Performance Evaluation of ML Workloads

A Comparative Study between Velox and PyTorch/TensorFlow

CSE-598 Data Intensive Systems for Machine Learning

Spring 2024

Submission by:

Nathaniel Ezra Lee Zheng, Yi-Hsiang Wang, Yuze Liao

Problem Statement

In the rapidly evolving field of machine learning, selecting an efficient framework is crucial for optimizing resource usage and improving execution efficiency. This report aims to compare three popular machine learning frameworks—Velox, PyTorch, and TensorFlow—on their performance in a standard image classification task using the MNIST dataset, which consists of 28x28 pixel grayscale images of handwritten digits. We conducted a detailed evaluation of how Velox, PyTorch, and TensorFlow handled various machine learning workloads, including Batch Normalization, Dropout, and Embedding, focusing on their impact on execution time and memory usage. The findings from this comparison will guide developers and businesses in choosing the most suitable framework for their specific needs, thus enhancing model performance while minimizing computational costs and resource utilization.

Literature Review

In the field of machine learning, the choice of framework significantly affects the efficiency and effectiveness of model training. Recent studies have focused on comparing the performance metrics of popular frameworks like PyTorch and TensorFlow across various computational environments. For instance, a detailed evaluation using the MNIST dataset showcased TensorFlow's superior performance in terms of training speed and energy efficiency on diverse hardware platforms, including AMD Ryzen CPU, Nvidia RTX GPU, and Apple M1 SoC [1]. This finding emphasizes the importance of framework optimization and compatibility with different computing environments, suggesting TensorFlow's advantage in scenarios that demand high efficiency and low power consumption.

On the other hand, another study examined the differences in training performance between TensorFlow and PyTorch within a single GPU environment [2]. This research evaluated the frameworks across a range of neural network models for tasks in computer vision, speech recognition, and natural language processing. The study identified key factors influencing performance, such as kernel implementations and memory management, providing valuable insights for practitioners when choosing a framework. The nuances in performance highlighted by this study underscore the critical role of task-specific requirements and resource management in selecting a framework [2].

These studies lay the groundwork for further exploration into how different machine learning frameworks, including the emerging Velox, perform across various domains such as image classification, natural language processing, and time series forecasting. Our research builds on these insights by comparing Velox with PyTorch and TensorFlow, focusing on aspects such as resource utilization, training time, and overall efficiency in handling machine learning workloads. By analyzing the strengths and limitations of each framework in specific contexts, our study aims to offer a detailed understanding of

their applicability to different machine learning tasks, thus guiding future framework selection and optimization efforts in the field.

Proposed Method and Algorithm

In this study, we focus on evaluating and comparing the performance of three prominent machine learning frameworks: Velox, PyTorch, and TensorFlow. Our primary evaluation metrics were execution time and memory usage, which we assessed by calculating averages based on 100 runs to ensure accuracy in measuring each framework's efficiency under various workloads.

We implemented a straightforward testing method, running a two-layer neural network across all frameworks with consistent input parameters—1000 samples and 500 dimensions—utilizing the MNIST dataset for image classification tasks. This network includes one hidden layer with ReLU activation and an output layer with softmax activation. Additionally, we explore how each framework handles additional machine learning workloads such as Batch Normalization, Dropout, and Embedding, to provide a comprehensive performance comparison.

Our hypothesis was that there would be notable differences in execution time and memory usage among the frameworks, especially when managing complex tasks like image classification combined with other machine learning workloads. We find that frameworks with superior memory management and execution optimization demonstrate enhanced performance. These differences provide valuable insights, guiding the selection of the most suitable framework for specific machine learning tasks.

This research provided a detailed assessment of each framework's performance, including graphical representations that visually demonstrated execution time and memory usage across various machine learning workloads.

Experimental Environment and Setup

We ensured the accuracy and reproducibility of our experiments by meticulously detailing the hardware and software environments used. The hardware setup included an AMD Ryzen 9 processor, which has 8 cores and a core speed of 4GHz, providing robust computational power. Additionally, the system was equipped with 16 GB of RAM, accommodating the extensive data processing and computational demands of complex models.

For the software setup, all experiments were conducted on the Linux operating system, known for its stability and efficiency. To maintain consistency across tests, we utilized Docker containers for deploying and running each of the machine learning frameworks. This method ensured a uniform and isolated environment, reducing potential variability that could impact the experimental results.

Furthermore, we standardized the configuration settings across the three frameworks: Velox, PyTorch, and TensorFlow. Each framework was set up to use the same two-layer neural network architecture, which included one hidden layer with ReLU activation and an output layer with softmax activation. The training and testing procedures were also uniform, with the same number of training iterations, batch size, and learning rate applied to ensure that each framework operated under identical conditions.

