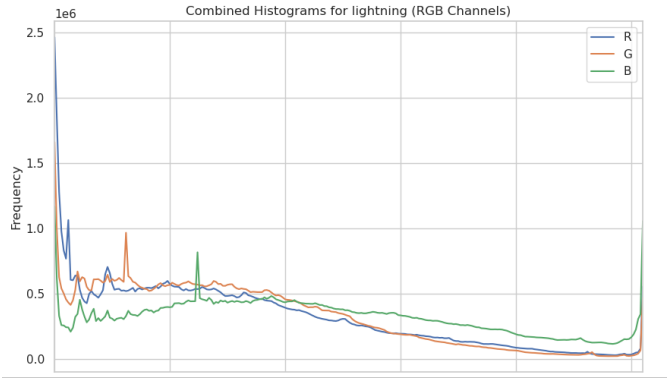
Exploratory Data Analysis (EDA) is a foundational step in data-driven projects, especially in the realm of deep learning for image classification, where understanding the underlying patterns and anomalies in the data can inform more effective model design and training strategies. In weather classification tasks, EDA involves heavy computational processes due to the large volume and high dimensionality of image data. Tasks such as histogram equalization, normalization, augmentation, and feature extraction are computationally intensive, often requiring the processing of gigabytes or even terabytes of image data to extract meaningful insights.

Histogram of some of the classes are as follows:

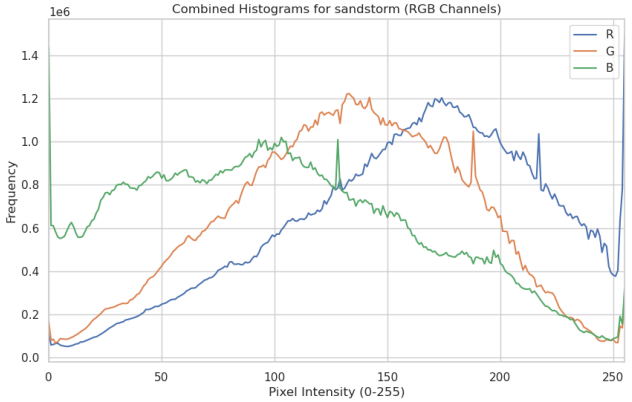
1. Rainbow



2. Lightning



3. Sandstrom



To address these computational demands, various parallelization strategies were experimented with. Sequential processing, while being the simplest form of execution, proved to be the least efficient, as indicated by the longest processing time. By contrast, Python's multiprocessing module significantly reduced the time taken by utilizing additional CPU cores. This experiment leveraged a CPU cluster with 8 cores, and as the number of cores increased, the speedup in processing time became evident, peaking at 8 cores with a substantial reduction in execution time. However, a notable drop in efficiency was observed as the CPU count reached 16, indicating a diminishing return on parallelization at this scale due to overheads like inter-process communication.

Moreover, the use of Dask provided an alternative parallel computation framework, which, while not outperforming the optimal multiprocessing setup, offered a balance between execution time and ease of use. Dask's delayed execution model allows for dynamic task scheduling and optimization [2], which is advantageous in complex EDA workflows where task dependencies are common. The flexibility and scalability of Dask make it a valuable tool for EDA in distributed computing environments, despite its longer execution time compared to the optimal 8-core multiprocessing setup [2].

This experiment highlights the importance of selecting the right parallelization strategy based on the task and available resources. It provides a clear demonstration of the trade-offs between the raw speedup offered by multiprocessing and the more sophisticated task scheduling and optimization capabilities of Dask.

Model Architecture

We employed a CNN based Neural Network using PyTorch [3] to perform the multi-label classification task on images. The layer configuration consisted of convolutional layers, batch normalization, activation functions (Tanh), pooling layers (AvgPool2d), and fully connected layers. A few dropout layers were also added to avoid overfitting in the model. The model was designed for image classification tasks, especially viable for datasets with moderate input sizes. The model is defined below.

```
class ConvolutionalModel(nn.Module):
    def __init__(self, output_size):
        super().__init__()
        self.net = nn.Sequential(
            ## ConvBlock 1
            nn.Conv2d(3, 6, kernel_size=4, stride=1, padding=0),
            # Input: (b, 3, 256, 256) || Output: (b, 6, 250, 250)
            nn.BatchNorm2d(6),
            nn.Tanh(),
            nn.AvgPool2d(kernel_size=5, stride=5, padding=0),
            # Input: (b, 6, 250, 250) || Output: (b, 6, 50, 50)

            ## ConvBlock 2
            nn.Conv2d(6, 16, kernel_size=5, stride=1, padding=0),
            # Input: (b, 6, 50, 50) || Output: (b, 16, 46, 46)
            nn.BatchNorm2d(16),
            nn.Tanh(),
            nn.AvgPool2d(kernel_size=2, stride=2, padding=0),
            # Input: (b, 16, 46, 46) || Output: (b, 16, 23, 23)

            ## ConvBlock 3
            nn.Conv2d(16, 32, kernel_size=8, stride=1, padding=0),
            # Input: (b, 16, 23, 23) || Output: (b, 32, 16, 16)
            nn.BatchNorm2d(32),
            nn.Tanh(),
            nn.AvgPool2d(kernel_size=4, stride=4, padding=0),
            # Input: (b, 32, 16, 16) || Output: (b, 32, 4, 4)

            ## ConvBlock 4
            nn.Conv2d(32, 120, kernel_size=4, stride=1, padding=0),
            # Input: (b, 32, 4, 4) || Output: (b, 120, 1, 1)
            nn.BatchNorm2d(120),
            nn.Tanh(),
            nn.Flatten(),  # flat to a vector
            # Input: (b, 120, 1, 1) || Output: (b, 120*1*1) = (b, 120)

            nn.Dropout(p=0.32), # Avoid Overfitting
            ## DenseBlock
            nn.Linear(120, 84),
            # Input: (b, 120) || Output: (b, 84)
            nn.Tanh(),
            nn.Linear(84, output_size)
            # Input: (b, 84) || Output: (b, 10)
        )
```

The training process for deep learning models, particularly for tasks such as weather classification, is a highly demanding computational endeavour [4]. The project conducted an evaluation of various training configurations to assess their efficiency and performance. This included the use of CPU-only training, single GPU, and multi-GPU setups with Data Parallel (DP) and Distributed Data Parallel (DDP) [5] frameworks.

The network comprises sequential layers—convolutional layers, batch normalization, Tanh activations, and average pooling—designed to process and transform the input image data progressively. The convolutional layers act as feature extractors, while the pooling layers reduce the spatial dimensions, helping to mitigate the computational load and prevent overfitting.

In the experiments, it was found that utilizing a single GPU significantly reduced training times compared to CPU-only training, highlighting the GPU's capability to handle parallel computations more effectively. Further enhancements in training efficiency were observed when employing two GPUs in a Data Parallel configurations, which yielded the fastest training time among the methods tested. However, an interesting observation was made when the number of GPUs was increased to four; the training time increased, which suggests that the overhead associated with managing multiple GPUs can sometimes outweigh the benefits of parallel processing.

