Project Technical Report

Course: Data Intensive Systems for Machine Learning (CSE 598) - Spring 2024

Team Number: 30

Team Name: Autobots

Project Title: Comparative analysis of Velox, PyTorch, and TensorFlow on ResNet workload

Group Members:

S.No	Name	ASU ID	E-mail
1	Rutwik Krushna Chaudhari	1229479292	rchaud32@asu.edu
2	Pratik Rajesh Agrawal	1229664022	pagarw40@asu.edu
3	Ashish Ambadas Wale	1229324631	awale1@asu.edu

1. Problem

The investigation at hand revolves around a critical challenge in the realm of machine learning: the daunting task of selecting the most suitable framework for a given project. We're focusing our efforts on evaluating the performance of various frameworks using ResNet, a widely recognized convolutional neural network architecture prominent in computer vision tasks. The ubiquity of ResNet makes it an ideal benchmark, offering a representative workload to gauge the effectiveness of different frameworks.

We aim to tackle the challenge of framework selection by honing in on key metrics such as inference time and ease of use. We're particularly interested in exploring the capabilities of Velox, a framework renowned for optimizing data-intensive operations within a single host. This choice intrigues us because it holds the promise of significant advantages, especially in scenarios where computational resources are limited or where efficient hardware utilization is crucial.

Moreover, this investigation is not just about solving a current problem; it's about staying ahead of the curve in the ever-evolving landscape of machine learning. Regular evaluations are vital to keep pace with advancements and trends in the field, ensuring that practitioners have access to the most effective tools and methodologies for their projects. By conducting systematic comparisons of machine learning frameworks, we're contributing to ongoing discussions on best practices in model deployment and inference, thus addressing a pertinent challenge faced by professionals in the machine learning community.

2. Literature review

The burgeoning success of Deep Learning (DL) models owes much to the confluence of factors such as the availability of vast datasets, increasing computational resources, and the accessibility of deep learning frameworks, notably TensorFlow and PyTorch. These frameworks, while heavily optimized for NVIDIA GPUs, have sparked numerous performance characterization studies primarily focused on GPU-based Deep Neural Network (DNN) training.

In this context, Arpan Jain and authors present a paper investigating TensorFlow and PyTorch's comparative performance in training DNNs on single-GPU setups. This study aims to unravel the performance disparities between the two frameworks, offering valuable guidance to end-users navigating framework selection. Through detailed benchmarking experiments encompassing seven popular neural network architectures spanning computer vision, speech recognition, and natural language processing (NLP), the research delineates key factors influencing performance, thereby empowering users to optimize model implementation for enhanced efficiency.

In a parallel endeavor, Hulin Dai and authors' effort undertakes an in-depth performance characterization of state-of-the-art DNNs across various CPU architectures and NVIDIA GPUs. This study not only sheds light on the performance disparities between CPU and GPU-based training but also explores the nuanced relationship between CPU/system characteristics and DNN specifications. The findings highlight the efficacy of multi-process (MP) training, even on single-node setups, over the single-process (SP) approach, and underscore the intricate interplay between hardware configurations and DNN architectures.

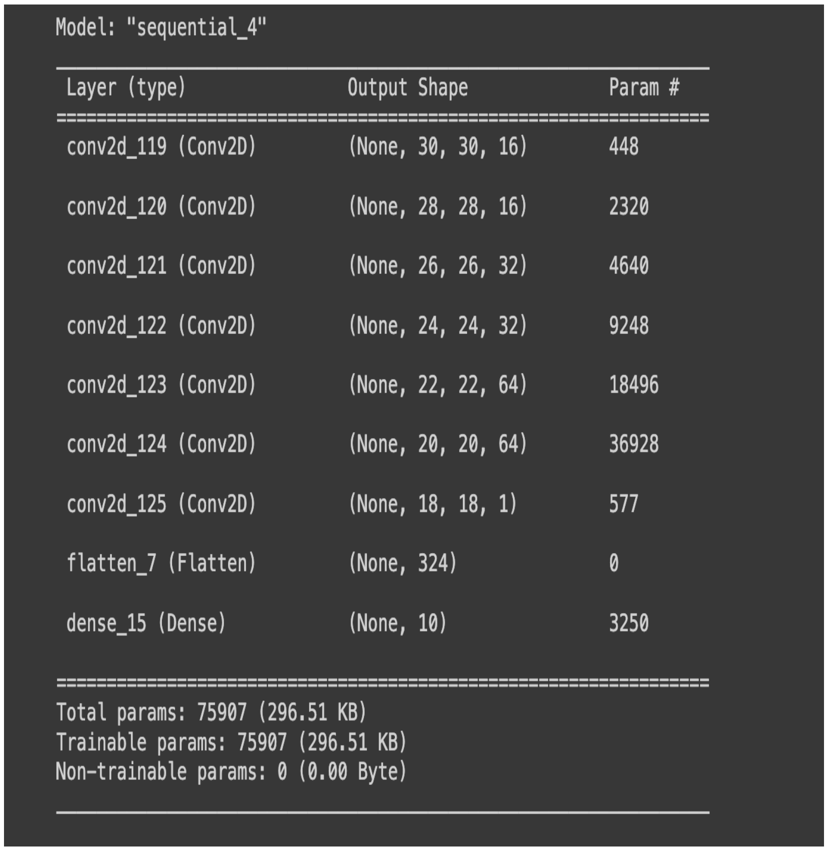
Furthermore, the comparative analysis between CPU and GPU-based training provides valuable insights into the relative advantages and limitations of each approach, facilitating informed decision-making for deep learning practitioners. Additionally, the profiling analysis for Horovod offers practical guidance for harnessing distributed computing paradigms effectively in deep learning workflows, thereby advancing the efficiency and efficacy of DL applications.

