Predict GPU floating-point performance metrics based on hardware specifications

Jing Yang
School of Computer Science
University of Bologna
Cesena, FC 47521
Email: jing.yang5@studio.unibo.it

Ge Li
School of Computer Science
University of Bologna
Cesena, FC 47521
Email: ge.li2@studio.unibo.it

***Abstract*—Based on the chips dataset [Sun et al.(2019)],we propose a machine learning approach to predict Graphics Processing Unit (GPU) floating-point performance metrics (FP32 GFLOPS) based on hardware specifications, specifically key hardware parameters: Die Size, TDP(thermal design power), and Transistors Counts.**

We begin by preprocessing the dataset, which includes removing rows with missing values, eliminating irrelevant columns, and converting specifications (e.g., die size, initially stored as categorical data) into normalized numerical features. Next, we perform feature analysis to identify parameters with the most significant impact on GPU performance. To visualize interactions, we generate a 3D plot where spatial coordinates represent hardware parameters, and color intensity reflects performance precision levels. Finally, we employ machine learning algorithms such as KNN, Random Forest, and AutoML to predict GPU performance based on these features.

1. Introduction

The group consists of two master's students: Ge Li, who initiated the research topic, is responsible for assignment writing and identifying related work; and Yang, who specializes in code structuring, refining model testing, execution, and debugging. The research focus, dataset selection, and model development were were collaborative efforts.

Given the background of GPU development and physical limits of transistor miniaturization [1] in the era of increasing General Purpose AI demand(training and inference), further progress in deep learning is largely gated by GPU performance [Dally et al.(2021)]. This research aims to identify specific hardware features influencing GPU performance and to analyze how these features contribute, enabling predictions of performance trends.

2. Related Work

The dataset utilized in this study builds upon prior work outlined in Summarizing CPU and GPU Design Trends with Product Data by [Sun et al.(2019)]. The original study aimed to investigate the validity of Moore's Law and Dennard Scaling amidst challenges posed by the practical limitations of transistor scaling. By collecting data on over 4,000 CPUs and GPUs from vendors like Intel, AMD, and NVIDIA, the study identified that GPU performance improvements were significantly driven by increases in GPU frequency, improvements in the thermal design power (TDP), and growth in die size.

The report Predicting GPU Performance[2] from Epoch AI develops a simple model to generate predictions of GPU performance, focusing on variables that directly contribute to higher FLOP/s (FP32), but different with our approach in feature and model selection. They plot individual features (e.g., number of shading units) directly with FP32 and chose to exclude transistor counts due to their strong correlation with process size as a downstream effect of smaller transistors. Also they chose a log-linear model

$$\log_{10}(FLOP/s) = \alpha \cdot \log_{10}(X)$$

to ensure that the variation of the feature is over time and not over the feature.

3. Proposed Method

3.1. Data Understanding

We started from the original dataset, which is 4945 rows × 13 columns, covering 2714 GPUs.

The semantics of the attributes are as follows: Power characteristics, such as TDP (thermal design power, measured in watts), indicate the maximum heat output that the cooling system must handle during normal operation. Insights into TDP distribution can reveal market segmentation, from low-power mobile GPUs to high-performance data center accelerators. And physical characteristics, including die size (measured in square millimeters) and transistor count (in millions), represent the silicon chip's area and fundamental building blocks that determine computational

1. https://epoch.ai/blog/predicting-gpu-performanceappendix-c—physical-size-limits-of-transistors-details

2. https://epoch.ai/blog/predicting-gpu-performanceappendix-c—physical-size-limits-of-transistors-details

capability. Frequency, meanwhile, is a critical factor influencing processing speed and overall performance.