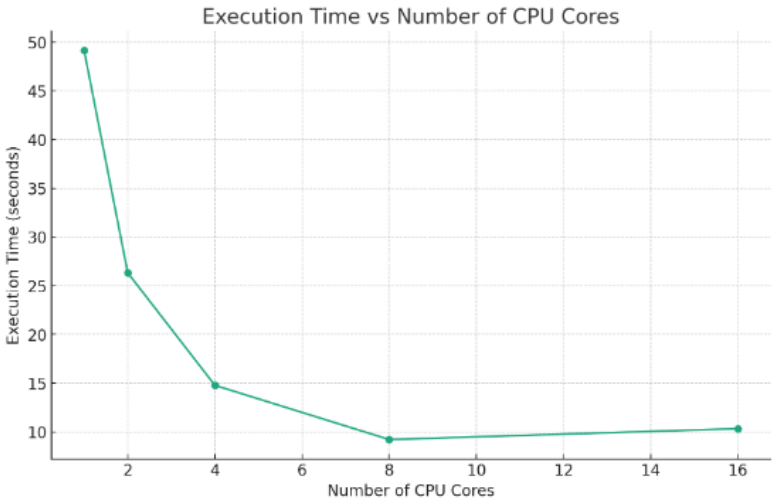
This effect was also reflected in the efficiency measurements, where the two-GPU setup showed a higher efficiency compared to the four-GPU setup [4]. These findings underscore the importance of an optimized balance between the number of processing units and the computational overhead they introduce. The experiments conducted provide valuable insights into the scalability and practical limitations of parallelized training within deep learning frameworks, particularly in the context of weather classification from image data.

Results and Analysis:

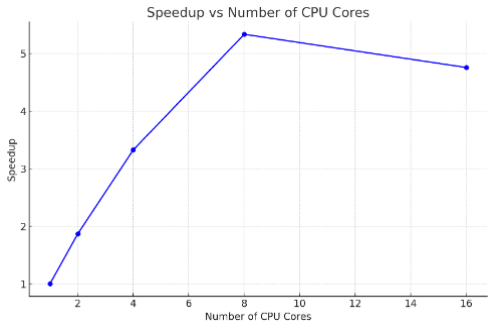
The results from the exploratory data analysis (EDA) and preprocessing phase indicate that the time taken to complete the process varies significantly with the number of CPU cores used and the parallelization method applied.

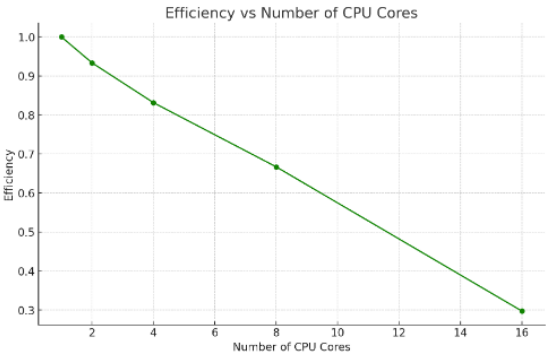
Process	Time taken(in sec)
Multiprocessing (with 1 CPU core)	49.15
Multiprocessing (with 2 CPU cores)	26.33
Multiprocessing (with 4 CPU cores)	14.78
Multiprocessing (with 8 CPU cores)	9.214
Multiprocessing (with 16 CPU cores)	10.33

When the process was executed sequentially, it took the longest time, specifically 54.66 seconds. Utilizing Python's multiprocessing module with just 1 CPU core offered a slight improvement, reducing the time to 49.15 seconds. More substantial time reductions were observed as additional CPU cores were employed. With 2 CPU cores, the time taken decreased to 26.33 seconds, and with 4 CPU cores, it further reduced to 14.78 seconds. The most significant efficiency was achieved with 8 CPU cores, which completed the task in just 9.214 seconds. However, an interesting uptick in processing time occurred when the number of CPU cores was increased to 16, taking 10.33 seconds, which was longer than the time taken with 8 CPU cores.



The speedup and efficiency as follows:

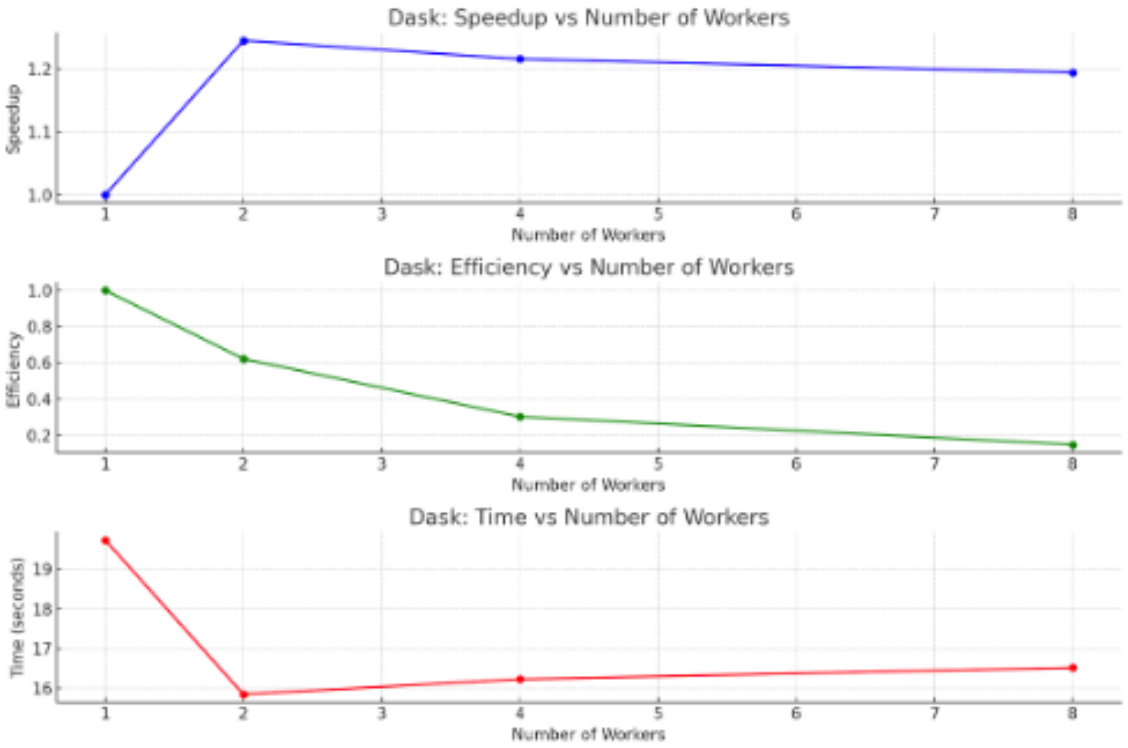




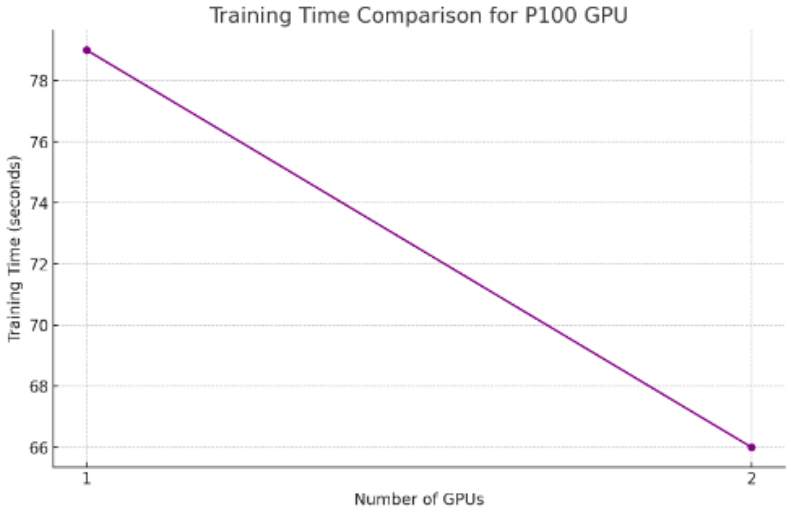
This suggests that there might be diminishing returns on performance when scaling up the number of cores beyond a certain point. Lastly, the use of Dask, a parallel computing library that scales across multiple cores with different numbers of cpu, resulted in a processing time of 19.73 seconds, 15.85 seconds, 16.23 seconds, 16.52seconds for 1,2,4,and 8 workers. While Dask did not outperform the optimal multiprocessing setup with 8 cores, it provides a more flexible and user-friendly parallel computing environment that can be beneficial for more complex or variable task pipelines.

