TECHNICAL REPORT

Performance Analysis of Distributed Training Frameworks

Sahil Sunil Amundkar, Nivetha Murugaiyan Kumar, Swetha Murugaiyan Kumar

PROBLEM STATEMENT

With the rise of complex deep learning models, distributed learning frameworks are crucial for efficient training on multiple machines. However, choosing the right framework requires understanding their strengths and weaknesses. This project tackles this challenge by comparing three popular options: Uber Horovod, PyTorch, and Microsoft Deepspeed. We'll evaluate factors like setup difficulty, performance across different scales, user-friendliness, compatibility with deep learning libraries, how they handle data updates across machines, and built-in optimizations. This comprehensive analysis aims to provide valuable guidance for researchers and practitioners working in distributed machine learning.

LITERATURE REVIEW

Uber Horovod:

- Horovod is an open-source library built by Uber that simplifies distributed deep learning training across multiple machines or GPUs.
- It supports various deep learning frameworks, including PyTorch, TensorFlow, Keras, and Apache MXNet.
- Horovod's key advantage lies in its ease of use. With minimal code changes to existing PyTorch training scripts, Horovod can scale training to hundreds of GPUs, significantly reducing training time.
- It focuses on making distributed training accessible and leverages message passing interface (MPI) for communication between machines.

PyTorch:

- PyTorch is a popular open-source deep learning framework known for its flexibility, ease of use, and dynamic computational graphs.
- It can be used for various deep learning tasks like computer vision, natural language processing, and recommender systems.
- While PyTorch offers some distributed training capabilities, it might not be the most efficient for large-scale training on multiple GPUs or machines.

Microsoft DeepSpeed:

- DeepSpeed is a library developed by Microsoft specifically for PyTorch.
- Similar to Horovod, it aims to accelerate distributed training, but with a focus on performance optimization within PyTorch itself.

- DeepSpeed offers features like gradient accumulation, zero redundancy optimization, and mixed-precision training to improve training speed and memory usage on large models.
- While DeepSpeed might require slightly more code modifications compared to Horovod, it can potentially achieve better performance on PyTorch models.

METHODOLOGY

1. Set up Horovod for distributed training on Tensorflow, Keras, Pytorch and Elastic to evaluate MNIST and CIFAR-10.
2. Set up Deepspeed for distributing training for Custom Mixed Precision Training Handling and ZeRo optimization.
3. Setup pytorch for distributed training using model parallelism.

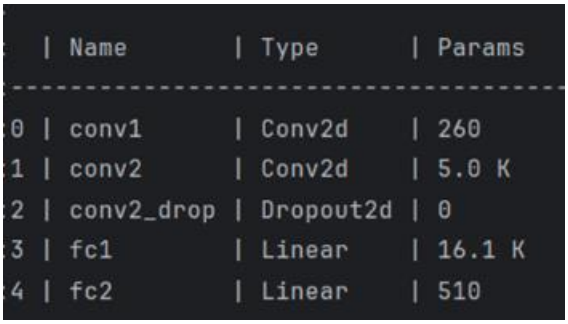
Our objective is to understand the working of each distributed training framework, comprehensively evaluate them on standard datasets and compare the results on benchmarked parameters.

EXPERIMENTAL SETUP

GPU Configuration: NVIDIA Titan Xp

a. Compute capability → 6.1
b. Video Memory Type → GDDR5
c. Memory → 12 GB
d. Bus Interface → PCIe 3.0 x 16

Convolutional layer architecture: 21.8K trainable parameters



	Name	Type	Params
0	conv1	Conv2d	260
1	conv2	Conv2d	5.0 K
2	conv2_drop	Dropout2d	0
3	fc1	Linear	16.1 K
4	fc2	Linear	510

Benchmarking standards → Batch size: 128, Epochs: 30

Datasets Used

MNIST:

- **Focus:** Handwritten digits classification (0-9)
- **Type:** Grayscale images (single channel)
- **Image size:** 28x28 pixels
- **Number of classes:** 10 (one for each digit)

- **Complexity:** Relatively simple due to low resolution, grayscale format, and limited number of classes. A good starting point for beginners in image classification.

CIFAR-10:

- **Focus:** Object recognition of everyday objects (airplanes, cars, etc.)
- **Type:** Color images (3 channels - RGB)
- **Image size:** 32x32 pixels
- **Number of classes:** 10 (e.g., airplanes, automobiles, birds, etc.)
- **Complexity:** More complex than MNIST due to color information and depicting real-world objects with variations. Often used as a stepping stone to more complex datasets.

