CSE 598 : Data Intensive Systems for Machine Learning

Comparative Analysis of Image Classification Performance: Velox, PyTorch, and TensorFloc

Deva Dharshini Ravichandran Lalitha, Junaita Davakumar Balaji Davakumar, Vallikannu Chockalingam

INTRODUCTION

The objective of this project is to implement a standard machine learning (ML) workload - specifically, an image classification task - across three distinct environments: Velox, PyTorch, and TensorFlow. By utilizing the CIFAR-10 dataset, this project aims to not only demonstrate the practical application of these frameworks but also to critically evaluate their performance in terms of training time, inference time, accuracy, and resource utilization.

PROBLEM STATEMENT

Image classification is an essential operation within the vast domain of machine learning, acting as a critical benchmark for assessing the performance of numerous machine learning platforms. It serves not only to categorize images into predefined categories but also as a litmus test for the sophistication and robustness of machine learning algorithms. While PyTorch and TensorFlow have emerged as the go-to frameworks for a myriad of image classification tasks, owing to their comprehensive libraries and supportive communities, the proficiency and effectiveness of Velox's ml_functions in this arena have yet to be fully uncovered. This oversight presents a notable gap in the comparative analysis of machine learning tools, particularly in the context of real-world applications.

This project is designed to address this discrepancy by conducting a thorough comparative study of these frameworks, specifically focusing on their performance in image classification tasks. By evaluating each framework's ease of use, scalability, support for diverse model architectures, and efficiency in processing large datasets, this study aims to illuminate the unique strengths and potential limitations of Velox's ml_functions relative to its more established counterparts. Through this analysis, we seek to provide a comprehensive overview that will aid researchers and developers in making informed decisions when selecting the most suitable framework for their specific machine learning projects, especially those involving complex image classification challenges.

PROPOSED SOLUTION

Dataset

- The CIFAR-10 dataset is a collection of 60,000 images, each measuring 32x32 pixels in full color, distributed across 10 distinct categories.
- These categories encompass a variety of objects and animals, making CIFAR-10 a versatile dataset for benchmarking image classification models. Given its diversity and size, CIFAR-10 challenges models to accurately recognize and differentiate between complex patterns and features, establishing it as the primary dataset for evaluating the performance of our project's machine learning frameworks.

Model Architecture

The core of our image classification model is a Convolutional Neural Network (CNN), a type of deep neural network especially suited for processing grid-like data such as images. Our CNN architecture is composed of several key components:

- **Convolutional Layers**: These layers are the building blocks of the CNN, designed to automatically and adaptively learn spatial hierarchies of features from input images.
- **ReLU Activation Functions**: The Rectified Linear Unit (ReLU) activation function introduces non-linearity into the model, allowing it to learn more complex patterns.
- **Pooling Layers**: Pooling (often max pooling) reduces the dimensionality of each feature map while retaining the most essential information, making the network more efficient and less prone to overfitting.
- **Fully Connected Layers**: Towards the end of the network, fully connected layers combine features learned by earlier layers to make final classification predictions.

Implementation Plan

- **Velox**: Leverage `ml_functions` for optimized data preprocessing and evaluate the feasibility and performance of implementing CNN layers directly in Velox.
- **PyTorch**: Utilize PyTorch's dynamic computation graph and extensive API to construct and train the CNN model.
- **TensorFlow**: Apply TensorFlow and Keras for model construction, benefiting from the static computation graph and deployment capabilities.

Performance Evaluation Criteria

- **Training and Inference Time**: Record the duration of model training and inference.
- **Accuracy**: Compare the classification accuracy on the CIFAR-10 test set.
- **Resource Utilization**: Monitor CPU/GPU usage and memory consumption.
- **Development Experience**: Assess the ease of implementation, debugging, and utilizing framework-specific features.

