GPU Similarity vs Performance

Aren Desai

2024-03-20

Introduction

Graphics Processing Units (GPUs) are a fundamental part of computer hardware. They are typically tasked with parallel computing for quick matrix calculations, which allows for complex graphics to be rendered quickly. Within the past few decades, it has also been discovered that the parallel computing capabilities of a GPU can also enhance the training and usage of a neural network, which similarly involves a high quantity of matrix calculations. In this report, datasets from alanjo and michaelbryantds are merged by the GPU model and analyzed to determine a few key characteristics revolving around the understanding of what GPUs with a similar architecture have in common in performance. This can be extended with unsupervised learning to cluster GPUs into three categories: low, middling, and high quality.

Dataset

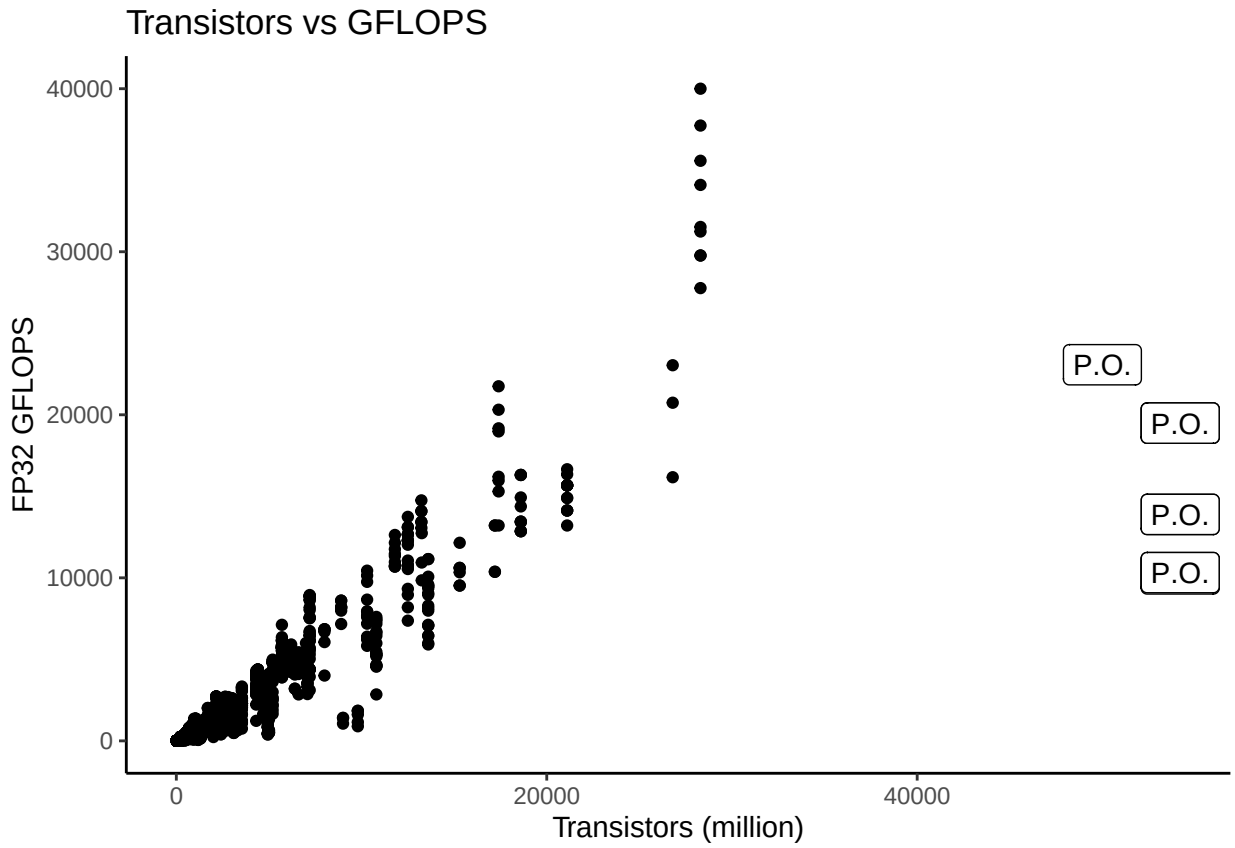
The dataset compiled here is indexed by the names of the models of various GPUs, and includes the memory size, the memory bus width, the gpu clock speed, the memory clock speed, unified shader count, TMU, ROP, the processer size, the die size, the transistor count, the frequency, the g-flops in 16/32/64 bytes, and the manufacturer of the GPU model. All of this is content in the csv file *gpu_data.csv*, where the manufacturer column is removed and the index is replaced with the model name. Basic statistics of the dataset can be seen in Table 1.