EVALUATION RESULTS

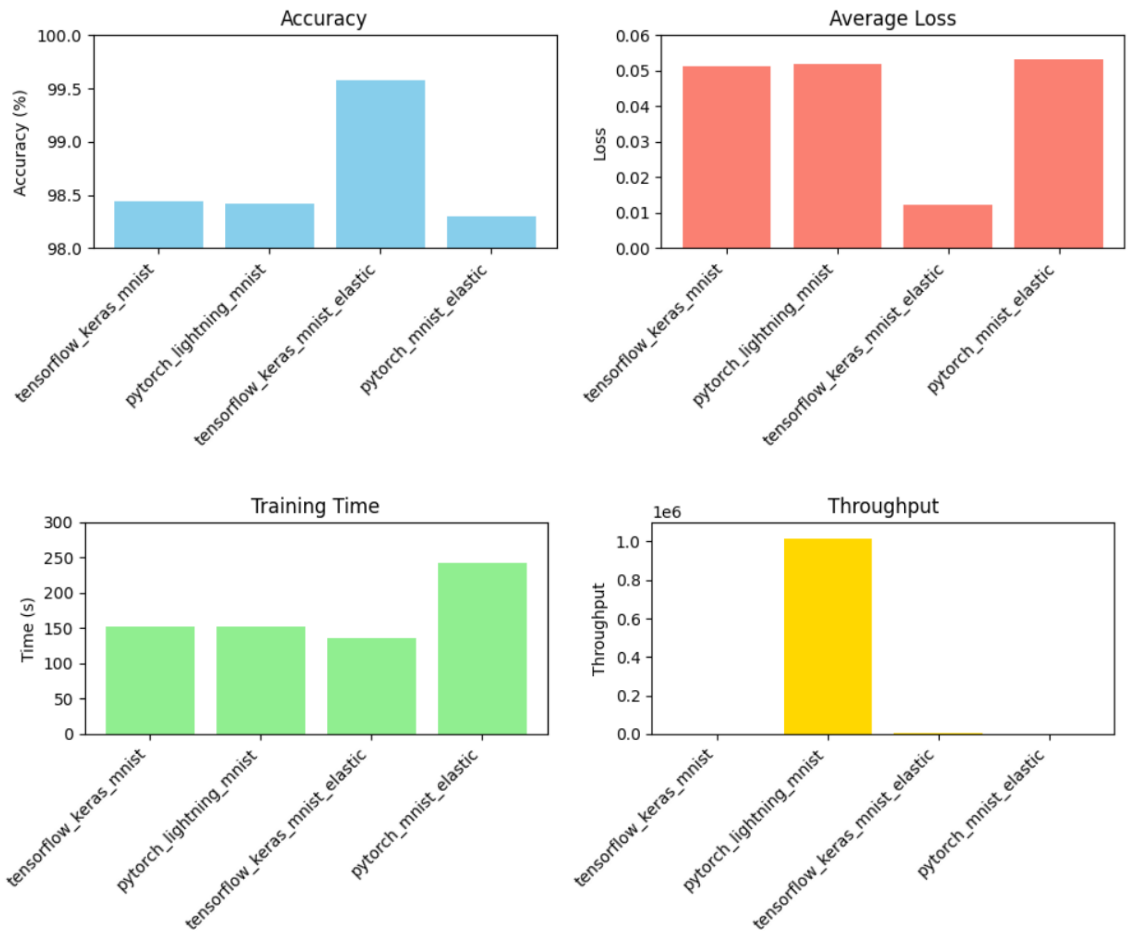
Uber Horovod

1. MNIST

	GPU memory-usage	Volatile GPU-util
Tensorflow2_keras_mnist	937 MiB/12288 MiB	2%
Pytorch_lightning_mnist	1131MiB/12288 MiB	0%
Tensorflow2_keras_mnist_elastic	1655 MiB/12288 MiB	30%
Pytorch_mnist_elastic	915 MiB/12288 MiB	4%