For the target variable, GFLOPS (Giga Floating-Point Operations Per Second), we chose it as a key representation of GPU performance. This metric is divided into three precision levels: FP16, FP32, and FP64, each designed for different use cases. Reduced-precision floating-point types, such as FP16 (half-precision) has become increasingly popular in AI applications. [Dongarra et al.(2024)] In contrast to AI, scientific computing often requires high-precision floating-point calculations. Most simulations in fields like molecular dynamics, computational mechanics, and fluid dynamics depend on FP64 precision to ensure the accuracy of their results. [Dongarra et al.(2024)] In comparison, FP32 precision is commonly used in gaming graphics, where visual performance and rendering efficiency take precedence over numerical precision.

	3 precision levels	Use Scenarios
FP16	half precision	AI inference
FP32	single precision	Standard for gaming and general computing
FP64	double precision	scientific computing

Figure 1. GFLOPS precisions

The data reflects large variability, particularly in FP16 performance. The wide spread in the maximum and minimum values highlights differences in GPU capabilities.

	Freq (GHz)	FP16 GFLOPS	FP32 GFLOPS	FP64 GFLOPS
count	4508.000000	800.000000	1685.000000	1278.000000
mean	1615.430790	19033.061063	5403.009359	1096.608263
std	1084.641452	44865.341218	11492.095538	5232.537404
min	100.000000	10.020000	12.800000	3.600000
25%	650.000000	1299.500000	384.000000	59.247500
50%	1400.000000	6136.500000	1248.000000	136.350000
75%	2500.000000	20175.000000	5069.000000	382.450000
max	4700.000000	653700.000000	93240.000000	81720.000000

Figure 2. data understanding.

3.2. Data Preprocessing

Data preprocessing is a crucial step to ensure the dataset is clean, consistent, and suitable for machine learning tasks. The raw dataset contained GPU specifications and performance metrics but also included irrelevant features, missing values, and inconsistent formatting. The preprocessing pipeline was designed to address these issues systematically

	Process Size (nm)	TDP (W)	Die Size (mm^2)	Transistors (million)	Freq (GHz)	FP16 GFLOPS	FP32 GFLOPS	FP64 GFLOPS
3786	28	150	366	5000	920.0	3297.0	3297.0	206.10
3819	28	190	366	5000	918.0	3290.0	3290.0	205.60
3827	28	250	366	5000	723.0	2961.0	2961.0	185.10
3829	28	250	366	5000	850.0	3482.0	3482.0	217.60
3899	28	100	366	5000	723.0	2961.0	2961.0	185.10
...	...	...	...	...	...	...	...	...
4940	8	80	200	8700	1042.0	6021.0	6021.0	94.08
4941	5	220	294	35800	1980.0	35480.0	35480.0	554.40
4942	5	285	379	45900	2340.0	44100.0	44100.0	689.00
4943	5	320	379	45900	2295.0	52220.0	52220.0	816.00
4944	5	285	609	76300	1155.0	71810.0	71810.0	1122.00

658 rows × 8 columns

Figure 3. data preprocessing.

and prepare the data for further analysis. The steps are outlined below:

3.2.1. Removing Non-GPU Data. The dataset originally included both GPU and CPU specifications under the `Type` column. Since this study focuses solely on GPU performance, all rows corresponding to CPUs were removed. This step ensured that the dataset contained only GPU-related records, avoiding contamination from irrelevant hardware types.

3.2.2. Removing Irrelevant Columns. Certain columns, such as `Product`, `Foundry`, `Release Date`, `Vendor`, and `Type`, were not directly related to the predictive task. These columns were dropped to reduce dimensionality and focus on the most relevant features, including hardware specifications and performance metrics.

3.2.3. Handling Missing and Unknown Values. The target variables (`FP16 GFLOPS`, `FP32 GFLOPS`, and `FP64 GFLOPS`) are essential for evaluating GPU performance. Any rows with missing values in these columns were removed to ensure the reliability of the dataset. Additionally, rows containing the placeholder value `'unknown'` in any column were identified and dropped. This step further improved data consistency and eliminated potentially ambiguous entries.