LITERATURE SURVEY

Chirodea, Florin, et al. in their study titled "Comparison of Tensorflow and PyTorch in Convolutional Neural Network-based Applications,"[1] highlighted the operational differences between PyTorch and TensorFlow. PyTorch employs a dynamic computation graph that allows for more procedural code execution, contrasting with TensorFlow's need for predefined models before execution. The study observed that while PyTorch offers faster training and execution times, it does so with a slight compromise on accuracy compared to TensorFlow. This suggests a nuanced trade-off between speed and precision when choosing between these frameworks for specific applications.

Another insightful comparison, presented in "PyTorch vs TensorFlow: Which One Is Right For You"[2] by Vast.ai, delves into performance benchmarks, training time, memory usage, and usability of both frameworks. It noted PyTorch's superior training speed, leveraging CUDA for faster task completion, albeit with higher memory consumption compared to TensorFlow. This comparison underlines the importance of project-specific requirements in selecting the most suitable framework, with PyTorch being favored for rapid development cycles and TensorFlow for its memory efficiency and structured environment conducive to large-scale deployments.

Built In's article, "PyTorch vs. TensorFlow for Deep Learning in 2024,"[3] offers a detailed analysis of the mechanisms underlying each framework, distributed training capabilities, visualization tools, and production deployment. It emphasizes TensorFlow's robust production deployment options, including TensorFlow Serving, and its comprehensive visualization tool, TensorBoard. PyTorch, while user-friendly and favored for research and development due to its Pythonic ease and dynamic graph construction, requires additional tools for deployment in production environments. This dichotomy underscores TensorFlow's suitability for production-level projects and PyTorch's advantages for research and development tasks.

EXPECTED OUTCOMES

This project is expected to yield a comprehensive comparison of Velox, PyTorch, and TensorFlow in handling an image classification task, providing insights into:

- The practical performance trade-offs between these frameworks.
- The suitability of Velox's `ml_functions` for ML workloads compared to established frameworks.
- Recommendations for framework selection based on specific project requirements, such as development speed, performance, or resource constraints.

IMPLEMENTATION AND EVALUATION PLAN

- **Setup**: Ensure access to a GPU-enabled environment for fair comparison across frameworks.
- **Development**: Implement the CNN model in Velox, PyTorch, and TensorFlow, ensuring equivalent model architecture and training conditions.
- **Evaluation**: Execute the performance evaluation plan, collecting data on training/inference time, accuracy, and resource utilization.
- **Analysis**: Analyze and interpret the collected data, preparing a comparative report on the findings.

CONCLUSION

This project aligns with the course's focus on ML systems architecture and design principles, offering hands-on experience with different ML frameworks. Through this comparative analysis, we aim to deepen our understanding of how different tools can be leveraged for ML tasks, guiding future decisions in framework selection for optimized performance and efficiency.

REFERENCES

[1] Chirodea, Florentin, et al. "Comparison of Tensorflow and PyTorch in Convolutional Neural Network-based Applications." ResearchGate, www.researchgate.net/publication/342344662_Comparison_of_Tensorflow_and_PyTorch_in_Convolutional_Neural_Network-based_Applications. Accessed 17 Feb 2024. This study highlights the performance differences between TensorFlow and PyTorch, noting that while PyTorch may offer faster training and execution times, TensorFlow provides slightly better accuracy.

[2] "PyTorch vs TensorFlow: Which One Is Right For You." Vast.ai, 9 Nov. 2023, vast.ai/article/PyTorch-vs-TensorFlow. Accessed 17 Feb 2024. This article provides a comparison between PyTorch and TensorFlow, focusing on aspects such as performance, training time, memory usage, and ease of use, to help readers understand which library may be more suitable for their project's needs.

[3] "PyTorch vs. TensorFlow for Deep Learning in 2024." Built In, builtin.com/software-engineering-perspectives/pytorch-vs-tensorflow-deep-learning. Accessed 17 Feb 2024. This article discusses the key differences between PyTorch and TensorFlow, including their computational graph mechanisms, distributed training, visualization tools, and production deployment capabilities.

TEAM INFORMATION:

Team name - Machine learning Misfits,
Team number - 10,
Team members -

- Deva Dharshini Ravichandran Lalitha (1229734859),
- Junaita Davakumar (1225427229),
- Vallikannu Chockalingam (1229609266)