Table 1: Summary Statistics

$Feature$	$Mean$	$Median$
$memSize$	2.618	1.024
$memBusWidth$	258	128
$gpuClock$	675	613.5
$memClock$	860.8	800
$unifiedShader$	689.3	240
tmu	46.83	24
rop	18.68	8
$Process\ Size(nm)$	58.48	40
$TDP\ (W)$	82.57	49
$Die\ Size\ (mm^2)$	205.8	148
$Transistors\ (million)$	2541	716
$Freq\ (MHz)$	674.3	620
$FP16\ GFLOPS$	2638.24	403.2
$FP32\ GFLOPS$	1691.59	384
$FP64\ GFLOPS$	206.76	33

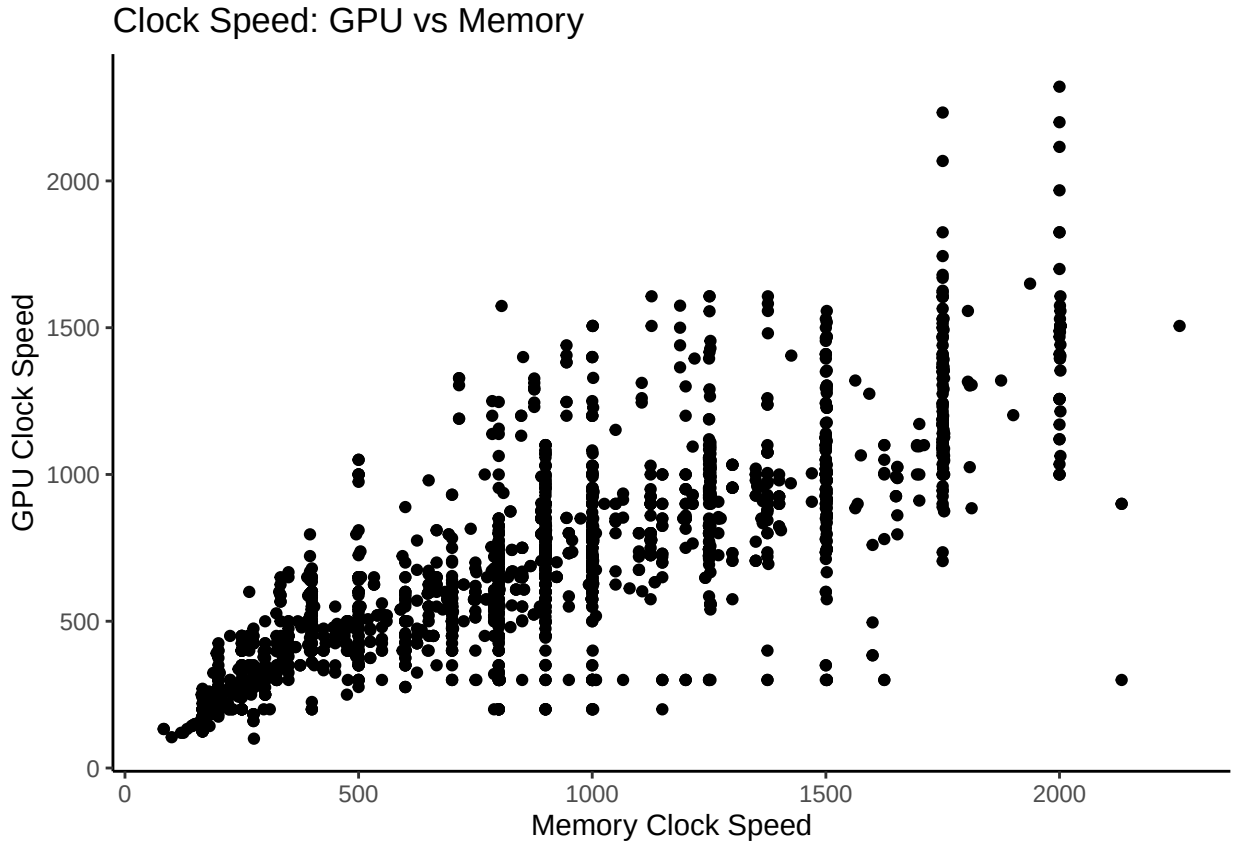
Some meaningful connections can be seen by simple plots. For example, the plot of transistors vs GFLOPS shows a somewhat linear or exponential correlation, with potential outliers labeled as P.O. (potential outlier).

