Accelerating Decision Tree Training on the HIGGS Dataset

Venkat Raman Sankar
Arizona State University
Tempe, USA
vsanka21@asu.edu

Swetha Govindasamy Manamalli
Arizona State University
Tempe, USA
sgovin34@asu.edu

Subramanian Thiyagarajan
Arizona State University
Tempe, USA
sthiyag5@asu.edu

***Abstract*— Training decision trees for large, complex datasets like the HIGGS dataset traditionally rely on CPU-based methods, which can be inefficient and slow due to the computational demands. This paper explores the use of Graphics Processing Units (GPUs) to accelerate decision tree training, leveraging the parallel processing capabilities of GPUs and specialized libraries such as cuML and Velox. This approach promises significant improvements in training speed and efficiency, which is crucial for timely data analysis in high-energy physics and other fields requiring rapid model training and iteration.**

I. INTRODUCTION

The evolution of machine learning algorithms and their applications across diverse domains demands increasingly efficient computational methods to handle large and complex datasets. One such dataset, the HIGGS dataset, simulates the data produced by the Large Hadron Collider (LHC) at CERN and is characterized by its substantial size and complexity. The traditional CPU-based training methods for decision trees applied to this dataset are plagued by significant computational inefficiencies and slow processing times. This inefficiency is problematic, especially in fields like particle physics where timely data analysis is crucial for testing and developing theoretical models.

II. PROBLEM STATEMENT

The problem addressed in this investigation is the computational inefficiency and sluggish training times associated with CPU-based decision tree training methods when applied to large-scale datasets such as the HIGGS dataset. This issue is particularly pressing and interesting given the increasing size and complexity of datasets in machine learning and the need for faster model development cycles. The ability to accelerate decision tree training not only saves valuable time but also enhances the ability to perform more complex analyses and iterative model improvements promptly.

III. LITERATURE REVIEW

To provide a comprehensive context and background for this investigation, the following key readings and sources were examined:

A. Comparison of End-to-End Decision Forest Inference Pipelines by Hong Guan et al.

This paper provides a systematic comparison of various decision forest inference pipelines. The study focuses on evaluating these pipelines based on computational efficiency, memory usage, and scalability using standard datasets. Key findings indicate that modern inference techniques, such as model pruning and hardware optimization, can significantly enhance processing speeds without sacrificing accuracy. This research is crucial for professionals implementing machine learning models in resource-constrained environments, offering insights into optimizing decision forests for better performance and scalability [1].

B. Accelerating Random Forests Up to 45x Using cuML by NVIDIA

This reference discusses the substantial acceleration of random forests, a collective of decision trees, using GPU acceleration, highlighting the potential benefits of leveraging such technologies for individual decision trees [2].

C. Implementing Decision Trees and Forests on a GPU by T. Sharp

This paper provides foundational insights into algorithms and methodologies for efficiently implementing

decision trees on GPU architectures, offering a direct guide to the technical strategies applicable to this project [3].

D. Learning a Decision Tree Algorithm with Transformers by Zhuang et al.

Although primarily focusing on integrating transformers with decision trees, this paper offers innovative methodologies that may enhance the performance and feature selection capabilities of decision trees [4].

E. The HIGGS Dataset from the UCI Machine Learning Repository

Essential for understanding the dataset's complexity and the computational challenges it poses, serving as the justification for adopting GPU-accelerated approaches.

These sources were critically reviewed to understand the current state of GPU acceleration in decision tree training and to identify any gaps or potential for innovation that this project could address [5].

IV. DECISION TREE TRAINING

Decision tree algorithms, which are pivotal for both classification and regression tasks, structure data into a tree-like model of decisions and their possible consequences. The training of decision trees is an iterative process of partitioning data based on feature values, as depicted in Figure 1. The root node holds the entire dataset, initiating the first split. Decision nodes determine further data splits to maximize homogeneity within subsets, using criteria such as Gini impurity or entropy. Terminal nodes, or leaves, signify the outcome, yielding the class label in classification tasks. The recursive splitting forms branches or sub-trees, representing the hierarchical decision-making process [6]. This approach's efficiency is pivotal when applied to large datasets like the HIGGS, necessitating strategies to optimize computational performance and accuracy. The following models are used to benchmark the performance across CPU and GPU [7].