3.2.4. Feature Engineering and Data Type Conversion. Several features required data type conversion and cleaning to ensure compatibility with machine learning models:

- **`TDP (W)`:** Converted to numeric format to accurately represent power consumption values.
- **`Freq (GHz)`:** Converted to numeric format to capture GPU frequency in gigahertz.
- **`Process Size (nm)`:** Initially stored as a string, the numeric values were extracted using regular expressions and converted to a numeric format. This ensured that the feature accurately represented the manufacturing process node size in nanometers.
- **`Die Size (mm`^{2}`)`:** Cleaned and converted from string to numeric format, ensuring consistency in the representation of GPU die size.
- **`Transistors (million)`:** Converted to numeric format to correctly represent the transistor count.

These transformations were essential for ensuring that all numerical features were in a compatible format for machine learning models, facilitating accurate and efficient computation.

3.2.5. Final Dataset. After preprocessing, the dataset consisted of clean and well-structured features, including `TDP (W)`, `Freq (GHz)`, `Process Size (nm)`, `Die Size (mm²)`, `Transistors (million)`, and performance metrics (`FP16 GFLOPS`, `FP32 GFLOPS`, `FP64 GFLOPS`). The resulting dataset was ready for feature selection, model training, and evaluation, as discussed in subsequent sections.

After processing, we keep 658 rows × 8 columns for further analysis.

3.3. Attributes Selection

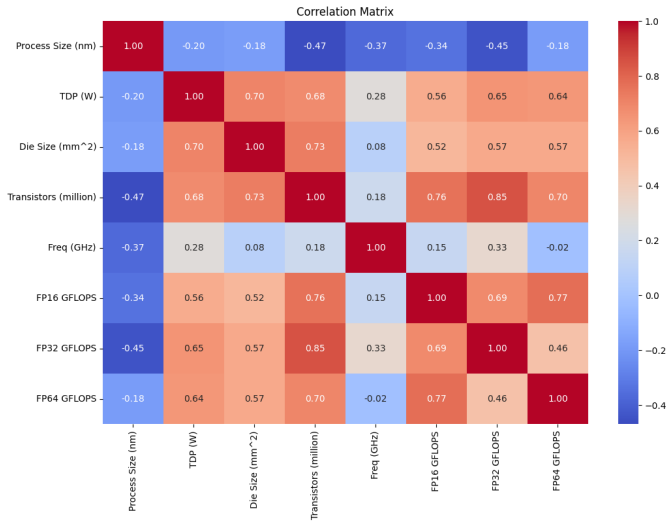

Figure 4. .Correlation matrix

3.3.1. Correlation Matrix. Based on the correlation matrix, the following attributes were selected for prediction: Highly Correlated Features: TDP (W): High correlation with all GFLOPS metrics, reflecting the relationship between power consumption and performance.

Die Size (mm²) and Transistors (million): Both strongly correlated with performance metrics, capturing the impact of hardware complexity and scale.

Excluded Features: Freq (GHz): Low correlation with GFLOPS metrics made it unsuitable for inclusion. Categorical and less relevant features (e.g., Vendor, Foundry, Type, Release Date) were excluded as they provided limited value in predicting floating-point performance.

By retaining only highly correlated attributes, the model could better focus and interpretability, reducing noise from irrelevant data.

3.3.2. Plot Visualization. Here is the 3D scatter plots for TDP (W), Die Size (mm²), and Transistors (million) against the GFLOPS metrics illustrate relationships between features and performance.

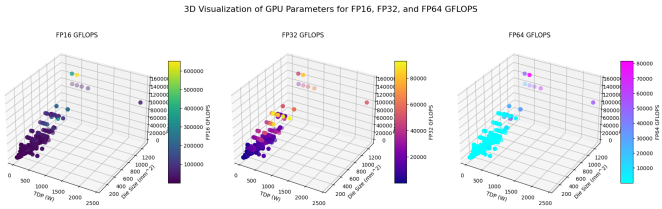

Figure 5. 3D Visualization of GPU Parameters for FP16, FP32, and FP64 GFLOPS.