Figure 1: A scatter plot showing the relation between the number of transistors in a GPU and its GFLOPS performance for FP32.



It is clear that a direct upgrade in technology (more transistors, better GPU speed, etc.) indicates a higher quality in other aspects of the GPU. This is to be expected of course, but the linearity of figures 1 and 2 are interesting to note. It appears as a direct correlation with high variance that better GPU architure leads to better performance.

Figure 2: A scatter plot depicting the relation between GPU clock speed and Memory Clock Speed.



Cluster Analysis

In the pursuit of analyzing similarity in GPUs, a question arose whether GPUs could be grouped into categories based on their attributes. K-Means clustering provided a satisfactory result as shown in Figure 3, a K-Means clustering analysis done on the memClock and gpuClock attributes. Three clear distinctions can be observed in the GPUs, each corresponding to lower, middling, and higher tier GPU build qualities.

Figure 3: Figure 2 separated into 3 clusters depicting low, middling, and high quality.

Increasing K-Means algorithm to include all of the features in the dataset would allow a decent method for classifying whether a new GPU introduced into the dataset would be low, middling, or high quality. In order to do this, spectral clustering was applied for easy visual analysis of the clustering.

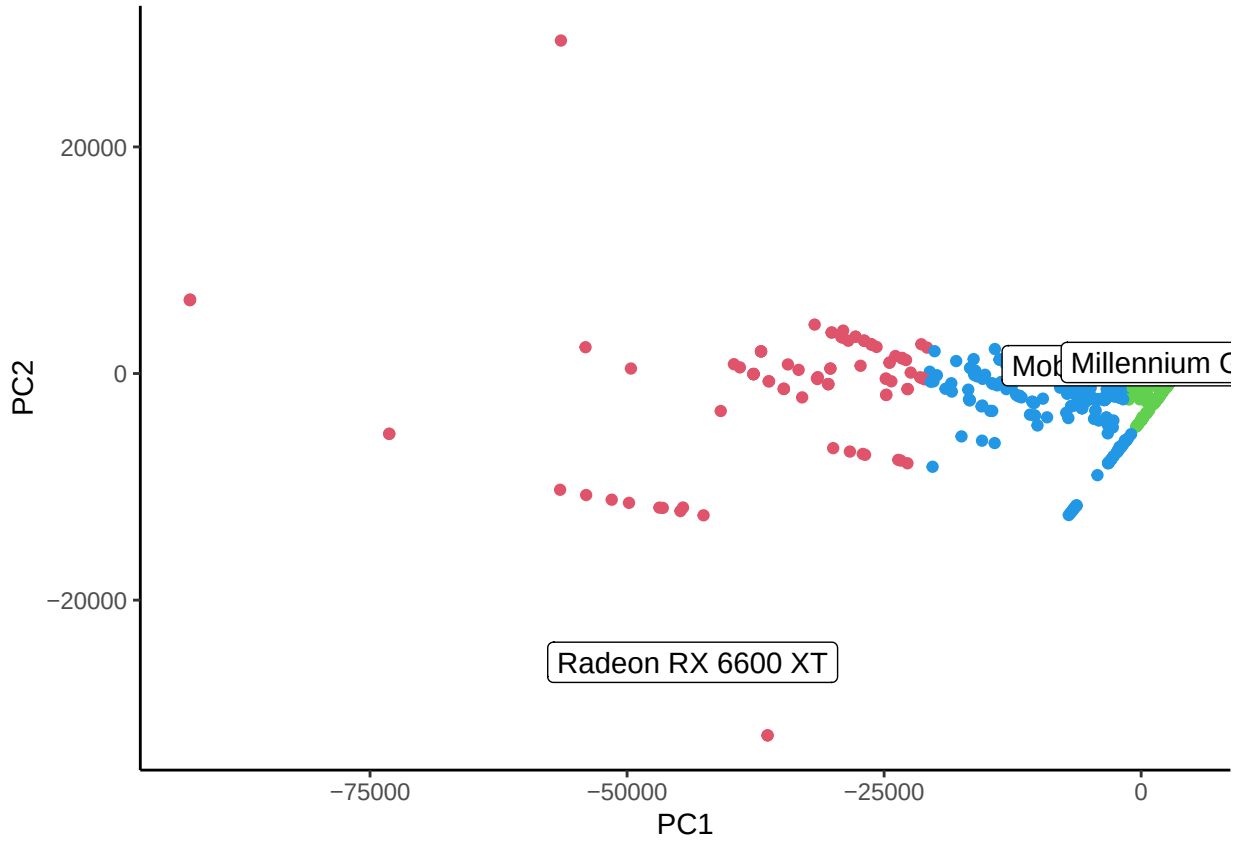
After performing PCA on the GPU dataset, the first two components corresponded to ~0.78 cumulative proportion of the variance. While this is slightly on the low side, it is rather good for simplifying from 15 to 2 features. K-Means clustering with 3 centroids provides Figure 4, which depicts the dataset of GPUs clustered by quality according to all available features.