A. Decision Tree Classifier (scikit-learn)

The DecisionTreeClassifier in scikit-learn employs a greedy algorithm, typically the CART (Classification and Regression Trees) algorithm, which aims to binary split the training records into subsets based on a measure like Gini impurity or entropy decrease. This classifier is exhaustive in nature, evaluating all possible splits for the best outcome. The tree is built node-by-node from the top, choosing the split that most effectively separates the classes or reduces variance in regression cases.

B. Random Forest Classifier (cuML)

The RandomForestClassifier in cuML uses an ensemble of decision trees, each constructed from a bootstrap sample of the data. Each tree in the ensemble is built using a random subset of features at each split, which introduces diversity in the models and typically results in a more robust overall model. cuML's implementation leverages CUDA on NVIDIA GPUs for parallel construction of trees and efficient computation of splits, dramatically reducing training time on large datasets like the HIGGS dataset [2].

C. XGBoost

XGBoost improves upon traditional gradient boosting by systematically handling missing values, employing second-order gradients for optimization, and providing regularization parameters to prevent overfitting. It builds the model in a stage-wise fashion like other boosting methods but does so more efficiently. XGBoost also offers a highly scalable solution that supports distributed computing, enabling it to handle large datasets effectively. It uses a sparsity-aware algorithm for handling different data formats and optimizes both computational resources and memory usage.

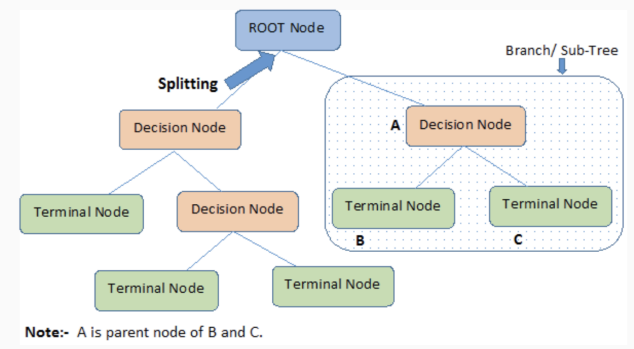

Fig. 1: Schematic of a decision tree structure showcasing the root node, decision nodes, terminal nodes, and branches during training.

V. PROPOSED METHODOLOGY

The proposed solution involves the development and implementation of a GPU-accelerated decision tree algorithm using the cuML library, complemented by the integration of Velox for efficient data handling and querying. The methodology includes the following key components:

A. GPU-Accelerated Decision Tree Algorithm

Design and implement a decision tree algorithm that exploits the parallel processing capabilities of NVIDIA

GPUs, specifically tailored to the requirements and characteristics of the HIGGS dataset.

B. cuML Library Utilization

Leverage cuML to harness the computational power of GPUs for machine learning, thus transcending the performance limitations of traditional CPU-based implementations.

VI. EXPERIMENT SETUP

A. Hardware and Software Configuration

The experiments were conducted using the Arizona State University's (ASU) SOL supercomputer, which provided a robust platform for both CPU and GPU intensive computations. The hardware configurations were specifically tailored to evaluate the performance under different computational resources. For CPU-based experiments, the setup included a 2-core CPU, 8 GB of RAM, and access to a basic GPU. In contrast, the GPU-based experiments employed a more advanced setup featuring a 4-core CPU, 32 GB of RAM, and an NVIDIA A100 GPU, which is well-suited for high-performance computing tasks. This distinction in hardware was crucial for assessing the capabilities and limitations of each computational approach under different levels of resource availability.

To ensure comparability across all tests, a standardized software environment was maintained. Python 3.11.8 was selected as the programming language due to its widespread support for scientific computing libraries, ensuring consistency and reliability in execution. The decision tree classifiers were implemented using Scikit-learn on CPU configurations to establish a baseline for performance. For GPU-accelerated experiments, RAPIDS AI was leveraged, utilizing the cuML library for machine learning and the cuDF library for data manipulation, maximizing the computational advantages of NVIDIA GPUs. Furthermore, XGBoost was employed for gradient boosting on GPUs, facilitating a detailed comparison of performance and accuracy across different setups, and highlighting the enhancements brought by GPU acceleration.