3. Model

Due to Velox's complexity, we describe our model as close to ResNet as possible. The layered architecture is described below.

There are a total of 9 layers. The first 7 convolutional layers perform the 2D convolution on the input image. The flattened layer is needed for TensorFlow before a final dense layer with the number of nodes equal to the number of prediction classes.

The architecture is missing the residual connections and Batch normalization layers from ResNet architecture.



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_119 (Conv2D)         (None, 30, 30, 16)        448

 conv2d_120 (Conv2D)         (None, 28, 28, 16)        2320

 conv2d_121 (Conv2D)         (None, 26, 26, 32)        4640

 conv2d_122 (Conv2D)         (None, 24, 24, 32)        9248

 conv2d_123 (Conv2D)         (None, 22, 22, 64)        18496

 conv2d_124 (Conv2D)         (None, 20, 20, 64)        36928

 conv2d_125 (Conv2D)         (None, 18, 18, 1)         577

 flatten_7 (Flatten)         (None, 324)               0

 dense_15 (Dense)            (None, 10)                3250

=================================================================
Total params: 75907 (296.51 KB)
Trainable params: 75907 (296.51 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

4. Experiment Setup

The experimental setup for our study involved conducting model training and testing on Google Cloud Platform (GCP) Compute instances, utilizing a specific configuration tailored to our requirements. Here's a breakdown of the key components of the setup:

RAM: The virtual machine instances were provisioned with 32 GB of RAM. Sufficient memory allocation is essential for handling large datasets and model parameters effectively, ensuring smooth execution of training and inference tasks without running into memory constraints.
Architecture: The architecture of the compute instances was x86_64, which is a common architecture for modern processors and compatible with a wide range of software and frameworks commonly used in machine learning tasks.
Operating System (OS): The compute instances were running on the Linux operating system. Linux is a popular choice for machine learning tasks due to its stability, performance, and extensive support for various tools and libraries commonly used in the field.

CPUs: The instances were equipped with 8 CPUs, providing ample processing power for training and inference tasks. Multi-core CPUs enable parallel processing, which can significantly expedite the execution of machine learning algorithms, especially those that can be parallelized effectively.

Cache Configuration: The CPUs featured different levels of cache memory to optimize data access speeds. This included L1 cache with 128 KiB, L2 cache with 4 MiB, and L3 cache with 24.8 MiB. Cache memory plays a crucial role in reducing memory latency by storing frequently accessed data closer to the CPU cores, thereby enhancing overall system performance.
NUMA Node Configuration: The CPUs were organized into a NUMA (Non-Uniform Memory Access) node configuration labeled as "NUMA node0 0-7". NUMA architecture is designed to optimize memory access in multi-socket systems by dividing memory into distinct nodes, each associated with a subset of CPUs. This configuration ensures efficient memory access and minimizes latency by prioritizing access to local memory within each NUMA node.

Overall, this setup provided a robust and well-configured environment for conducting our experiments, allowing us to assess the performance of machine learning models under controlled conditions with ample computational resources.

5. Dataset

The dataset under scrutiny comprises 60,000 images, each sized 32x32 pixels, and containing color information. These images are categorized into 10 distinct classes, with each class housing 6,000 images. The classes include Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck. This structured organization ensures a balanced representation across different categories, facilitating robust training and evaluation of machine learning models.