In order to properly identify the groups, these GPUs were isolated by quality and identified on the graph:

- High: Radeon RX 6600 XT
- Middling: Mobility Radeon HD 5145
- Low: Millennium G450

However, the categories as represented on figure 4 were not as clearly defined as expected. Instead, GPUs like the Radeon RX 6600 XT appear largely dissimilar to the majority of other GPUs, while the Mobility Radeon HD 5145 and Millennium G450 appear close together. Most likely, this graph is capturing the similarities in GPU architecture without as much regard to their performance as expected. In this case, GPUs like the Radeon RX 6600 XT are most likely specially developed GPUs that deviate from typical GPU architecture to enhance their niche case.

Figure 4: A scatter plot depicting three GPU clusters with all features, minimized into two dimensions with PCA



Classification

The GPU clock speed (represented as gpuClock) was set as the target for supervised learning, and the other attributes used as training data for our model. The goal was to predict the GPU clock speed based on those other attributes. In the case that a new GPU is being invented, its GPU clock speed could be predicted with this model before its creation.

Formulaing parameters for a parametric model seemed both difficult and unreasonable, as the GPUs being created don't neccessarily have to follow an underlying distribution (especially with the quick-moving technology sector). Therefore, the non-parametric supervised learning method K-Nearest Neighbors was selected, with k=3. This resulted in a ~99% accuracy for all classes, which can be seen in Table 2. This may be considered overfitting in other contexts, but since there are only three classes being detected with 14 features, this can be considered negligible.

