Benchmarking image classification models using parallelisation techniques

Abhinav Gorantla Niranjan E N S Sudhanva Moudgalya

{*agorant2, nerappan, srmoudga*}*@asu.edu*

School of Computing and Augmented Intelligence

Arizona State University

1 Problem Formulation

As the utilization of computer vision technologies—such as autonomous vehicles—continues to grow, the importance of image classification correspondingly increases. In certain applications of computer vision that rely on image classification, the performance of the model is critically important. For instance, autonomous vehicles require models that deliver both rapid and precise results to facilitate immediate decision-making. Similarly, other applications in computer vision demand high accuracy and swift performance from these models due to their time-sensitive nature.

In this research, we aim to benchmark two prominent image classification models: RESNET50 and Vision Transformers. Our training will involve the CIFAR10 dataset, utilizing different methodologies within PyTorch, including eager execution, Horovod, and Gpipe. Each model will undergo training three times using these varied approaches. We will assess and compare the performance of these models, focusing on their accuracy and training efficiency. During the Gpipe implementations, the models will be tested across various GPU configurations, incorporating a range of A30 and A100 GPUs.

The outcomes of this project will facilitate a deeper understanding of the benefits of employing model and data parallelization techniques, particularly in the training of large-scale models or when handling extensive datasets. This insight is crucial for enhancing both the efficiency and effectiveness of model training processes.

This study is interesting because the study's focus on comparing various parallelization techniques and hardware configurations offers valuable data on optimizing computational resources, which is crucial for advancing current machine learning infrastructure and practices. The fact that we run our experiments on latest image classification models like SWIN Transformer make this study very compelling as it ensures we are leveraging state-of-the-art techniques in image classification .

2 Literature Survey

RESNET50 is an image classification deep neural network model. It was introduced by [1] to solve the problem of training error degradation as we increase the number of layers in a deep neural network. RESNET solves this problem by introducing a deep residual network framework. In [1], the authors use a plain network which uses an architecture similar to VGG-19[2]. But the plain network used in [1] architecture has much lower FLOPs than VGG-19 (almost 18% lower). The RESNET architecture, based on the plain baseline networks use shortcut connections to put ”residual learning” into the picture. To compare the performance of the RESNET architecture with the baseline plain network, the authors use the CIFAR10 dataset. In both training and testing scenarios, the network with residual connections has almost 10% lower classification error rate.

RESNET has multiple architectures with varying number of layers, we plan to use the RESNET50 architecture in our project. The reason behind this decision is the performance gap between the various RESNET architectures from RESNET-34 to RESNET-50, there is almost a 3% decrease in top-1 error, but from RESNET-50 to RESNET-152, there is only 1% decrease in the error value for 10 crop testing on ImageNet validation data.

In this paper, we have selected RESNET and Vision Transformer (ViT) as the primary models for the comparative analysis of parallelization techniques due to several strategic and insightful reasons. Firstly, RESNET and ViT offer a diverse range of architectures, with RESNET embodying convolutional neural networks (CNNs) and ViT representing a newer class of models based on transformer architectures. This diversity enables a thorough evaluation of parallelization techniques across fundamentally distinct computational frameworks. Secondly, both models have demonstrated state-of-the-art performance in various vision applications. Analyzing parallelization techniques using these high-performance models allows us to glean insights into scalability and efficiency improvements that can be achieved across different model architectures. Lastly, the distinct computational characteristics of RESNET and ViT make them ideal candidates for this study. RESNET primarily utilizes convolutional operations, which, while well-optimized on current hardware, can still significantly benefit from advanced parallelization techniques to manage increasingly deep architectures. Conversely, ViT, which operates on an attention-based mechanism, demands extensive memory and computational resources, particularly due to the self-attention mechanism that processes image patches. This necessitates an exploration of how different parallelization strategies can optimize performance and resource utilization in such models.

Drawing from the success of the Transformer model in natural language processing, a similar idea was introduced for 2D image data called Vision Transformers(ViT) [3]. ViT adapts the self-attention mechanism to handle 2D image data. Unlike CNNs which process the image through localized filters, ViT divides an image into a sequence of patches and treats these patches as tokens similar to words in a sentence. In detail, ViT architecture splits an images into fixed-size patches, these patches are then flattened and transformed into embeddings, which is then added with positional embeddings to retain positional information. The encoder consists of alternating layers of multi-head self-attention and MLP blocks, with each layer followed by normalization and equipped with residual connections. The output of the encoder goes through a classification head, typically a single linear layer, to predict the image label. This architecture leverages the power of self-attention to focus on relevant parts of the image, regardless of their spatial location, making it inherently more flexible than CNNs.