The visualization revealed distinct patterns in prediction accuracy across different floating-point precision levels: FP32 (Single Precision) showed the most distinctive correlation patterns, clearer color gradients in visualization compared to FP16 and FP64. So in the following model regression we choose FP32 GFLOPS as the target variable.

This may due to both technical architecture and the market reasons: Since the hardware units for FP32 are the "native" processing elements, it has the direct relationship with transistor count. Besides, FP32's maturity in gaming graphics has led to standardized performance requirements.

On the contrary, the performance can be less predictable for FP16 because different AI acceleration architectures and various optimization approaches across vendors.

3.4. Model Selection

So in this study, we aim to predict GPU performance, measured by FP32 GFLOPS, using a set of features selected. Given the complexity of the task and the potential non-linear relationships within the data, we carefully selected three machine learning models: K-Nearest Neighbors (KNN), Random Forest (RF), and XGBoost. These models represent a spectrum of approaches, ranging from simple instance-based learning to sophisticated ensemble and boosting techniques, allowing us to comprehensively evaluate different methodologies.

3.4.1. Model Characteristics and Rationale.

K-Nearest Neighbors (KNN):. KNN is one of the simplest machine learning algorithms. It is based on the idea that similar data points are likely to have similar outcomes. The model makes predictions by identifying the k nearest neighbors to a given input, based on a chosen distance metric (typically Euclidean distance), and computes the average of their target values. The algorithm does not involve a training phase, making it memory-intensive as all training data must be stored for use during prediction.

Despite its simplicity, KNN has several advantages that make it suitable for this study. Firstly, it makes no assumptions about the data distribution, making it well-suited for capturing local, non-linear relationships. Secondly, it serves as an effective baseline model to compare against more complex methods. However, its performance can degrade

with high-dimensional data due to the "curse of dimensionality," and it is sensitive to feature scaling and the choice of k. To mitigate this, normalization was applied to the input features, and k was optimized using grid search.

Random Forest (RF):. Random Forest is an ensemble learning method that builds multiple decision trees during training and aggregates their predictions for regression tasks by averaging. Each tree is trained on a bootstrapped subset of the data, and a random subset of features is considered for each split. This randomization reduces overfitting, making RF a robust model that generalizes well to unseen data.

RF is particularly suited for datasets with complex feature interactions and moderate to high dimensionality. For example, in our GPU dataset, the interplay between transistor count, TDP, and die size could involve non-linear relationships and intricate patterns. Random Forest can effectively capture these without requiring extensive preprocessing. Additionally, it handles missing data and outliers gracefully, providing a robust framework for real-world datasets. However, its interpretability can be limited, as the aggregated predictions obscure the logic of individual trees. To optimize RF, hyperparameters such as the number of trees (`n_estimators`) and maximum depth (`max_depth`) were tuned.

XGBoost:. XGBoost is a state-of-the-art implementation of gradient boosting that combines the power of decision trees with gradient-based optimization. It improves on traditional Gradient Boosting Decision Trees (GBDT) by introducing regularization to control model complexity and prevent overfitting. XGBoost also incorporates efficient data handling techniques, such as sparsity awareness and parallel computing, which accelerate training and improve performance.

XGBoost is particularly suitable for our study due to its ability to model non-linear relationships and capture subtle patterns in the data. The GPU dataset includes features with potentially complex dependencies, such as the influence of process size and transistor count on performance metrics. XGBoost's regularization techniques make it effective for high-dimensional data, while its flexibility allows for fine-tuned optimization of parameters like learning rate, maximum depth, and the number of boosting rounds. However, its high computational cost and sensitivity to hyperparameter settings necessitate careful tuning, which was achieved using grid search and automated tools.

3.4.2. Evaluation Framework. The evaluation framework was designed to assess the predictive performance of each model systematically. Two primary metrics were used: the R^2 score and Mean Squared Error (MSE).