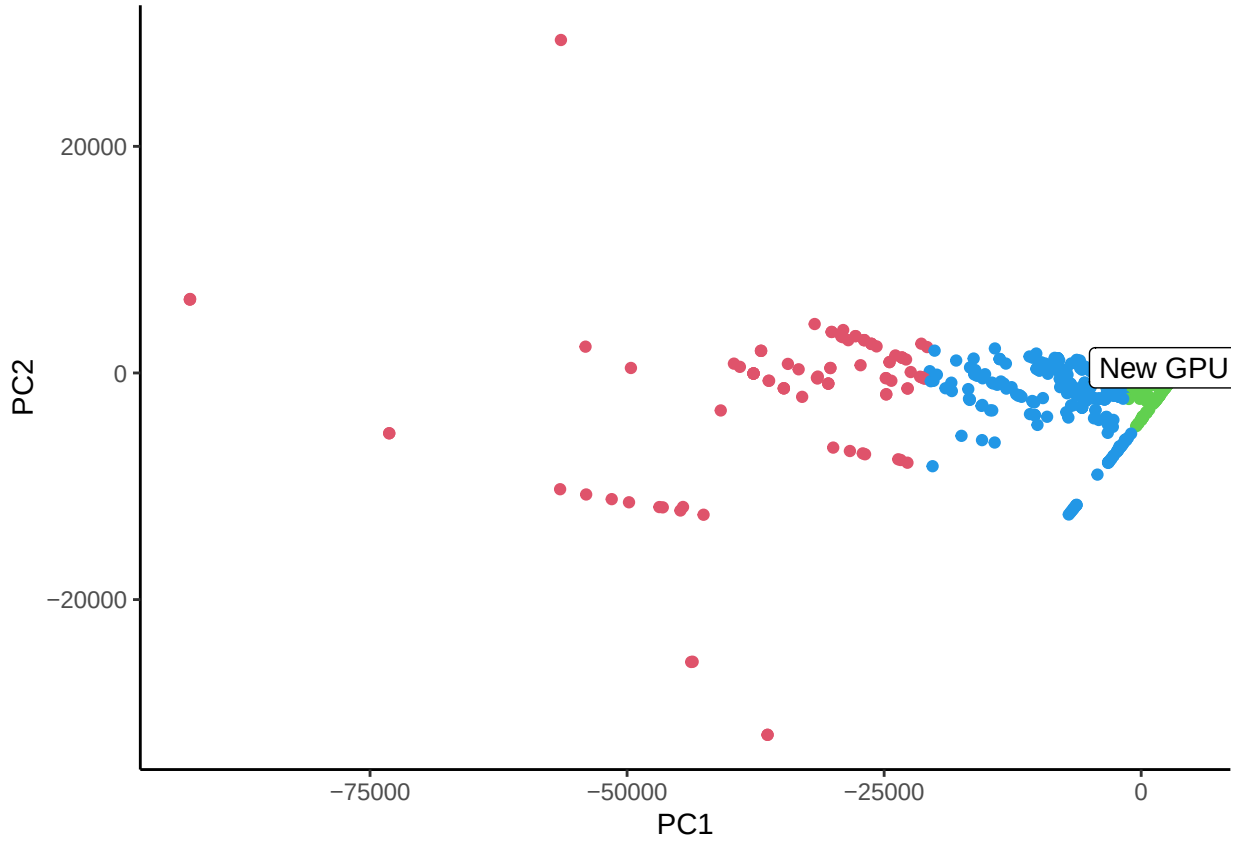
Statistic	*Class* 1	*Class* 2	*Class* 3
Sensititvity	0.9909	1	0.9979
Specificity	0.9980	0.99912	0.9976
Pos Pred Value	0.9880	0.97917	0.9995
Neg Pred Value	0.9985	1	0.9907
Prevalence	0.1404	0.03988	0.8197
Detection Rate	0.1392	0.03988	0.8180
Detection Prevalence	0.1409	0.04073	0.9978
Balanced Accuracy	0.9945	0.99956	0.9978

New GPU Simulation

To conclude this project, we'll assume a new GPU is being created and simulate its quality and specifications. This new GPU will be based on slighly higher values of the median per feature (with some random variance), and will have the following qualities:

$memSize$	1.024
$memBusWidth$	256
$gpuClock$	800
$memClock$	700
$unifiedShader$	345
tmu	38
rop	10
$Process\ Size(nm)$	45
$TDP\ (W)$	50
$Die\ Size\ (mm^2)$	200
$Transistors\ (million)$	1000
$Freq\ (MHz)$	600
$FP16\ GFLOPS$	1200
$FP32\ GFLOPS$	900
$FP64\ GFLOPS$	200

The K-NN categorized this GPU as class 1, and after spectral decomposition, it was placed in the following cluster:



Conclusion

While the methods presented in this project didn't run into any large errors and, honestly, had better results than expected given the heavy data cleaning done to merge the two initial datasets, it doesn't appear great at classifiying GPUs by quality, as the class of the GPU was 1 (which was supposed to indicate low quality) and in the spectral decomposition, the GPU was placed in a high-density area of class 1. However, it appears that the analysis done in this project has revealed a certain "average GPU" architecture, which can identify when a GPU may have been created that deviates from the norm. This is shown in how the GPU prototype achieved a class that places it similar to most other GPUs and in a high-density area, most likely due to its features being around the median.