2. CIFAR-10

	GPU memory-usage	Volatile GPU-util
Tensorflow2_keras_cifar10	1639 MiB/12288 MiB	39%
Pytorch_lightning_cifar10	1631 MiB/12288 MiB	11%
Tensorflow2_keras_cifar10_elastic	2679 MiB/12288 MiB	30%
Pytorch_cifar10_elastic	1655 MiB/12288 MiB	7%



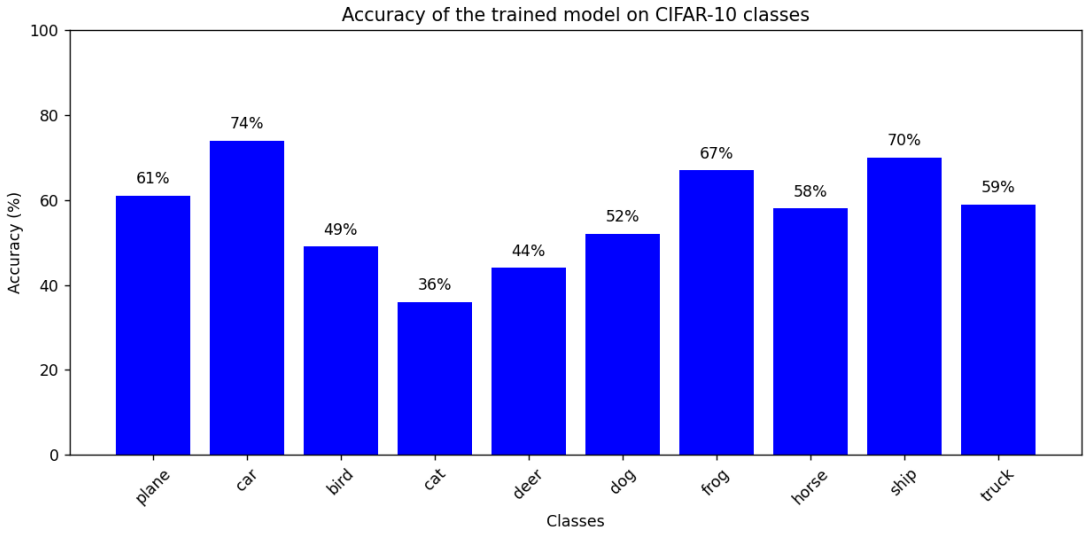
Microsoft Deepspeed

Internally, Deepspeed operates on the following principles.