R^2 Score:. The R^2 score, also known as the coefficient of determination, measures the proportion of variance in the target variable explained by the model. An R^2 score of 1 indicates perfect predictions, while a score of 0 suggests that the model performs no better than predicting the mean of the target variable. This metric was chosen to evaluate the models' ability to explain the variability in GPU performance.

Mean Squared Error (MSE):. MSE quantifies the average squared difference between predicted and actual values, providing insight into the model's accuracy. A lower MSE indicates better performance. Unlike R^2, which is scale-free, MSE is in the same unit as the squared target variable, making it a complementary metric for error analysis.

The dataset was split into training (70%) and testing (30%) sets to evaluate the models. Cross-validation was employed on the training set to estimate generalization performance, and normalization was applied to the input features to ensure compatibility with KNN and enhance convergence for XGBoost. Random Forest was robust to the lack of normalization but benefited from cross-validation to fine-tune hyperparameters.

3.4.3. Hyperparameter Tuning. To ensure fair comparisons and optimize model performance, each algorithm underwent hyperparameter tuning:

- **KNN:** The number of neighbors (k) was optimized using grid search over a range of values. The optimal k balanced bias and variance, minimizing MSE on the validation set.
- **Random Forest:** Key parameters, including the number of trees (`n_estimators`) and maximum tree depth (`max_depth`), were tuned. Grid search was used to identify configurations that maximized R^2 while preventing overfitting.
- **XGBoost:** A comprehensive search was performed for parameters such as learning rate (`eta`), maximum depth (`max_depth`), and the number of boosting rounds (`n_estimators`). Automated tools were employed to streamline the tuning process and identify the optimal combination.

By selecting KNN, Random Forest, and XGBoost, we incorporated a diverse range of modeling approaches to predict GPU performance. KNN provides a straightforward baseline, while Random Forest and XGBoost excel in capturing non-linear relationships and complex interactions. This diversity allows for a comprehensive evaluation of the dataset and ensures robustness in identifying the best predictive framework. The results and a detailed comparison of the models are presented in the subsequent sections.

4. Results

In this section, we present the evaluation results of the three models—K-Nearest Neighbors (KNN), Random Forest (RF), and XGBoost—on the task of predicting GPU performance (FP32 GFLOPS). The results are analyzed based on R^2 score, Mean Squared Error (MSE), and visual comparisons between predicted and actual values.

4.1. K-Nearest Neighbors (KNN)

The performance of KNN was optimized by selecting the best number of neighbors (k). As shown in Figure 6,

the R^2 score reaches its peak when $k = 2$. This optimal value was determined using grid search, ensuring a balance between overfitting and underfitting. The final performance metrics for KNN are:

- R^2 Score: 0.9064
- Mean Squared Error (MSE): 16,046,345.18

KNN provided a reasonable baseline performance. However, its relatively lower R^2 score and higher MSE compared to the other models indicate its limitations in capturing the complex, non-linear relationships within the data.

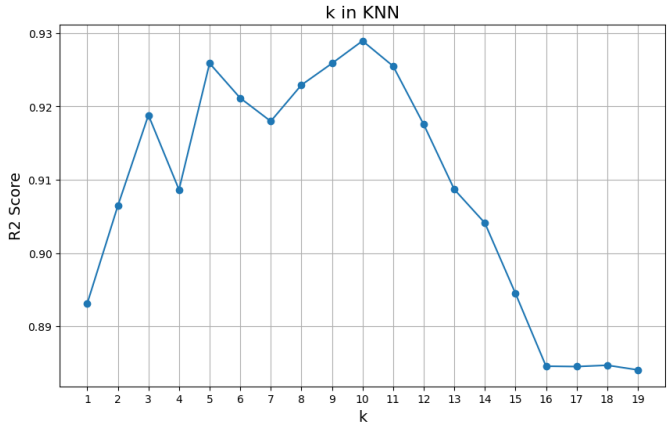

Figure 6. Effect of k on R^2 Score in KNN. The performance peaks at $k = 2$.