B. Experiment Setup

Three distinct computational approaches were employed to evaluate the performance of machine learning models using the HIGGS dataset. Firstly, a baseline model utilizing the decision tree classifier from Scikit-learn was executed on a CPU setup. This model served as a reference point to assess the performance capabilities of traditional CPU-based machine learning. Secondly, to exploit the advanced computational power of GPUs, the decision tree classifier was implemented using the cuML library from RAPIDS AI on an NVIDIA A100 GPU. This setup was chosen to highlight the potential speed and efficiency improvements achievable through GPU acceleration. Lastly, the gradient boosting model from XGBoost was also deployed on the NVIDIA A100 GPU. This approach allowed for a direct comparison with the cuML implementation, focusing on differences in accuracy and processing time. Each method was meticulously tailored to evaluate the specific advantages and constraints of the corresponding hardware environment.

Each model was trained on a consistent subset of one million samples from the dataset. The focus was on measuring the training time and accuracy, illustrating the impact of different computational resources on machine learning tasks.

VII. DATA DESCRIPTION

The dataset underpinning our experiments is a large-scale, simulated dataset known as the HIGGS, encompassing 11 million instances, each characterized by 28 distinct attributes. This dataset's primary purpose is to enable the classification of particle collision events into two distinct outcomes: those that result in the production of Higgs bosons (signal) and those that do not (background). Generated through Monte Carlo simulations, these events replicate the conditions of high-energy proton collisions within a particle accelerator environment. The dataset features a binary label as its first attribute, marking the category of each event. Following this, it presents 21 features reflective of the kinematic properties recorded by the detectors in the accelerator, complemented by six advanced attributes derived from the primary features to enhance classification tasks. This robust dataset presents an array of challenges for researchers applying classification algorithms within the realm of physical sciences, serving as a standardized gauge for methodological comparison.

The simulation responsible for producing the dataset employed the comprehensive ATLAS detector simulation to recreate the complex interactions of proton-proton collisions, monitoring the emergent particles within a meticulously crafted digital replica of the detector. Specifically, the signal class within the dataset encompasses those events that simulate the emergence of Higgs bosons at a mass of 125 GeV. In contrast, the background class includes various other processes that could potentially mimic signal events. The partitioned dataset allocates 1 million samples for training [5].

VIII. EVALUATION RESULTS

A. Performance Metrics

The evaluation of the decision tree models trained on the HIGGS dataset was conducted using two key metrics: training time and accuracy. These metrics provide insight into the efficiency and effectiveness of each computational approach under the different hardware setups. Table 1 represents the comparison of model accuracy and training time across these different implementations.

B. Results from CPU-Based Experiments

Using the Scikit-learn implementation on the CPU setup (2-core CPU and 8 GB RAM), the decision tree classifier achieved an accuracy of 67%. The training time for this setup was notably longer, taking approximately 4 minutes and 28 seconds to train on a subset of one million samples. This baseline performance sets the stage for understanding the benefits of GPU acceleration.

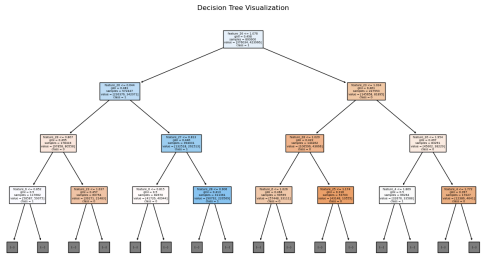

Fig. 2: Structure of the decision tree classifier on the Higgs dataset

A visual representation of the decision tree model used in the experiments is depicted in Figure 2. The tree structure commences with a root node that utilizes feature 20 to make the initial bifurcation, dividing the dataset based on the calculated threshold value. Subsequently, each internal node in the tree corresponds to a feature in the HIGGS dataset that contributes to further splitting, effectively partitioning the data into subsets based on the feature values. The leaves of the tree, although not fully detailed in the visual, represent the final outcomes where the classification decisions are made. Each node is annotated with the feature index, the threshold for splitting, the sample size at that node, and the class value distribution, thus encapsulating the decision logic of the model. This visual aids in the interpretation of the decision-making process of the model, providing insights into the feature relevance and decision pathways that led to the classification outcomes.