Process	Time taken(in sec)
Dask (with 1 worker)	19.73
Dask (with 2 worker)	15.85
Dask (with 4 worker)	16.23
Dask (with 8 worker)	16.52

In the training phase of the project, various configurations were tested to determine their impact on the time taken to train the deep learning models for weather classification.

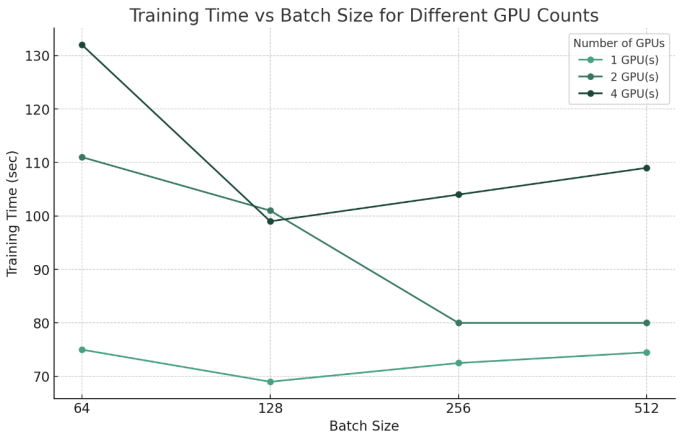


Initially, training on a CPU alone took 270 seconds. However, when the training was conducted using a single p100 GPU, the time was significantly reduced to 79 seconds. Further improvement was observed when employing Data Parallelism with 2 GPUs, which decreased the training time to 66 seconds, the most efficient of all the tested configurations.



Using Data Parallel

Interestingly, when the number of GPUs was increased to 4 under Data Parallelism, the training time increased to 111 seconds. This suggests that scaling up the number of GPUs does not always correlate with better performance, likely due to the overhead of synchronizing data across more devices.



We reran the experiment to compare performance of the code with DP implementation in multiple RTX3070GPUs for different batch sizes as shown below.

Batch size	64	128	256	512
2 GPU with DP	111s	101s	80s	79s
4 GPU with DP	132s	99s	104s	109s

In terms of speedup and efficiency, the configuration with 2 GPUs using Data Parallel achieved a speedup of 1.197 times and an efficiency of 0.598. On the other hand, the setup with 4 GPUs using Data Parallel exhibited a speedup of 0.712 times and an efficiency of 0.178, showing diminished returns with the addition of GPUs beyond a certain point.

Batch size	64	128	256	512
Speed Up	0.67	0.74	0.93	0.94
Efficiency	0.33	0.37	0.46	0.47

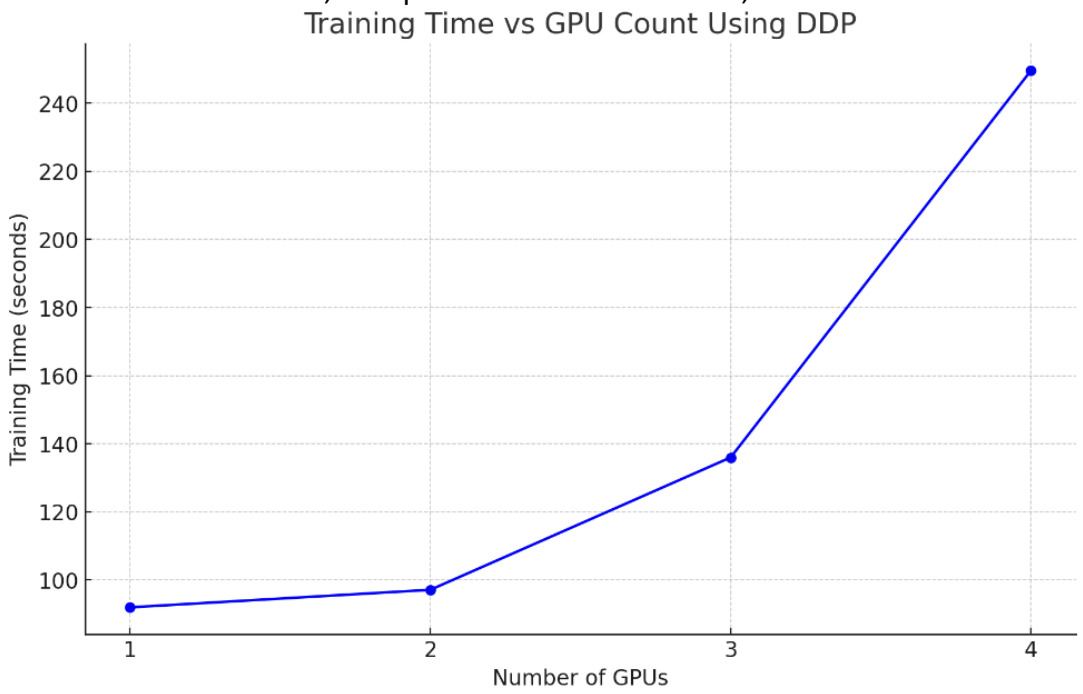
These results are crucial for understanding the trade-offs between the increased computational resources and the actual performance gains in the context of model training.

Batch size	64	128	256	512
Speed Up	0.56	0.75	0.72	0.68
Efficiency	0.14	0.18	0.18	0.17

Using Distributed Data Parallel

In the distributed data parallel (DDP)[5] training of a deep learning model using PyTorch[3] across NVIDIA RTX 3070 GPUs, an unexpected trend in performance scaling has been observed. Ideally, the addition of GPUs should result in a reduction of training time due to parallel computation; however, the results show an increase in training time as more GPUs are added: 91.98 seconds for a single GPU, 97.12 seconds for two GPUs, 136 seconds for three GPUs, and significantly longer at 249.6 seconds for four GPUs. This inverse scaling can be attributed to several potential factors. Firstly, the overhead of synchronizing gradients across multiple GPUs can increase with the number of GPUs, especially if the model is not sufficiently large to benefit from the parallelism offered by multiple GPUs. Secondly, the interconnect bandwidth can play a crucial role; if GPUs are not connected optimally the data transfer between GPUs can become a bottleneck, leading to longer training times. Thirdly, the batch size and workload distribution may not be optimally configured for the number of GPUs, which can lead to inefficient utilization of the GPUs' computational resources.

To conclude, the observed performance degradation with an increasing number of GPUs suggests that the system may be encountering scaling inefficiencies related to inter-GPU communication overhead, suboptimal resource utilization, or hardware bottlenecks.