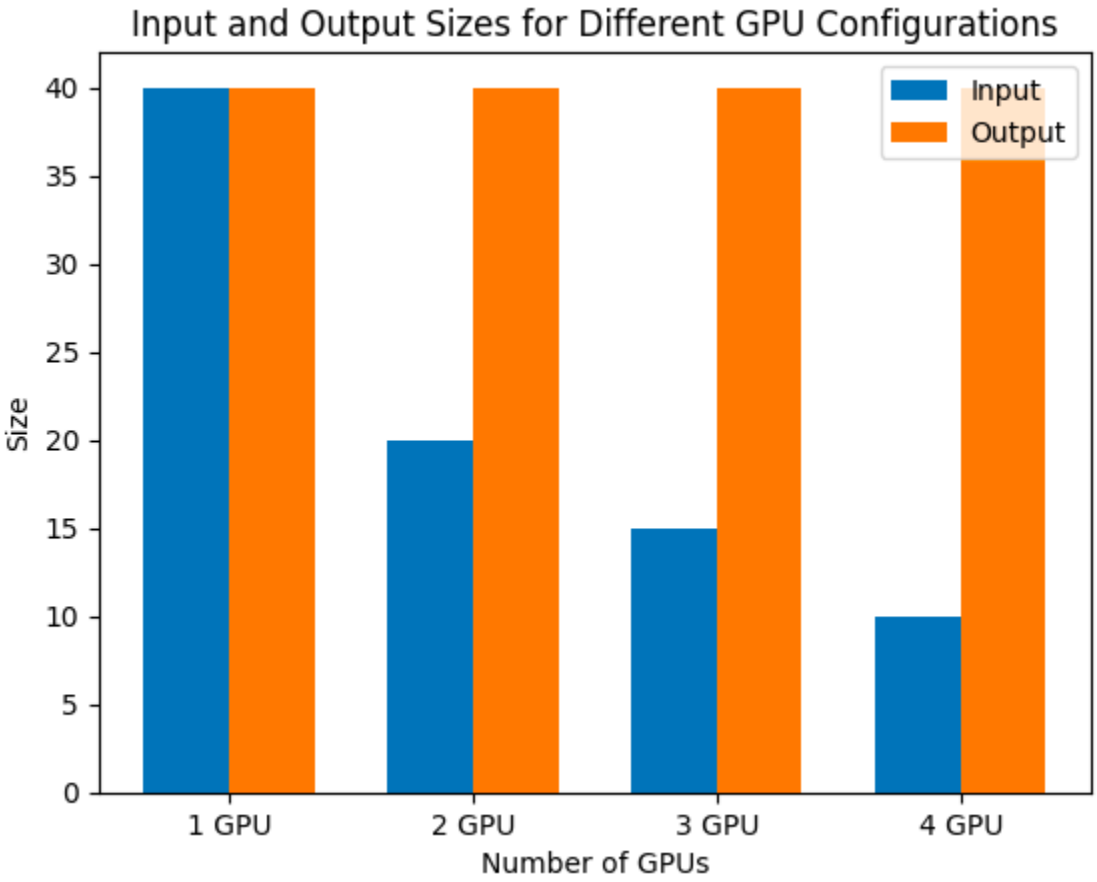
- **Gradient Averaging**: in distributed data parallel training, backward ensures that gradients are averaged across data parallel processes after training on a train_batch_size.
- **Loss Scaling**: in FP16/mixed precision training, the DeepSpeed engine automatically handles scaling the loss to avoid precision loss in the gradients.
- **Learning Rate Scheduler**: when using a DeepSpeed's learning rate scheduler (specified in the ds_config.json file), DeepSpeed calls the step() method of the scheduler at every training step (when model_engine.step() is executed). When not using DeepSpeed's learning rate scheduler:

```
ds_config = {
    "train_batch_size": 3,
    "steps_per_print": 2000,
    "optimizer": {
        "type": "Adam",
        "params": {
            "lr": 0.001,
            "betas": [0.8, 0.999],
            "eps": 1e-8,
            "weight_decay": 3e-7,
        },
    },
"gradient_clipping": 1.0,
"prescale_gradients": False,
"bf16": {"enabled": args.dtype == "bf16"},
"fp16": {
    "enabled": args.dtype == "fp16",
    "fp16_master_weights_and_grads": False,
    "loss_scale": 0,
    "loss_scale_window": 1000,
    "hysteresis": 3,
    "min_loss_scale": 1,
    "initial_scale_power": 16,
},
```

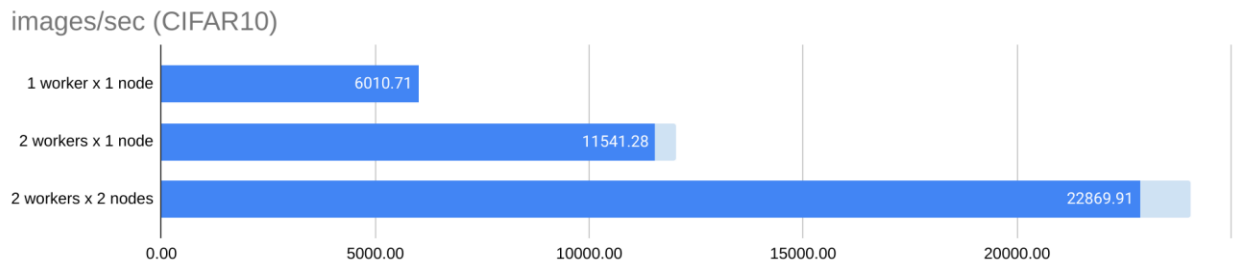
The accuracy of the network on the test set is 57%. The accuracy varies for different classes, with the highest accuracy observed for the "car" class (74%) and the lowest for the "cat" class (36%).



Pytorch Distributed



Regardless of the GPU configuration, PyTorch distributed training demonstrated consistent output sizes despite variations in batch processing. The results suggest that PyTorch distributed training is adaptable to various CNN architectures and can effectively handle the complexities of training convolutional models, however, not as effectively as Deepspeed (for CIFAR-10).



The blue bar shows the training throughput almost increases linearly with the number of workers from 1 to 4 workers (across two nodes).