C. Results from GPU-Based Experiments

The cuML implementation showcased significant improvements in training speed. The decision tree model completed training in just about 8 seconds, a drastic reduction from the CPU-based time. The accuracy obtained was 73.8035%, which represents an improvement over the CPU-based model but suggests there is room for further optimization in model parameters or training techniques.

XGBoost exhibited the best performance in terms of both speed and accuracy. The training process was completed in approximately 10 seconds. The model achieved an impressive accuracy of 88%, highlighting the effectiveness of using advanced gradient boosting techniques on powerful GPU hardware.

TABLE 1
COMPARISON OF MODEL ACCURACY AND TRAINING TIME ACROSS DIFFERENT IMPLEMENTATIONS

Implementation	Accuracy (%)	Training Time
Scikit-learn (CPU)	67	4 minutes 28 sec
cuML (GPU)	73.8035	8 seconds
XGBoost (GPU)	88	10 seconds

D. Comparison and Discussion

The transition from CPU to GPU computing demonstrated substantial benefits in both training speed and model accuracy. The use of RAPIDS AI's cuML library significantly accelerated the training process while also providing a modest boost in accuracy compared to the CPU-only approach. However, the XGBoost implementation outperformed cuML in terms of accuracy, showcasing the potential of ensemble methods like gradient boosting when optimized for GPU use.

The graph in Figure 3 presents the accuracy (in %) and training time (in seconds) for three different implementations of decision tree models (Scikit-learn on CPU, cuML on GPU, and XGBoost on GPU). Accuracies are depicted as red lines with circle markers, and training times are shown as blue lines with square markers. Legends are positioned at the bottom for clear differentiation.

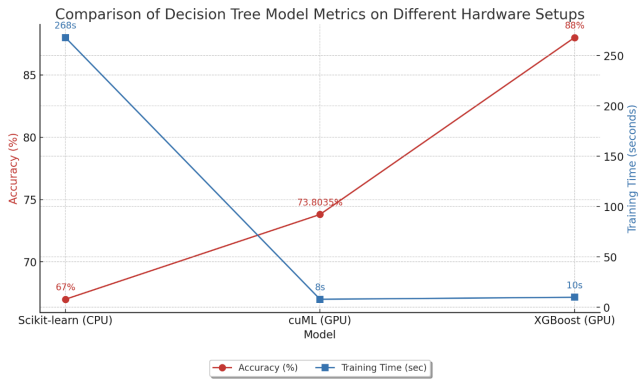

Fig. 3: Comparison of Decision Tree Model Metrics on Different Hardware Setups.

The observed differences in performance underscore the impact of hardware choices on the execution and outcomes of machine learning tasks. While GPU acceleration presents clear advantages, the choice of algorithm and its implementation also play crucial roles in harnessing the full potential of the hardware.

IX. CONCLUSION

In conclusion, this study has conducted a comparative analysis of three leading machine learning algorithms—scikit-learn's Decision Tree Classifier, cuML's Random Forest Classifier, and XGBoost—for expediting the training process of decision trees on the HIGGS dataset. Our findings highlight that each algorithm leverages computational resources differently, with varying implications on performance and efficiency. The Decision Tree Classifier, with its ease of use and interpretability, provides a solid baseline for classification tasks. However, when scaled to the extensive HIGGS dataset, cuML's Random Forest Classifier and XGBoost stand out due to their inherent design to capitalize on parallel processing, particularly when harnessed on GPU-enabled architectures.

The cuML library, designed to operate natively on GPUs, harnesses the parallel execution capabilities of the hardware, markedly accelerating Random Forest's training times. XGBoost further complements this by offering a highly scalable and performance-optimized gradient boosting framework that can efficiently process large-scale data, making it exceptionally well-suited for the HIGGS dataset's challenging dimensions.