Experiments 1:

GPU type: RTX 3070

GPU count: 1

```
Thu Dec 14 02:38:11 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.125.06   Driver Version: 525.125.06   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:03:00.0 Off |                  N/A |
|  0%   36C    P2    96W / 240W |   1649MiB /  8192MiB |     57%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA GeForce ...  On   | 00000000:04:00.0 Off |                  N/A |
|  0%   29C    P8    12W / 240W |      1MiB /  8192MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  NVIDIA GeForce ...  On   | 00000000:05:00.0 Off |                  N/A |
|  0%   30C    P8    23W / 240W |      1MiB /  8192MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  NVIDIA GeForce ...  On   | 00000000:81:00.0 Off |                  N/A |
|  0%   31C    P8    25W / 240W |      1MiB /  8192MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

Batch size:32

```
root@C.7849290:/home$ CUDA_VISIBLE_DEVICES=0 python ddp.py
[########                                ] | 20% Completed | 831.05 mslibpng wa
[#################                       ] | 44% Completed | 1.66 slibpng warn
libpng warning: iCCP: known incorrect sRGB profile
[########################################] | 100% Completed | 4.00 s
(6862, 256, 256, 3)
Number of available GPUs: 1
Rank: 0 || Epoch: 0 || Loss: 1.969 || Accuracy:  0.37 || Time: 6.56
Rank: 0 || Epoch: 1 || Loss: 1.682 || Accuracy:  0.46 || Time: 4.62
Rank: 0 || Epoch: 2 || Loss: 1.590 || Accuracy:  0.50 || Time: 4.97
Rank: 0 || Epoch: 3 || Loss: 1.537 || Accuracy:  0.52 || Time: 4.51
Rank: 0 || Epoch: 4 || Loss: 1.486 || Accuracy:  0.53 || Time: 4.41
Rank: 0 || Epoch: 5 || Loss: 1.444 || Accuracy:  0.54 || Time: 4.36
Rank: 0 || Epoch: 6 || Loss: 1.385 || Accuracy:  0.55 || Time: 4.43
Rank: 0 || Epoch: 7 || Loss: 1.306 || Accuracy:  0.58 || Time: 4.52
Rank: 0 || Epoch: 8 || Loss: 1.236 || Accuracy:  0.61 || Time: 4.47
Rank: 0 || Epoch: 9 || Loss: 1.194 || Accuracy:  0.61 || Time: 4.55
Rank: 0 || Epoch: 10 || Loss: 1.131 || Accuracy:  0.63 || Time: 4.48
Rank: 0 || Epoch: 11 || Loss: 1.092 || Accuracy:  0.64 || Time: 4.47
Rank: 0 || Epoch: 12 || Loss: 1.074 || Accuracy:  0.65 || Time: 4.30
Rank: 0 || Epoch: 13 || Loss: 1.038 || Accuracy:  0.66 || Time: 4.32
Rank: 0 || Epoch: 14 || Loss: 1.024 || Accuracy:  0.65 || Time: 4.31
Rank: 0 || Epoch: 15 || Loss: 0.994 || Accuracy:  0.67 || Time: 4.53
Rank: 0 || Epoch: 16 || Loss: 0.987 || Accuracy:  0.67 || Time: 4.52
Rank: 0 || Epoch: 17 || Loss: 0.957 || Accuracy:  0.69 || Time: 4.53
Rank: 0 || Epoch: 18 || Loss: 0.946 || Accuracy:  0.69 || Time: 4.50
Rank: 0 || Epoch: 19 || Loss: 0.926 || Accuracy:  0.69 || Time: 4.62
91.9847617149353
```

Batch size:64

```
Rank: 0 || Epoch: 8 || Loss: 1.338 || Accuracy:  0.58 || Time: 4.57
Rank: 0 || Epoch: 9 || Loss: 1.319 || Accuracy:  0.58 || Time: 4.63
Rank: 0 || Epoch: 10 || Loss: 1.245 || Accuracy:  0.61 || Time: 4.60
Rank: 0 || Epoch: 11 || Loss: 1.230 || Accuracy:  0.61 || Time: 4.60
Rank: 0 || Epoch: 12 || Loss: 1.196 || Accuracy:  0.61 || Time: 4.58
Rank: 0 || Epoch: 13 || Loss: 1.154 || Accuracy:  0.62 || Time: 4.56
Rank: 0 || Epoch: 14 || Loss: 1.139 || Accuracy:  0.63 || Time: 4.62
Rank: 0 || Epoch: 15 || Loss: 1.125 || Accuracy:  0.63 || Time: 4.60
Rank: 0 || Epoch: 16 || Loss: 1.090 || Accuracy:  0.64 || Time: 4.63
Rank: 0 || Epoch: 17 || Loss: 1.083 || Accuracy:  0.65 || Time: 4.54
Rank: 0 || Epoch: 18 || Loss: 1.043 || Accuracy:  0.66 || Time: 4.57
Rank: 0 || Epoch: 19 || Loss: 1.023 || Accuracy:  0.67 || Time: 4.57
Number of available GPUs: 1
93.80670881271362
```

Batch size: 128

```
Rank: 0 || Epoch: 9 || Loss: 1.475 || Accuracy:  0.54 || Time: 4.46
Rank: 0 || Epoch: 10 || Loss: 1.452 || Accuracy:  0.54 || Time: 4.45
Rank: 0 || Epoch: 11 || Loss: 1.414 || Accuracy:  0.56 || Time: 4.41
Rank: 0 || Epoch: 12 || Loss: 1.400 || Accuracy:  0.57 || Time: 4.42
Rank: 0 || Epoch: 13 || Loss: 1.364 || Accuracy:  0.58 || Time: 4.40
Rank: 0 || Epoch: 14 || Loss: 1.345 || Accuracy:  0.58 || Time: 4.37
Rank: 0 || Epoch: 15 || Loss: 1.314 || Accuracy:  0.59 || Time: 4.45
Rank: 0 || Epoch: 16 || Loss: 1.320 || Accuracy:  0.59 || Time: 4.42
Rank: 0 || Epoch: 17 || Loss: 1.311 || Accuracy:  0.59 || Time: 4.48
Rank: 0 || Epoch: 18 || Loss: 1.304 || Accuracy:  0.58 || Time: 4.47
Rank: 0 || Epoch: 19 || Loss: 1.234 || Accuracy:  0.61 || Time: 4.46
Number of available GPUs: 1
91.07798266410828
```

Batch size: 256

```
Rank: 0 || Epoch: 9 || Loss: 1.604 || Accuracy:  0.50 || Time: 4.55
Rank: 0 || Epoch: 10 || Loss: 1.578 || Accuracy:  0.51 || Time: 4.51
Rank: 0 || Epoch: 11 || Loss: 1.553 || Accuracy:  0.52 || Time: 4.52
Rank: 0 || Epoch: 12 || Loss: 1.531 || Accuracy:  0.52 || Time: 4.53
Rank: 0 || Epoch: 13 || Loss: 1.499 || Accuracy:  0.54 || Time: 4.61
Rank: 0 || Epoch: 14 || Loss: 1.483 || Accuracy:  0.54 || Time: 4.57
Rank: 0 || Epoch: 15 || Loss: 1.546 || Accuracy:  0.52 || Time: 4.93
Rank: 0 || Epoch: 16 || Loss: 1.483 || Accuracy:  0.54 || Time: 4.64
Rank: 0 || Epoch: 17 || Loss: 1.443 || Accuracy:  0.55 || Time: 4.91
Rank: 0 || Epoch: 18 || Loss: 1.427 || Accuracy:  0.56 || Time: 4.61
Rank: 0 || Epoch: 19 || Loss: 1.392 || Accuracy:  0.57 || Time: 4.63
Number of available GPUs: 1
93.35435628890991
```

Experiments 2:

GPU type: RTX 3070

GPU count: 2