These careful considerations in setting up the experimental environment allowed us to conduct a fair and controlled comparison of the frameworks, providing clear insights into their performance and establishing a reliable foundation for future research.

Data Used for Experiments

We utilized the MNIST dataset for our experiments, a public dataset extensively used for benchmarking machine learning models in image classification tasks. Developed by the United States National Institute of Standards and Technology (NIST), the MNIST dataset serves as a standardized platform enabling researchers to test and compare various machine learning algorithms.

The structure of the dataset comprises 70,000 grayscale images of handwritten digits, each 28x28 pixels in size. It is segmented into two subsets: 60,000 images for training and 10,000 for testing, with each image labeled according to the digit it represents, from 0 to 9.

To prepare the data for effective model training, we carried out a series of preprocessing steps:

1. Normalization: We normalized all pixel values to a range between 0 and 1, enhancing the training process's speed and stability.
2. Reshaping: The image data was converted from a two-dimensional array format to a one-dimensional vector format, making it suitable for neural network input.

These preprocessing measures ensured the data's consistency and applicability, laying a solid foundation for the validity and reproducibility of our experiments.

Experimental Results

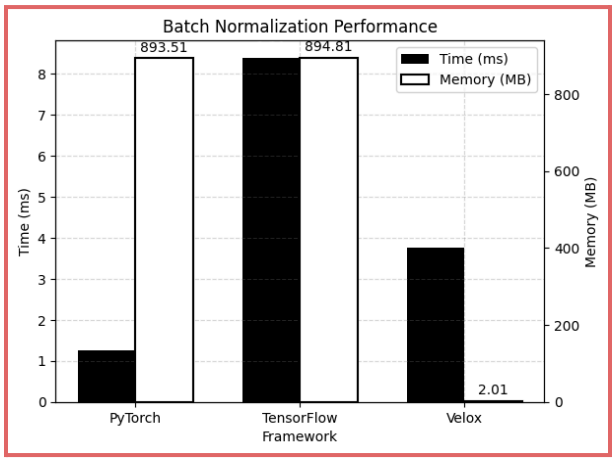

Figure 1. Batch Normalization Performance

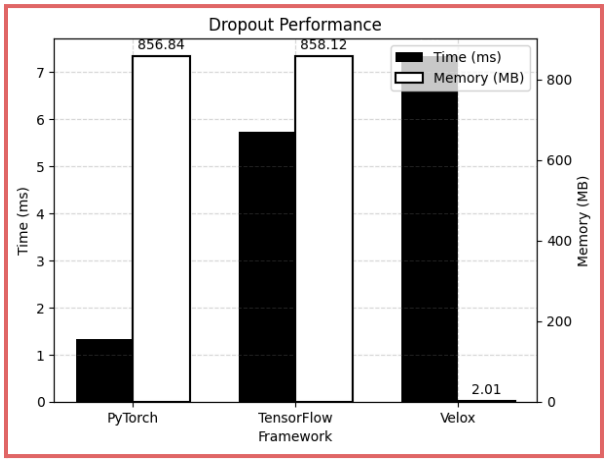

Figure 2. Dropout Performance

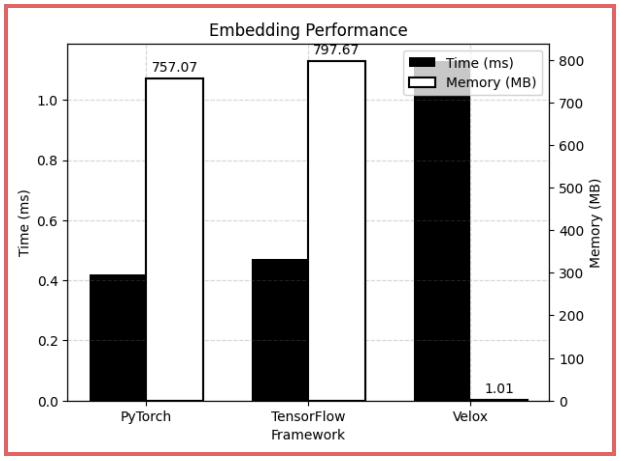

Figure 3. Embedding Performance

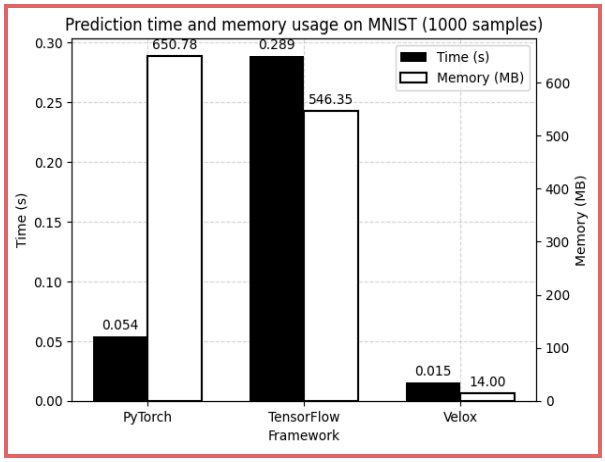

Figure 4. NN Performance on MNIST

In the experiment, we compared the performance of three machine learning frameworks across various ML workloads. The results revealed that Velox achieves notable memory savings in tasks such as batch normalization, dropout, and embedding. This characteristic makes Velox potentially more suitable for environments where memory resources are limited.

However, it was also observed that Velox's low memory usage might compromise its execution speed, especially in more complex machine learning workloads. In contrast, while PyTorch and TensorFlow might take longer in execution times, they do not offer the same level of memory efficiency as Velox.

Overall, the choice of the right machine learning framework should be based on the specific needs of the application. If memory resources are a constraint, Velox could be a preferred option; however, for applications where execution speed is a priority, PyTorch or TensorFlow might be more appropriate. This experiment provides deeper insights into the performance characteristics of each framework, helping guide future framework selection and optimization efforts.

Conclusion

This study conducted a detailed comparison of Velox's preprocessing performance against other frameworks. We delved into its efficiency in running neural network models. Remarkably, Velox emerged as a frontrunner in prediction speed for neural network models while consuming less memory compared to PyTorch and Tensorflow. This finding underscores Velox's prowess in optimizing resource utilization without compromising on computational speed, positioning it as a highly competitive choice for deploying neural network models in production environments.