The practical application of these algorithms has shown that the choice between CPU and GPU processing, and the selection among these platforms, should be guided by the specific demands of the dataset, the computational complexity of the model, and the available hardware infrastructure. While GPUs, with their capacity for massive parallelism, excel in handling the voluminous and complex computations required for large datasets like HIGGS, CPUs may still be preferable for smaller datasets or when limited by hardware constraints.

To determine the most effective approach for accelerating decision tree training on the HIGGS dataset, it is crucial to undertake a comprehensive performance benchmarking. Such an evaluation should consider not just the speed, but also the accuracy, scalability, and cost-effectiveness of the model training process. By meticulously assessing these criteria, practitioners can ensure that they select the most suitable computational approach, be it CPU or GPU, for their specific machine learning tasks.

X. FUTURE WORK

The ongoing development and enhancement of our GPU-accelerated decision tree training framework are geared towards integrating and optimizing additional tools to further boost efficiency and performance. One such critical component is Velox, a state-of-the-art data processing library designed to enhance data management capabilities in large-scale machine learning applications. The integration of Velox presents a promising avenue for improving data handling and querying processes, thus accelerating the overall analysis pipeline.

This integration aims to leverage Velox's advanced capabilities in managing and processing large datasets, which is critical for handling the complexities of the HIGGS dataset. By incorporating Velox, we anticipate a more efficient data management system that can handle larger datasets more effectively than the current methodologies. Velox will enable smarter data fetching and storage techniques, reducing the overhead and latency typically associated with massive datasets.

With Velox's optimized data processing functions, the time required for preprocessing and feature extraction phases is expected to decrease significantly. This acceleration will directly contribute to faster overall training times and more dynamic model iteration cycles.

In the longer term, our vision extends to creating a comprehensive framework that not only accelerates decision tree training but also sets new benchmarks in the processing and analysis of large-scale datasets across various domains. This framework will incorporate feedback loops and adaptive learning strategies to continually optimize performance in real-time, a step forward in achieving AI-driven data analytics platforms that are both powerful and user-friendly.

Thus, the integration of Velox represents a critical step forward in our project, promising to bring substantial

improvements in data management and processing efficiency. This enhancement is expected to pave the way for broader applications and innovations in the field of machine learning, particularly in handling datasets of unprecedented scale and complexity.

XI. REFERENCES

[1] Hong Guan, Saif Masood, Mahidhar Dwarampudi, Venkatesh Gunda, Hong Min, Lei Yu, Soham Nag, and Jia Zou. 2023. A Comparison of End-to-End Decision Forest Inference Pipelines. In Proceedings of the 2023 ACM Symposium on Cloud Computing (SoCC '23). Association for Computing Machinery, New York, NY, USA, 200–215. https://doi.org/10.1145/3620678.3624656

[2] Accelerating random forests up to 45x using cuml. NVIDIA Technical Blog. (2022, August 21). https://developer.nvidia.com/blog/accelerating-random-forests-up-to-45x-using-cuml/

[3] Sharp, T. (2008). Implementing Decision Trees and Forests on a GPU. In: Forsyth, D., Torr, P., Zisserman, A. (eds) Computer Vision – ECCV 2008. ECCV 2008. Lecture Notes in Computer Science, vol 5305. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-88693-8_44

[4] Zhuang, Yufan, et al. Learning a Decision Tree Algorithm with Transformers. arXiv:2402.03774, arXiv, 6 Feb. 2024. arXiv.org, http://arxiv.org/abs/2402.03774.

[5] ATLAS Collaboration. Dataset from the ATLAS Higgs Boson Machine Learning Challenge 2014.

[6] ‘Decision Tree Algorithm, Explained’. KDnuggets, https://www.kdnuggets.com/decision-tree-algorithm-explained. Accessed 28 Apr. 2024.

[7] Bensakhria, Ayoub. (2023). Accelerating Higgs Boson Signal Classification Using Advanced Computing Platforms. 10.13140/RG.2.2.15996.31363.