```
root@C.7849290:/home$ nvidia-smi
Thu Dec 14 02:48:46 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.125.06   Driver Version: 525.125.06   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:03:00.0 Off |                  N/A |
|  0%   42C    P2    82W / 240W |   2183MiB /  8192MiB |     46%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA GeForce ...  On   | 00000000:04:00.0 Off |                  N/A |
|  0%   38C    P2    71W / 240W |   1589MiB /  8192MiB |     49%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  NVIDIA GeForce ...  On   | 00000000:05:00.0 Off |                  N/A |
|  0%   31C    P8    23W / 240W |      1MiB /  8192MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  NVIDIA GeForce ...  On   | 00000000:81:00.0 Off |                  N/A |
|  0%   31C    P8    25W / 240W |      1MiB /  8192MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

Batch size: 32

```
Rank: 1 || Epoch: 11 || Loss: 1.277 || Accuracy:  0.58 || Time: 4.18
Rank: 0 || Epoch: 11 || Loss: 1.274 || Accuracy:  0.59 || Time: 4.19
Rank: 0 || Epoch: 12 || Loss: 1.222 || Accuracy:  0.61 || Time: 4.05
Rank: 1 || Epoch: 12 || Loss: 1.190 || Accuracy:  0.62 || Time: 4.05
Rank: 0 || Epoch: 13 || Loss: 1.195 || Accuracy:  0.61 || Time: 4.01
Rank: 1 || Epoch: 13 || Loss: 1.203 || Accuracy:  0.62 || Time: 4.01
Rank: 1 || Epoch: 14 || Loss: 1.138 || Accuracy:  0.63 || Time: 4.08
Rank: 0 || Epoch: 14 || Loss: 1.145 || Accuracy:  0.63 || Time: 4.08
Rank: 0 || Epoch: 15 || Loss: 1.158 || Accuracy:  0.62 || Time: 3.95
Rank: 1 || Epoch: 15 || Loss: 1.122 || Accuracy:  0.63 || Time: 3.96
Rank: 0 || Epoch: 16 || Loss: 1.147 || Accuracy:  0.63 || Time: 4.07
Rank: 1 || Epoch: 16 || Loss: 1.119 || Accuracy:  0.64 || Time: 4.07
Rank: 0 || Epoch: 17 || Loss: 1.114 || Accuracy:  0.63 || Time: 4.02
Rank: 1 || Epoch: 17 || Loss: 1.113 || Accuracy:  0.64 || Time: 4.02
Rank: 1 || Epoch: 18 || Loss: 1.049 || Accuracy:  0.66 || Time: 4.57
Rank: 0 || Epoch: 18 || Loss: 1.047 || Accuracy:  0.66 || Time: 4.57
Rank: 0 || Epoch: 19 || Loss: 1.035 || Accuracy:  0.66 || Time: 4.18
97.1252875328064
Rank: 1 || Epoch: 19 || Loss: 1.029 || Accuracy:  0.66 || Time: 4.18
97.12546634674072
```

Batch size: 64

```
Rank: 0 || Epoch: 16 || Loss: 1.337 || Accuracy:  0.58 || Time: 3.69
Rank: 1 || Epoch: 16 || Loss: 1.346 || Accuracy:  0.57 || Time: 3.69
Rank: 1 || Epoch: 17 || Loss: 1.331 || Accuracy:  0.58 || Time: 3.51
Rank: 0 || Epoch: 17 || Loss: 1.305 || Accuracy:  0.60 || Time: 3.51
Rank: 1 || Epoch: 18 || Loss: 1.289 || Accuracy:  0.59 || Time: 3.52
Rank: 0 || Epoch: 18 || Loss: 1.303 || Accuracy:  0.59 || Time: 3.52
Rank: 1 || Epoch: 19 || Loss: 1.281 || Accuracy:  0.60 || Time: 3.49
Number of available GPUs: 2
79.61142706871033
Rank: 0 || Epoch: 19 || Loss: 1.276 || Accuracy:  0.60 || Time: 3.49
Number of available GPUs: 2
79.61195635795593
```

Batch size: 128

```
Rank: 0 || Epoch: 14 || Loss: 1.490 || Accuracy:  0.54 || Time: 3.45
Rank: 1 || Epoch: 14 || Loss: 1.491 || Accuracy:  0.54 || Time: 3.45
Rank: 1 || Epoch: 15 || Loss: 1.458 || Accuracy:  0.54 || Time: 2.74
Rank: 0 || Epoch: 15 || Loss: 1.455 || Accuracy:  0.55 || Time: 2.74
Rank: 0 || Epoch: 16 || Loss: 1.459 || Accuracy:  0.55 || Time: 2.66
Rank: 1 || Epoch: 16 || Loss: 1.471 || Accuracy:  0.54 || Time: 2.66
Rank: 0 || Epoch: 17 || Loss: 1.444 || Accuracy:  0.56 || Time: 3.09
Rank: 1 || Epoch: 17 || Loss: 1.437 || Accuracy:  0.56 || Time: 3.09
Rank: 1 || Epoch: 18 || Loss: 1.433 || Accuracy:  0.55 || Time: 2.84
Rank: 0 || Epoch: 18 || Loss: 1.419 || Accuracy:  0.56 || Time: 2.84
Rank: 0 || Epoch: 19 || Loss: 1.395 || Accuracy:  0.57 || Time: 2.68
Number of available GPUs: 2
63.36701250076294
Rank: 1 || Epoch: 19 || Loss: 1.383 || Accuracy:  0.58 || Time: 2.68
Number of available GPUs: 2
63.366753578186035
```

Batch size: 256

```
Rank: 1 || Epoch: 11 || Loss: 1.741 || Accuracy:  0.46 || Time: 4.40
Rank: 1 || Epoch: 12 || Loss: 1.700 || Accuracy:  0.48 || Time: 3.38
Rank: 0 || Epoch: 12 || Loss: 1.703 || Accuracy:  0.47 || Time: 3.39
Rank: 1 || Epoch: 13 || Loss: 1.681 || Accuracy:  0.48 || Time: 3.24
Rank: 0 || Epoch: 13 || Loss: 1.700 || Accuracy:  0.46 || Time: 3.24
Rank: 1 || Epoch: 14 || Loss: 1.666 || Accuracy:  0.49 || Time: 3.26
Rank: 0 || Epoch: 14 || Loss: 1.676 || Accuracy:  0.48 || Time: 3.26
Rank: 1 || Epoch: 15 || Loss: 1.653 || Accuracy:  0.48 || Time: 3.20
Rank: 0 || Epoch: 15 || Loss: 1.643 || Accuracy:  0.50 || Time: 3.20
Rank: 0 || Epoch: 16 || Loss: 1.610 || Accuracy:  0.50 || Time: 3.09
Rank: 1 || Epoch: 16 || Loss: 1.636 || Accuracy:  0.49 || Time: 3.09
Rank: 1 || Epoch: 17 || Loss: 1.616 || Accuracy:  0.50 || Time: 3.26
Rank: 0 || Epoch: 17 || Loss: 1.637 || Accuracy:  0.48 || Time: 3.26
Rank: 1 || Epoch: 18 || Loss: 1.599 || Accuracy:  0.51 || Time: 3.47
Rank: 0 || Epoch: 18 || Loss: 1.593 || Accuracy:  0.50 || Time: 3.47
Rank: 0 || Epoch: 19 || Loss: 1.592 || Accuracy:  0.50 || Time: 3.37
Number of available GPUs: 2
66.64922547340393
Rank: 1 || Epoch: 19 || Loss: 1.578 || Accuracy:  0.51 || Time: 3.38
Number of available GPUs: 2
66.64951729774475
```

Experiments 3:

GPU type: RTX 3070

GPU count: 3

```
root@C.7849290:/home$ nvidia-smi
Thu Dec 14 02:52:42 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.125.06   Driver Version: 525.125.06   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:03:00.0 Off |                  N/A |
|  0%   38C    P2    69W / 240W |   2777MiB /  8192MiB |     16%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA GeForce ...  On   | 00000000:04:00.0 Off |                  N/A |
|  0%   36C    P2    61W / 240W |   1589MiB /  8192MiB |     64%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  NVIDIA GeForce ...  On   | 00000000:05:00.0 Off |                  N/A |
|  0%   37C    P2    74W / 240W |   1589MiB /  8192MiB |     31%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  NVIDIA GeForce ...  On   | 00000000:81:00.0 Off |                  N/A |
|  0%   31C    P8    25W / 240W |      1MiB /  8192MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

Batch size: 32

```
Rank: 1 || Epoch: 11 || Loss: 1.277 || Accuracy:  0.58 || Time: 4.18
Rank: 0 || Epoch: 11 || Loss: 1.274 || Accuracy:  0.59 || Time: 4.19
Rank: 0 || Epoch: 12 || Loss: 1.222 || Accuracy:  0.61 || Time: 4.05
Rank: 1 || Epoch: 12 || Loss: 1.190 || Accuracy:  0.62 || Time: 4.05
Rank: 0 || Epoch: 13 || Loss: 1.195 || Accuracy:  0.61 || Time: 4.01
Rank: 1 || Epoch: 13 || Loss: 1.203 || Accuracy:  0.62 || Time: 4.01
Rank: 1 || Epoch: 14 || Loss: 1.138 || Accuracy:  0.63 || Time: 4.08
Rank: 0 || Epoch: 14 || Loss: 1.145 || Accuracy:  0.63 || Time: 4.08
Rank: 0 || Epoch: 15 || Loss: 1.158 || Accuracy:  0.62 || Time: 3.95
Rank: 1 || Epoch: 15 || Loss: 1.122 || Accuracy:  0.63 || Time: 3.96
Rank: 0 || Epoch: 16 || Loss: 1.147 || Accuracy:  0.63 || Time: 4.07
Rank: 1 || Epoch: 16 || Loss: 1.119 || Accuracy:  0.64 || Time: 4.07
Rank: 0 || Epoch: 17 || Loss: 1.114 || Accuracy:  0.63 || Time: 4.02
Rank: 1 || Epoch: 17 || Loss: 1.113 || Accuracy:  0.64 || Time: 4.02
Rank: 1 || Epoch: 18 || Loss: 1.049 || Accuracy:  0.66 || Time: 4.57
Rank: 0 || Epoch: 18 || Loss: 1.047 || Accuracy:  0.66 || Time: 4.57
Rank: 0 || Epoch: 19 || Loss: 1.035 || Accuracy:  0.66 || Time: 4.18
97.1252875328064
Rank: 1 || Epoch: 19 || Loss: 1.029 || Accuracy:  0.66 || Time: 4.18
97.12546634674072
```

Batch size: 64

```
Rank: 2 || Epoch: 17 || Loss: 1.430 || Accuracy:  0.56 || Time: 5.84
Rank: 0 || Epoch: 18 || Loss: 1.404 || Accuracy:  0.55 || Time: 4.78
Rank: 2 || Epoch: 18 || Loss: 1.415 || Accuracy:  0.57 || Time: 4.78
Rank: 1 || Epoch: 18 || Loss: 1.407 || Accuracy:  0.57 || Time: 4.78
Rank: 1 || Epoch: 19 || Loss: 1.392 || Accuracy:  0.57 || Time: 4.72
Number of available GPUs: 3
101.59220385551453
Rank: 2 || Epoch: 19 || Loss: 1.368 || Accuracy:  0.58 || Time: 4.72
Number of available GPUs: 3
101.59203910827637
Rank: 0 || Epoch: 19 || Loss: 1.397 || Accuracy:  0.57 || Time: 4.72
Number of available GPUs: 3
101.5934100151062
```

Batch size: 128

```
Rank: 2 || Epoch: 17 || Loss: 1.591 || Accuracy:  0.50 || Time: 4.21
Rank: 2 || Epoch: 18 || Loss: 1.565 || Accuracy:  0.51 || Time: 3.93
Rank: 0 || Epoch: 18 || Loss: 1.519 || Accuracy:  0.53 || Time: 3.93
Rank: 1 || Epoch: 18 || Loss: 1.548 || Accuracy:  0.51 || Time: 3.93
Rank: 1 || Epoch: 19 || Loss: 1.530 || Accuracy:  0.52 || Time: 4.36
Number of available GPUs: 3
87.89397931098938
Rank: 0 || Epoch: 19 || Loss: 1.552 || Accuracy:  0.51 || Time: 4.36
Number of available GPUs: 3
87.89493036270142
Rank: 2 || Epoch: 19 || Loss: 1.552 || Accuracy:  0.51 || Time: 4.36
Number of available GPUs: 3
87.89747834205627
```

Batch size: 256

```
Rank: 0 || Epoch: 14 || Loss: 1.744 || Accuracy:  0.45 || Time: 4.14
Rank: 2 || Epoch: 15 || Loss: 1.714 || Accuracy:  0.46 || Time: 4.36
Rank: 1 || Epoch: 15 || Loss: 1.709 || Accuracy:  0.46 || Time: 4.36
Rank: 0 || Epoch: 15 || Loss: 1.711 || Accuracy:  0.45 || Time: 4.36
Rank: 0 || Epoch: 16 || Loss: 1.740 || Accuracy:  0.44 || Time: 4.02
Rank: 1 || Epoch: 16 || Loss: 1.730 || Accuracy:  0.46 || Time: 4.02
Rank: 2 || Epoch: 16 || Loss: 1.725 || Accuracy:  0.46 || Time: 4.02
Rank: 0 || Epoch: 17 || Loss: 1.696 || Accuracy:  0.47 || Time: 3.99
Rank: 2 || Epoch: 17 || Loss: 1.706 || Accuracy:  0.47 || Time: 3.99
Rank: 1 || Epoch: 17 || Loss: 1.743 || Accuracy:  0.46 || Time: 3.99
Rank: 1 || Epoch: 18 || Loss: 1.699 || Accuracy:  0.46 || Time: 4.62
Rank: 2 || Epoch: 18 || Loss: 1.741 || Accuracy:  0.45 || Time: 4.62
Rank: 0 || Epoch: 18 || Loss: 1.715 || Accuracy:  0.46 || Time: 4.63
Rank: 0 || Epoch: 19 || Loss: 1.675 || Accuracy:  0.46 || Time: 3.97
Rank: 2 || Epoch: 19 || Loss: 1.650 || Accuracy:  0.48 || Time: 3.97
Number of available GPUs: 3
Number of available GPUs: 3
88.003422498703
88.00323367118835
Rank: 1 || Epoch: 19 || Loss: 1.687 || Accuracy:  0.45 || Time: 3.97
Number of available GPUs: 3
88.00375008583069
```

Experiments 4:

GPU type: RTX 3070

GPU count: 4

```
root@C.7849290:/home$ nvidia-smi
Thu Dec 14 03:09:10 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.125.06   Driver Version: 525.125.06   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:03:00.0 Off |                  N/A |
|  0%   38C    P2    51W / 240W |   4143MiB /  8192MiB |     15%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA GeForce ...  On   | 00000000:04:00.0 Off |                  N/A |
|  0%   36C    P2    43W / 240W |   2357MiB /  8192MiB |     22%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  NVIDIA GeForce ...  On   | 00000000:05:00.0 Off |                  N/A |
|  0%   39C    P2    58W / 240W |   2357MiB /  8192MiB |     76%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  NVIDIA GeForce ...  On   | 00000000:81:00.0 Off |                  N/A |
|  0%   39C    P2    64W / 240W |   2357MiB /  8192MiB |     35%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