In addition, we explored Velox's capabilities in resource management, finding it to perform well in terms of memory usage and processing speed, which is crucial for running large machine learning models. The results demonstrate that Velox exhibits superior performance in preprocessing large datasets, particularly in terms of efficiency during data cleaning and transformation processes.

Our attempts to utilize Velox for building models beyond neural networks revealed that while Velox performs adequately with some model types, its performance could be improved in scenarios involving

greater complexity or larger data volumes. This indicates that Velox, as a versatile machine learning framework, is broadly applicable but may require further optimization for specific applications.

Future Work

For future research directions, it is recommended to further explore optimization techniques for Velox in different types of data preprocessing, particularly with unstructured data. Additionally, extending research into Velox's resource management capabilities, especially its performance in multitasking and parallel processing environments, would be beneficial.

Moreover, future studies should consider expanding the use of Velox to build a wider variety of machine learning models, such as decision trees and ensemble methods, and assess its suitability and performance across different machine learning tasks. These studies will provide a more comprehensive evaluation of Velox's performance and offer theoretical and empirical foundations for its improvement in practical applications.

Reference

[1] Germino, J., Inzitari, T., & Le, N. (n.d.). Benchmarking PyTorch's and TensorFlow's speed and power performance building classifiers on the MNIST dataset. Retrieved February 17, 2024, from http://www.tsinzitari.com/Portfolio/Benchmarking%20Pytorch%20and%20Tensorflow.pdf

[2] Dai, H., Peng, X., Shi, X., et al. (2022). Reveal training performance mystery between TensorFlow and PyTorch in the single GPU environment. Science China Information Sciences, 65, 112103. https://doi.org/10.1007/s11432-020-3182-1

[3] R. M. Rakshith, V. Lokur, P. Hongal, V. Janamatti and S. Chickerur, "Performance Analysis of Distributed Deep Learning using Horovod for Image Classification," 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2022, pp. 1393-1398, doi: 10.1109/ICICCS53718.2022.9788247

[4] A. Jain, A. A. Awan, Q. Anthony, H. Subramoni and D. K. D. Panda, "Performance Characterization of DNN Training using TensorFlow and PyTorch on Modern Clusters," 2019 IEEE International Conference on Cluster Computing (CLUSTER), Albuquerque, NM, USA, 2019, pp. 1-11, doi: 10.1109/CLUSTER.2019.8891042

[5] Mahfoudh, S., & Gannouni, S. (2021). Design and Implementation of a Big Data Analytics Platform Based on Apache Spark and Kafka. Cluster Computing, 24, 1811–1833. https://doi.org/10.1007/s10586-021-03240-4

[6] Abadi, M., Barham, P., Chen, J., et al. (2018). TensorFlow: A System for Large-Scale Machine Learning. arXiv preprint arXiv:1802.09941. https://arxiv.org/abs/1802.09941