4.2. Random Forest (RF)

Random Forest demonstrated superior performance compared to KNN. The ensemble nature of RF allowed it to capture intricate feature interactions and complex patterns in the data. The final performance metrics for RF are:

- R^2 Score: 0.9269
- Mean Squared Error (MSE): 12,535,499.80

This improvement in performance can be attributed to RF's ability to aggregate multiple decision trees, reducing overfitting while maintaining predictive power.

4.3. XGBoost

XGBoost achieved the best overall performance among the three models. Its gradient boosting framework, combined with regularization techniques, allowed it to capture non-linear dependencies and complex relationships in the dataset effectively. The final performance metrics for XGBoost are:

- R^2 Score: 0.9342
- Mean Squared Error (MSE): 11,288,906.16

These results highlight XGBoost's capacity for accurate and robust predictions, making it the most effective model for this task.

4.4. Visual Comparison of Predicted and Actual Values

Figure 7 illustrates a visual comparison of the predicted and actual values across all three models. The XGBoost model's predictions are closest to the actual values, demonstrating its superior accuracy. While RF also performs well, KNN exhibits larger deviations, particularly in extreme cases.

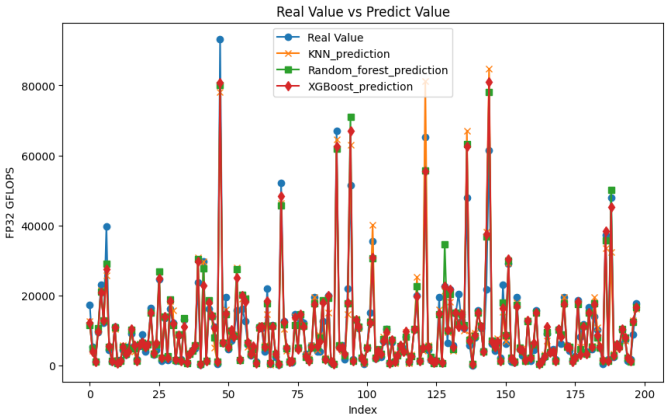

Figure 7. Comparison of Predicted and Actual Values for KNN, Random Forest, and XGBoost. The XGBoost model provides the closest alignment with the real values.

4.4.1. Summary of Results. Table 1 summarizes the performance metrics of the three models. XGBoost outperformed the others in both R^2 score and MSE, followed by Random Forest. KNN, while effective as a baseline, showed limited capability in handling the dataset's complexity.

Model	R^2 Score	MSE
KNN	0.9064	16,046,345.18
Random Forest	0.9269	12,535,499.80
XGBoost	0.9342	11,288,906.16

TABLE 1. PERFORMANCE METRICS FOR KNN, RANDOM FOREST, AND XGBOOST.

5. Conclusion

In this study, we investigated GPU floating-point performance metrics by analyzing hardware specifications and leveraging machine learning techniques. By processing and enriching a comprehensive dataset, we identified key features that significantly influence performance across FP16, FP32, and FP64 precision levels. Our approach, combining predictive modeling with visualization, highlights the relationship between hardware parameters and performance. These insights not only validate the efficacy of regression models like Random Forest and AutoML but also emphasize their potential for guiding future GPU design and optimization efforts.

References

[Dally et al.(2021)] William J Dally, Stephen W Keckler, and David B Kirk. 2021. Evolution of the graphics processing unit (GPU). *IEEE Micro* 41, 6 (2021), 42–51.

[Dongarra et al.(2024)] Jack Dongarra, John Gunnels, Harun Bayraktar, Azzam Haidar, and Dan Ernst. 2024. Hardware Trends Impacting Floating-Point Computations In Scientific Applications. *arXiv preprint arXiv:2411.12090* (2024).

[Sun et al.(2019)] Yifan Sun, Nicolas Bohm Agostini, Shi Dong, and David Kaeli. 2019. Summarizing CPU and GPU design trends with product data. *arXiv preprint arXiv:1911.11313* (2019).