Batch size: 32

```
Rank: 1 || Epoch: 16 || Loss: 1.313 || Accuracy:  0.59 || Time: 12.11
Rank: 0 || Epoch: 16 || Loss: 1.314 || Accuracy:  0.58 || Time: 12.11
Rank: 3 || Epoch: 16 || Loss: 1.309 || Accuracy:  0.58 || Time: 12.11
Rank: 3 || Epoch: 17 || Loss: 1.300 || Accuracy:  0.59 || Time: 12.35
Rank: 1 || Epoch: 17 || Loss: 1.301 || Accuracy:  0.59 || Time: 12.35
Rank: 0 || Epoch: 17 || Loss: 1.261 || Accuracy:  0.61 || Time: 12.35
Rank: 2 || Epoch: 17 || Loss: 1.284 || Accuracy:  0.59 || Time: 12.35
Rank: 1 || Epoch: 18 || Loss: 1.245 || Accuracy:  0.61 || Time: 12.48
Rank: 0 || Epoch: 18 || Loss: 1.228 || Accuracy:  0.61 || Time: 12.48
Rank: 3 || Epoch: 18 || Loss: 1.234 || Accuracy:  0.61 || Time: 12.48
Rank: 2 || Epoch: 18 || Loss: 1.254 || Accuracy:  0.60 || Time: 12.48
Rank: 2 || Epoch: 19 || Loss: 1.210 || Accuracy:  0.62 || Time: 12.59
Number of available GPUs: 4
249.65013790130615
Rank: 3 || Epoch: 19 || Loss: 1.218 || Accuracy:  0.62 || Time: 12.59
Number of available GPUs: 4
249.65002608299255
Rank: 1 || Epoch: 19 || Loss: 1.225 || Accuracy:  0.62 || Time: 12.59
Number of available GPUs: 4
249.650949716568
Rank: 0 || Epoch: 19 || Loss: 1.229 || Accuracy:  0.61 || Time: 12.59
Number of available GPUs: 4
249.65137434005737
```

Batch size: 64

```
Rank: 2 || Epoch: 18 || Loss: 1.481 || Accuracy:  0.54 || Time: 11.97
Rank: 0 || Epoch: 19 || Loss: 1.481 || Accuracy:  0.52 || Time: 11.50
Number of available GPUs: 4
236.01985788345337
Rank: 3 || Epoch: 19 || Loss: 1.464 || Accuracy:  0.55 || Time: 11.50
Number of available GPUs: 4
236.0199897289276
Rank: 1 || Epoch: 19 || Loss: 1.448 || Accuracy:  0.56 || Time: 11.50
Number of available GPUs: 4
236.02083134651184
Rank: 2 || Epoch: 19 || Loss: 1.434 || Accuracy:  0.56 || Time: 11.50
Number of available GPUs: 4
236.02105331420898
```

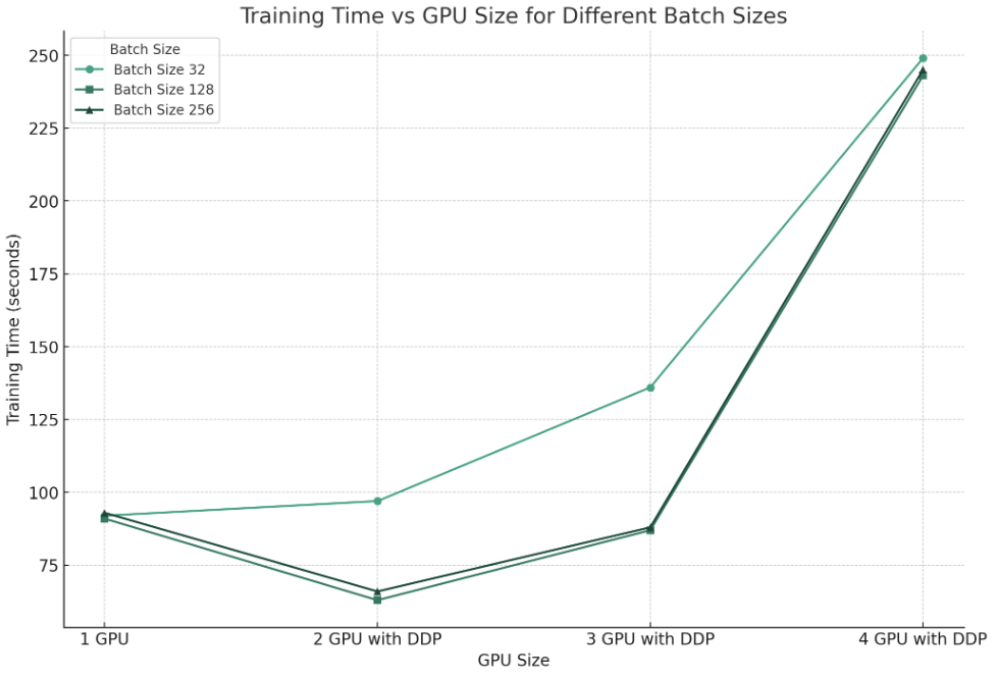
Batch size: 128

```
Rank: 1 || Epoch: 18 || Loss: 1.766 || Accuracy:  0.45 || Time: 12.46
Rank: 2 || Epoch: 18 || Loss: 1.737 || Accuracy:  0.48 || Time: 12.46
Rank: 0 || Epoch: 19 || Loss: 1.735 || Accuracy:  0.47 || Time: 11.95
Number of available GPUs: 4
245.20328736305237
Rank: 3 || Epoch: 19 || Loss: 1.714 || Accuracy:  0.48 || Time: 11.95
Number of available GPUs: 4
245.20352578163147
Rank: 1 || Epoch: 19 || Loss: 1.767 || Accuracy:  0.46 || Time: 11.95
Number of available GPUs: 4
245.20385456085205
Rank: 2 || Epoch: 19 || Loss: 1.749 || Accuracy:  0.46 || Time: 11.95
Number of available GPUs: 4
245.20393633842468
```

Batch size: 256

```
Rank: 0 || Epoch: 17 || Loss: 1.768 || Accuracy:  0.45 || Time: 12.41
Rank: 1 || Epoch: 17 || Loss: 1.742 || Accuracy:  0.47 || Time: 12.41
Rank: 2 || Epoch: 17 || Loss: 1.755 || Accuracy:  0.47 || Time: 12.41
Rank: 3 || Epoch: 18 || Loss: 1.740 || Accuracy:  0.46 || Time: 12.46
Rank: 0 || Epoch: 18 || Loss: 1.744 || Accuracy:  0.46 || Time: 12.46
Rank: 1 || Epoch: 18 || Loss: 1.766 || Accuracy:  0.45 || Time: 12.46
Rank: 2 || Epoch: 18 || Loss: 1.737 || Accuracy:  0.48 || Time: 12.46
Rank: 0 || Epoch: 19 || Loss: 1.735 || Accuracy:  0.47 || Time: 11.95
Number of available GPUs: 4
245.20328736305237
Rank: 3 || Epoch: 19 || Loss: 1.714 || Accuracy:  0.48 || Time: 11.95
Number of available GPUs: 4
245.20352578163147
Rank: 1 || Epoch: 19 || Loss: 1.767 || Accuracy:  0.46 || Time: 11.95
Number of available GPUs: 4
245.20385456085205
Rank: 2 || Epoch: 19 || Loss: 1.749 || Accuracy:  0.46 || Time: 11.95
Number of available GPUs: 4
245.20393633842468
```

Performance for different batch sizes on different GPUs



Batch size	32	64	128	256
1 GPU	92s	94s	91s	93s
2 GPU with DDP	97s	79s	63s	66s
3 GPU with DDP	136s	101s	87s	88s
4 GPU with DDP	249s	236s	243s	245s

Performance Results:

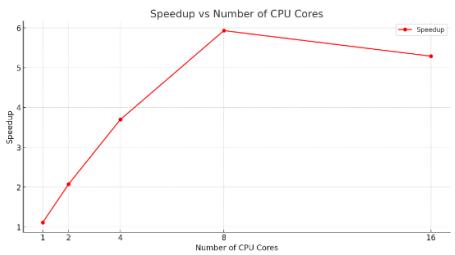
The results section is expanded to include detailed analysis of the processing times recorded during the exploratory data analysis (EDA) and data loading phases.