We plan to use the CIFAR-10 dataset to benchmark the image classification models in this study. CIFAR-10 [4] consists of 60,000 32x32 color images distributed evenly across 10 classes, with each class containing 6,000 images. CIFAR10 is known for its manageable size and complexity, unlike large datasets such as ImageNet which contains millions of images across thousands of categories, CIFAR10's modest size gives way to rapid experimentation and model development. The other key advantage of using CIFAR10 is the relatively low resolution of images which furthur reduces computational requirements.

Krishevsky et. al [4] introduced CIFAR10 as part of the experimentation of CNNs for image classification, and then their work on AlexNet achieved state-of-the-art results on CIFAR10. Since then, CIFAR10 has been used for been used to evaluate performance of many other architectures such as VGG, ResNet and DenseNet. Its accessibility, format and diverse class distribution makes it an ideal choice to evaluate image classification on different parallelization techniques.

One of the model parallelism framework we plan to evaluate in this study is Gpipe. Gpipe [5] is a framework introduced by the engineers at Google for paralellizing deep learning workflows using a pipeline based approach. Gpipe can be applied on any deep learning model in which the model can be divided into multiple layers. Both RESNET and swinTransformer can be divided into multiple layers, hence, Gpipe can be used to parallelize training workflows in both of these models. Gpipe divides the deep network into smaller sub-networks or stages and each stage is given the job of processing a prt of the input data or the gradient computation task. Each of these stages are then sequentially trained on separate cores, forming a pipeline. This allows for the parallelization.

Dividing network into multiple smaller stages allows for parallelism and also efficient use of available parallel power. This becomes very important when the size of datasets and models increase. The impact in the training time when using Gpipe with deep learning models is explored in [6]. [6] concluded that recompute strategy used by gpipe will negatively impact the training time of the DNN models. We plan to verify this claim in our study.

The other parallelism we plan to use in this study is Horovod. Horovod [7] was introduced by the engineers at Uber. It uses the Ring-AllReduce algorithm to distribute the gradient and parameter updates across the network, enabling efficient communication and synchronization between nodes during training. This approach helps to significantly reduce the training time of complex deep learning models by leveraging the computational power of multiple GPUs and nodes without requiring substantial changes to the existing model code.

Furthermore, Horovod supports advanced features such as tensor fusion, which groups small communication operations together into larger ones to improve the efficiency of network operations, and gradient compression, which reduces the amount of data that needs to be transferred between nodes.

In conclusion, Horovod is an effective and adaptable distributed training framework that aids developers in effectively scaling their deep learning models across numerous GPUs and nodes. Its performance optimization features and connection with well-known deep learning frameworks make it an invaluable tool for speeding up the training of complicated models in a distributed computing environment. We plan to evaluate this method of model parallelism with the one offered by Gpipe and check which of the two performs better.

3 Proposed Method

Our proposed method includes evaluating image classification algorithms on their train time when using model paralellization / data paralellization as compared to a plain model implementation on pytorch. For the model paralellization technique we have decided to go with GPipe and for the data parallelization method, we have decided to go with Horovod. Both of these parallelization methods have a pip package which make them easy to use.

For the image classification models, we decided to go with RESNET50 and SWIN Transformer. We decided to go with these two because RESNET50 is one of the standard models in the image classification application area. It is a convolutional neural network which has 50 layers. In a CNN, the image is input as a 2D (or 3D) matrix. To this matrix representation of the image, filters are applied to the input image at each stage and these filters are also applied at different resolutions. For example, earlier in the network, the filters are applied to extract higher level features and as we go deeper into the network, we focus on the finer features.

The second image classification model we used for our analysis was the SWIN model [6]. SWIN is a model based on the transformer architecture. It is known to speed up while also improving accuracy for image classification tasks. In addition to image classification, SWIN Transformers can also perform image segmentation tasks and is currently one of the state of the art models for image classification and segmentation tasks.

We ran all our experiments on ASU's Sol supercomputer using its A100 and A30 super computers. We go into the details of the exact setup we are using for running the experiments and for the baseline in the following section.

4 Experimental Setup

We conducted our experiments using the ASU Sol supercomputer, employing two distinct hardware configurations, each equipped with two NVIDIA GPUs: the A30 and the A100 models. Both configurations were supported by 1 CPU cores and 8 GB of RAM. The software environment was standardized across setups, consisting of NVIDIA CUDA version 12.1, Python 3.11.8, and an array of Python libraries designed for deep learning applications. This included NumPy 1.26.4, PyTorch 2.1.2, torchaudio 2.1.2, torchvision 0.16.2, torchgpipe 0.0.7, and Horovod 0.28.0.

For the baseline experiments involving ResNet and SWIN, we utilized the A100 GPU on the ASU Sol supercomputer. To explore the effectiveness of Gpipe, we established two distinct setups: one configured with two A30 GPUs, and the other with two A100 GPUs. We tested two balancing strategies for Gpipe; in the first, both segments of the image classification model were run on a single GPU with a balance configuration of [2], while in the second setup, each segment was allocated to a separate GPU with a balance of [1, 1].