CONCLUSION

Horovod – MNIST

- Elastic training improves performance: Both TensorFlow Keras and PyTorch frameworks achieved better accuracy and training time with their elastic versions, suggesting that dynamically adjusting resources during training can be beneficial.
- TensorFlow Keras excels in accuracy: The elastic TensorFlow Keras model achieved the highest accuracy, highlighting the effectiveness of this approach for this task.
- PyTorch Lightning leads in throughput**:** While not achieving the highest accuracy, PyTorch Lightning's model demonstrated the best throughput, indicating its efficiency in processing training data. Further optimizations might be needed to improve its accuracy while maintaining its speed.

Horovod – CIFAR10

- PyTorch Lightning achieved the best results with the highest accuracy and lowest loss, suggesting its potential effectiveness when employing Horovod for distributed training on larger datasets that could benefit from it.
- Interestingly, elastic training didn't significantly improve TensorFlow Keras or PyTorch models for CIFAR-10, unlike the MNIST case. This implies that the elastic approach might be more suitable for smaller datasets like MNIST, while CIFAR-10 may not require such dynamic resource adjustments.
- PyTorch models consistently demonstrated higher throughput compared to TensorFlow Keras, indicating potential advantages in processing CIFAR-10 data. This efficiency aspect

could be further amplified when leveraging Horovod for distributed training on larger datasets.

Deepspeed – CIFAR10

- Efficient Training: DeepSpeed effectively manages training on CIFAR-10.
- Decent Accuracy: The model achieves a 57% overall accuracy on the test set.
- Class Variability: Performance varies across classes; "car" and "ship" show higher accuracies, while "cat" and "bird" perform relatively poorly.
- Optimization Techniques: DeepSpeed employs optimization strategies like gradient accumulation and learning rate scheduling.
- Room for Improvement: Further analysis, including examining misclassifications and experimenting with architecture and hyperparameters, could enhance model performance.
- Smooth Execution: Training progresses without errors, indicating DeepSpeed's robustness.

Despite this, DeepSpeed demonstrates efficiency in managing the training process, offering a solid foundation for enhancing model performance on the CIFAR-10 dataset.

Pytorch Distributed - CIFAR10

- With 1 GPU, the model processes batches internally, resulting in smaller batch-wise outputs, while collectively handling larger datasets outside the GPU boundary, maintaining consistency in output size.
- Transitioning to 2 GPUs enables parallel processing of larger batch sizes within each GPU, yet the output size remains unaffected, showcasing efficient data distribution across multiple GPUs.
- Regardless of GPU configuration, the model efficiently manages input data, ensuring consistent output sizes despite variations in batch processing.

Overall Comparison

Feature	Horovod	DeepSpeed	PyTorch Distributed
Ease of Use	Easiest	Moderate	Moderate
Performance (Large)	Good	Excellent	Good
Features	Limited	Extensive	Moderate
Compatibility	Multiple libraries	PyTorch only	PyTorch only
Data Updates	Allreduce	Allreduce, ZeRO	Allreduce (default)
Optimizations	Basic	Advanced (ZeRO)	Moderate

1. Horovod scaled well for common models on multiple GPUs.
2. DeepSpeed excelled for huge models (trillions of parameters) with memory optimisations.
3. PyTorch scaled well, but DeepSpeed offers additional techniques for extreme scale.

Our results demonstrate that smaller batch sizes, lighter models and a larger number of nodes will require faster GPU-to-GPU communication for distributed training to become efficient. Overall, Horovod, Deepspeed and Pytorch are excellent platforms for distributed training and are distinguishable only by their applications. Deepspeed is suitable for larger models across multiple nodes, Horovod is malleable, and handles both lighter and larger models effectively while Pytorch is lightweight and has an easy-to-access API.

FUTURE WORKS

- Bigger models, bigger datasets: Move from MNIST/CIFAR-10 to massive datasets like ImageNet to see how frameworks handle real-world complexity.
- Scale it up: Don't limit testing to single machines. Evaluate how frameworks perform across multiple machines/GPUs for large-scale training.
- Explore more options: Include new frameworks like Transformers or MXNet to get a broader picture of the distributed training landscape.

REFERENCES

1.Sergeev, Alexander, and Mike Del Balso. "Horovod: fast and easy distributed deep learning in TensorFlow." arXiv preprint arXiv:1802.05799 (2018).

2.Rasley, Jeff, et al. "Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters." Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020.

3.Li, Shen, et al. "Pytorch distributed: Experiences on accelerating data parallel training." arXiv preprint arXiv:2006.15704 (2020).