Appendix

```
library(dplyr)
library(ggplot2)
library(tibble)
library(MASS)
library(dbscan)
library(factoextra)
library(class)
library(caret)

df = read.csv("gpu_data.csv")
dfz = df
df = column_to_rownames(df, "Product")

summary(df)

figure1 = ggplot(df, aes(x = Transistors..million., y = FP32.GFLOPS)) +
  geom_point() +
  theme_classic() +
  xlab('Transistors (million)') +
  ylab('FP32 GFLOPS') +
  ggtitle('Transistors vs GFLOPS') +
  geom_label(data = subset(df, Transistors..million. > 40000), label="P.O.")

figure2 = ggplot(df, aes(y =gpuClock, x = memClock)) +
  geom_point() +
  theme_classic() +
  ylab('GPU Clock Speed') +
  xlab('Memory Clock Speed') +
  ggtitle('Clock Speed: GPU vs Memory')

df1 = data.frame('memClock' = df['memClock'][[1]], 'gpuClock' = df['gpuClock'][[1]])
km.res = kmeans(df1, 3, nstart=25)

figure3 = ggplot(df1, aes(y = gpuClock, x = memClock)) +
  geom_point(col=km.res$cluster+1) +
  theme_classic() +
  ylab("GPU Clock Speed") +
  xlab("Memory Clock Speed") +
```

```
  ggtitle("K-Means Clustering (n=3)")

Sigma = cov(df)
gpu_pcacov = princomp(covmat = Sigma, scores=TRUE)
gpu_scores = data.frame(prcomp(df)$x)
# Eliminating outlier
gpu_scores = gpu_scores %>% filter(PC1 > -100000)
km.res2 = kmeans(gpu_scores, 3, nstart=25)

df2 = dfz %>% filter(Product != 'Radeon Instinct MI100') %>% column_to_rownames("Product")
df2$Cluster = km.res2$cluster

figure4 = ggplot(gpu_scores, aes(x = PC1, y = PC2)) +
  geom_point(col=km.res2$cluster+1) +
  theme_classic() +
  geom_label(data = subset(gpu_scores, row.names(gpu_scores) == "Radeon RX 6600 XT"), label="Radeon RX
  geom_label(data = subset(gpu_scores, row.names(gpu_scores) == "Mobility Radeon HD 5145"), label="Mobi
  geom_label(data = subset(gpu_scores, row.names(gpu_scores) == "Millennium G450"), label="Millennium G

predictors = subset(df2, select = -Cluster)
responses = df2$Cluster
knnres = knn(predictors, predictors, responses, k=3)
conf_matrix = confusionMatrix(knnres, factor(df2$Cluster))

df3 = c(
"memSize" = 1.024,
"memBusWidth" = 256,
"gpuClock" = 800,
"memClock" = 700,
"unifiedShader" = 345,
"tmu" = 38,
"rop" = 10,
"Process.Size..nm." = 45,
"TDP..W." = 50,
"Die.Size..mm.2." = 200,
"Transistors..million." = 1000,
"Freq..MH.)" = 600,
"FP16.GFLOPS" = 1200,
"FP32.GFLOPS" = 900,
"FP64.GFLOPS" = 200
)

knn(predictors, df3, responses, k=3)

df4 = rbind(df, df3)
Sigma = cov(df4)
gpu_pcacov = princomp(covmat = Sigma, scores=TRUE)
gpu_scores = data.frame(prcomp(df4)$x)
# Eliminating outlier
gpu_scores = gpu_scores %>% filter(PC1 > -100000)
km.res2 = kmeans(gpu_scores, 3, nstart=25)

figure5 = ggplot(gpu_scores, aes(x = PC1, y = PC2)) +
```

```
  geom_point(col=km.res2$cluster+1) +
  theme_classic() +
  geom_label(data = subset(gpu_scores, row.names(gpu_scores) == "2359"), label="New GPU")
```