For the experiments involving Horovod, we utilized a dual A100 GPU setup to train and test the models. This configuration was chosen to assess the scalability and efficiency of distributed training under substantial computational load.

5 Dataset Description

The CIFAR-10 dataset comprises of 60,000 color images distributed evenly across 10 classes — airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck—with each class containing 6,000 32x32 pixel images. It is extensively utilized within the machine learning community, particularly valued for its balance of complexity and manageability. This dataset is segmented into a training set of 50,000 images and a testing set of 10,000 images, ensuring equitable representation of each class in both subsets [4].

CIFAR-10's origins from the 80 million tiny images dataset and its introduction by [4]. present it as a dataset designed to challenge object recognition algorithms with a variety of image orientations, lighting conditions, and background scenarios. Despite the small size of the images, the dataset effectively simulates real-world visual diversity, making it a robust tool for testing image classification models.

For our project, CIFAR-10's modest image size and comprehensive class distribution are advantageous, enabling rapid iteration and development of models. This makes it ideal for evaluating the performance impacts of parallelization techniques on image classification algorithms. The dataset's reduced computational demand compared to larger sets like ImageNet allows for more frequent and faster training cycles. This facilitates a focused examination of how model and data parallelization techniques, such as those implemented with Gpipe and Horovod, influence training efficiency and model accuracy.

In essence, CIFAR-10 serves as an optimal platform for this project by allowing us to concentrate on the parallelization impacts without the extensive computational overhead associated with larger datasets.

6 Results

	RESNET50 (A100)	SWIN (A100)
Accuracy	83.53%	91.2%
Train Time	56.4 minutes	120.6 minutes

Table 1: Base Experiment results

Table 1 shows the base performance results on an A100 GPU, where RESNET50 achieved 83.53% accuracy in 56.4 minutes, while the SWIN Transformer reached 91.2% accuracy in 120.6 minutes. Table 2 shows performance using GPipe on different hardware setups. For RESNET50, the A30 x2 GPUs yielded 83.41% accuracy in 87 minutes, and the A100 x2 GPUs resulted in 83.04% accuracy in 51 minutes. In comparison, the SWIN Transformer model on A30 x2 GPUs achieved 90.46% accuracy in 150.2 minutes, and on A100 x2 GPUs, it achieved 90.08% accuracy in 102.4 minutes. The GPipe framework generally reduced training time for the SWIN Transformer, especially on the A100 x2 setup.

	RESNET50 (A30 x 2)	RESNET50 (A100 x 2)	SWIN (A30 x 2)	SWIN (A100 x 2)
Accuracy	83.41%	83.04%	90.46%	90.08%
Train Time	87 minutes	51 minutes	150.2 minutes	102.4 minutes

Table 2: Performance when using Gpipe

	RESNET50 (A1000 x 2)	SWIN (A100 x 2)
Accuracy	83.04%	90.08%
Train Time	39.2 minutes	92.6 minutes

Table 3: Performance when using Horovod

Table 3 shows that the performance improvements are evident when using Horovod on two A100 GPUs. Here, RESNET50's accuracy slightly decreased to 83.04%, but the training time was significantly reduced to 39.2 minutes. The SWIN Transformer's accuracy remained high at 90.08%, and the training time was reduced to 92.6 minutes, indicating that using Horovod with dual A100 GPUs enhances training efficiency, particularly for the SWIN Transformer.

7 Conclusion

The results we obtained by implementing model and data paralellization were not something we expected. From the literature we surveyed, we expected to have a speed-up of 2 - 2.5 when using GPipe (model parallelization) and a speed up of 3 - 3.5 when using Horovod (data parallelization). But this was nowhere close to the results we got, in fact, when running using RESNET50 and SWIN using parallelization techniques (Horovod, GPipe), it took us slightly longer to train these models although the accutacy remained approximately the same across all the training methods (serial, model parallel and data parallel).

8 Future Work

In the future we plan to extend this study to include more parallelization techniques like DeepSpeed, more hardware configurations (more number of GPUs and also more GPU architectures). We also plan to run these different configurations on different sizes of datasets with varying image sizes (image resolution). We think that this will give us a better idea on how these models will scale on varying types of image data and GPU architectures.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020.

[4] A. Krizhevsky, "Learning multiple layers of features from tiny images," pp. 32–33, 2009.

[5] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and z. Chen, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[6] P. Zhang, B. Lee, and Y. Qiao, "Experimental evaluation of the performance of gpipe parallelism," *Future Generation Computer Systems*, vol. 147, pp. 107–118, 2023.

[7] A. Sergeev and M. D. Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *CoRR*, vol. abs/1802.05799, 2018.