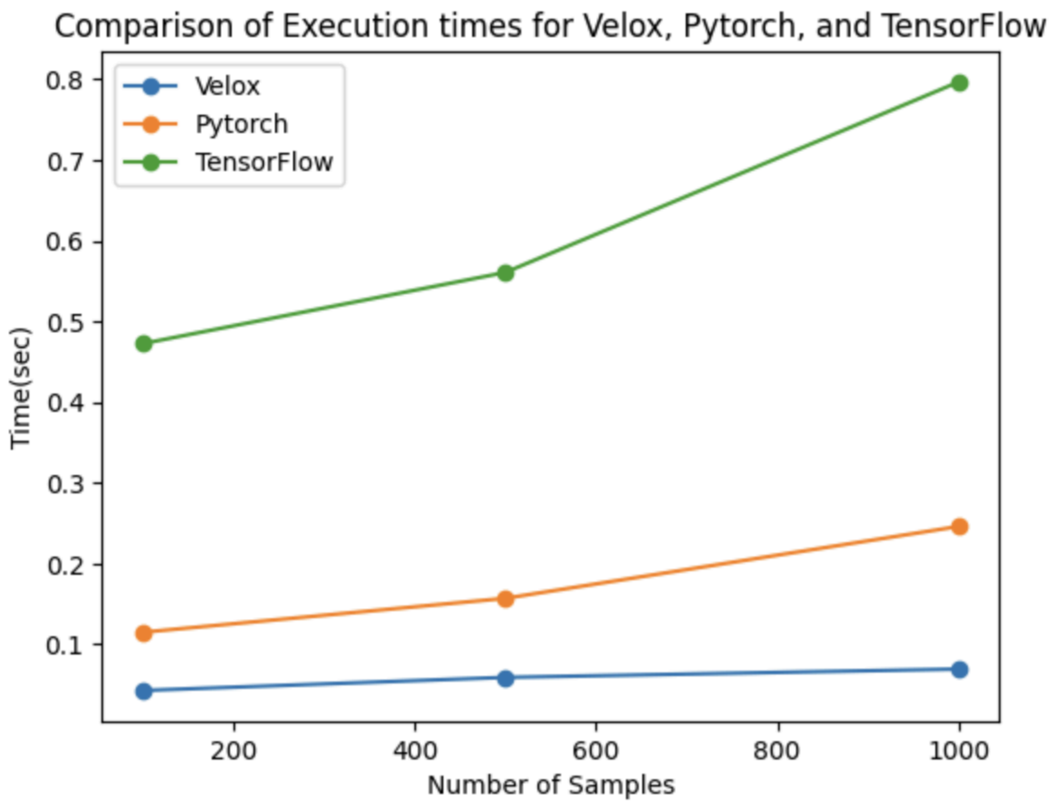
Here are the classes in the dataset, as well as 10 random images from each:
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

In terms of data partitioning, the dataset is split into a training set and a test set. The training set consists of 50,000 images, while the test set comprises 10,000 images. This division ensures that models are trained on a substantial portion of the data while still leaving a separate subset for unbiased evaluation of their performance. Each image in the dataset is associated with a single label representing its class, allowing for supervised learning approaches to be employed.

For the C++ dataset, the data is serialized in a specific format where the first byte denotes one of the 10 target labels corresponding to the image's class. Following this byte, the next 3,072 bytes contain the pixel values of the image. This format enables efficient storage and processing of the dataset, facilitating seamless integration into C++-based machine learning pipelines and applications.

6. Evaluation Results

The comparison of execution times for Velox, Pytorch, and TensorFlow provides valuable insights into the performance of these frameworks across varying sample sizes, with time as a primary metric. Velox consistently outperforms both Pytorch and TensorFlow across all sample sizes, showcasing its efficiency in handling larger datasets. The metrics reveal that as the number of samples increases, all three frameworks experience a rise in execution times, reflecting increased computational demands. However, Velox maintains notably lower execution times compared to its counterparts, indicating its prowess in swift data processing. Pytorch follows with moderately higher execution times, while TensorFlow exhibits the longest execution times, especially evident with larger sample sizes. These findings underscore Velox's competitive edge in terms of time efficiency, making it a compelling option for tasks demanding fast data processing compared to established frameworks like Pytorch and TensorFlow.



7. Conclusion and Future Work

In conclusion, our study has shed light on the performance of machine learning frameworks, particularly in the context of ResNet workload evaluation on VeloxML. Through our experiments, we have demonstrated the potential of VeloxML in optimizing data-intensive operations within a single host, showcasing its competitive edge in inference time and overall efficiency. Additionally, our findings underscore the importance of systematic evaluations in guiding practitioners toward selecting the most suitable framework for their specific project requirements.

Moving forward, several avenues for future work emerge, including implementing Residual Block Logic directly within VeloxML to further enhance its capabilities and efficiency in handling complex neural network architectures. Moreover, simplifying the Velox API to improve user experience and extending its functionality to encompass a broader range of machine learning functions will be pivotal steps towards fostering its adoption and usability among practitioners. Additionally, experimenting with more hyperparameters such as model size, number of execution threads, data splits, etc. will help in understanding how Velox performs in comparison with other popular Machine Learning frameworks. Lastly, conducting comparison studies with different hardware setups will provide valuable insights into the scalability and adaptability of VeloxML across diverse computational environments, thereby contributing to its continued refinement and evolution in the ever-expanding landscape of machine learning frameworks.

8. References

[1] Pedreira, Pedro, et al. "Velox: meta's unified execution engine." *Proceedings of the VLDB Endowment* 15.12 (2022): 3372-3384.

[2] Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." *Advances in neural information processing systems* 32 (2019).

[3] Abadi, Martín, et al. "{TensorFlow}: a system for {Large-Scale} machine learning." *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. 2016.

[4] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[5] Jain, Arpan, et al. "Performance characterization of dnn training using tensorflow and pytorch on modern clusters." *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2019.

[6] Dai, Hulin, et al. "Reveal training performance mystery between TensorFlow and PyTorch in the single GPU environment." *Science China Information Sciences* 65 (2022): 1-17.