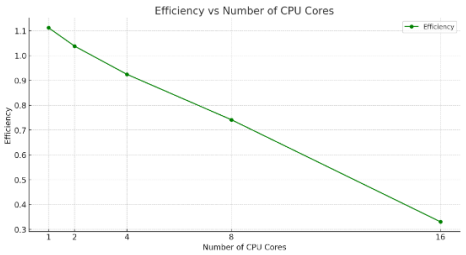
Sequential vs. Parallel Processing

The report presents a comprehensive comparison between sequential and parallel processing. The findings indicate that sequential execution is markedly slower than its parallel counterparts, with a significant increase in speed observed even when moving from one to two CPU cores in multiprocessing [1].

Impact of CPU Core Utilization

As the number of cores increases, the processing time decreases, but the efficiency tends to drop when the number of processes exceeds the number of available CPU cores. This phenomenon is attributed to context switching and overheads, which becomes particularly evident when the CPU count is doubled from the actual number of cores.





Dask's Computational Model

Dask's performance is noted to be superior to sequential execution and competitive with multiprocessing in certain scenarios. However, its overhead in task management suggests that Dask's strengths lie in its ability to handle larger, more complex tasks that can benefit from its advanced scheduling rather than simple, small-scale operations.

Batch size vs training time

the relationship between batch size and training time is not directly proportional when a single GPU is used for processing. As the batch size increases, the training time rises only slightly, suggesting that larger batches are being processed more efficiently to a certain extent. This efficiency gain is attributed to the GPU's ability to leverage its parallel processing capabilities more effectively when handling larger batches of data.

Increasing GPUs

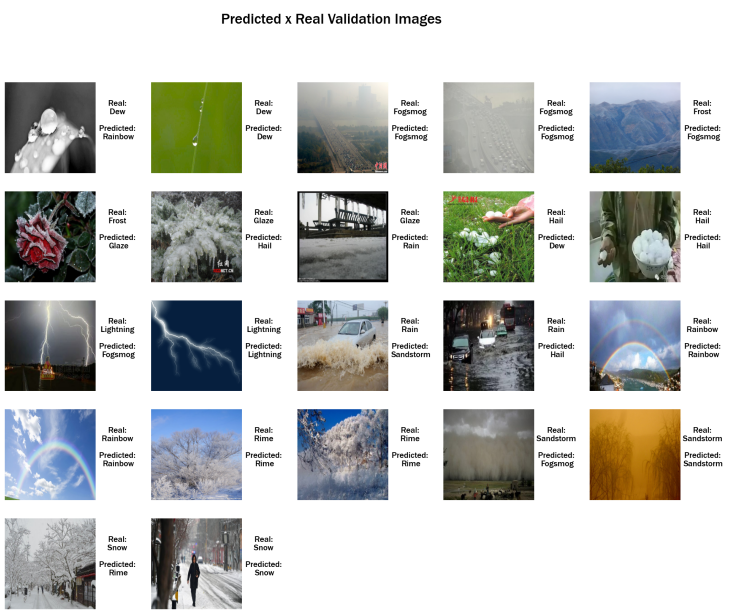
When scaling up the number of GPUs, the impact varies depending on the batch size. With smaller batches (64 and 128), increasing from one to two GPUs initially results in increased training times. It implies that the parallelization overhead, particularly the cost of communication between GPUs, may be more significant than the speedup gained from parallel processing when dealing with less data.

However, for larger batch sizes (256 and 512), the training times decrease when the GPU count is increased from one to two. This decrease suggests that, for substantial amounts of data, the benefits of parallel processing with two GPUs outweigh the inter-GPU communication costs, leading to more effective parallelization and improved training efficiency.

The trend reverses when the number of GPUs is further increased to four. The training times go up for all batch sizes, which could point to a scenario where the scaling is not efficient anymore. The additional overhead associated with synchronization and communication across more GPUs becomes substantial, detracting from the computational benefits and possibly leading to inefficiencies.

The optimal point for this particular training setup appears to be the use of two GPUs, especially for the larger batch sizes of 256 and 512. This configuration likely offers the most balanced trade-off between the efficiency of parallel computation and the overhead incurred from communication between GPUs, resulting in the lowest training times observed in the analysis.

Classification Result: The test accuracy of the model is 55% and the performance on the new dataset is shown below



Conclusion:

Standard Python (Sequential Execution): By using only a single core for computation, this approach serves as a baseline for performance comparison, demonstrating the least efficiency and longest execution time.

Multiprocessing with 2 CPUs: Doubling the computing resources cuts the execution time significantly, validating the benefits of parallel processing even with a modest increase in resources.

Multiprocessing with 4 CPUs: This approach leverages four cores to nearly quadruple the performance over the sequential execution, indicating an almost linear scale-up in speed as additional cores are engaged.

Multiprocessing with 8 CPUs: Engaging all available cores results in the maximum performance gain, showcasing the effectiveness of fully utilizing the hardware for parallelizable tasks.

Multiprocessing with 16 CPUs: Introducing more processes than available cores does not yield further improvements; instead, it incurs additional overhead due to context switching, which outweighs the benefits of additional parallelism, leading to decreased performance.

Dask: Although not always superior to multiprocessing in simpler tasks, Dask provides advanced task scheduling and can handle larger, more complex computations efficiently, especially when dealing with out-of-memory data or tasks that can benefit from lazy evaluation and other optimizations.

GPUs: This optimal configuration is crucial for enhancing training efficiency in complex deep learning tasks such as weather classification from images, where managing computational resources effectively is paramount for timely and accurate model development.

Efficiency Trade-off: Data parallelism across multiple GPUs does not always showcase superior performance compared to single GPUs due to overhead.

Optimized Computation Time: Dask and multiprocessing exhibit significant improvements in execution time compared to serial execution, with Dask Array notably enhancing NumPy Array computation.

Strategic GPU Utilization: Optimal performance is achieved by leveraging Dask and multiprocessing on a configuration with 4 GPUs, balancing efficiency and computational speed.

Model-Specific Parallelism Advantage**:** Model parallelism demonstrates a distinct advantage on multiple GPUs compared to multiple CPUs, showcasing enhanced speed and scalability for our specific model architecture.

Reference:

1. https://docs.python.org/3/library/multiprocessing.html
2. https://docs.dask.org/en/stable/
3. https://pytorch.org/tutorials/intermediate/ddp_tutorial.html
4. https://arxiv.org/pdf/2212.07936.pdf
5. https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2020EA001604
6. https://pytorch.org/docs/stable/index.html