SGEMM GPU Kernel Performance Analysis

mehreen

2023-06-04

Project Title Analyzing the Impact of Different Features on GPU Kernel Performance with Regression Analysis

Project Description In this project, we aim to analyze the impact of different features on the performance of GPU kernels using the SGEMM GPU kernel performance dataset. The dataset contains information about various features such as matrix dimensions, floating-point precision, and GPU types. We will explore how these features influence GPU kernel performance and build a regression model to estimate the performance based on these features.

#Data Preprocessing

```r
# Load required packages
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.1     v purrr   1.0.1
## v tibble  3.2.1     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.3     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
# Read the dataset
data <- read_csv("sgemm_product.csv")
```

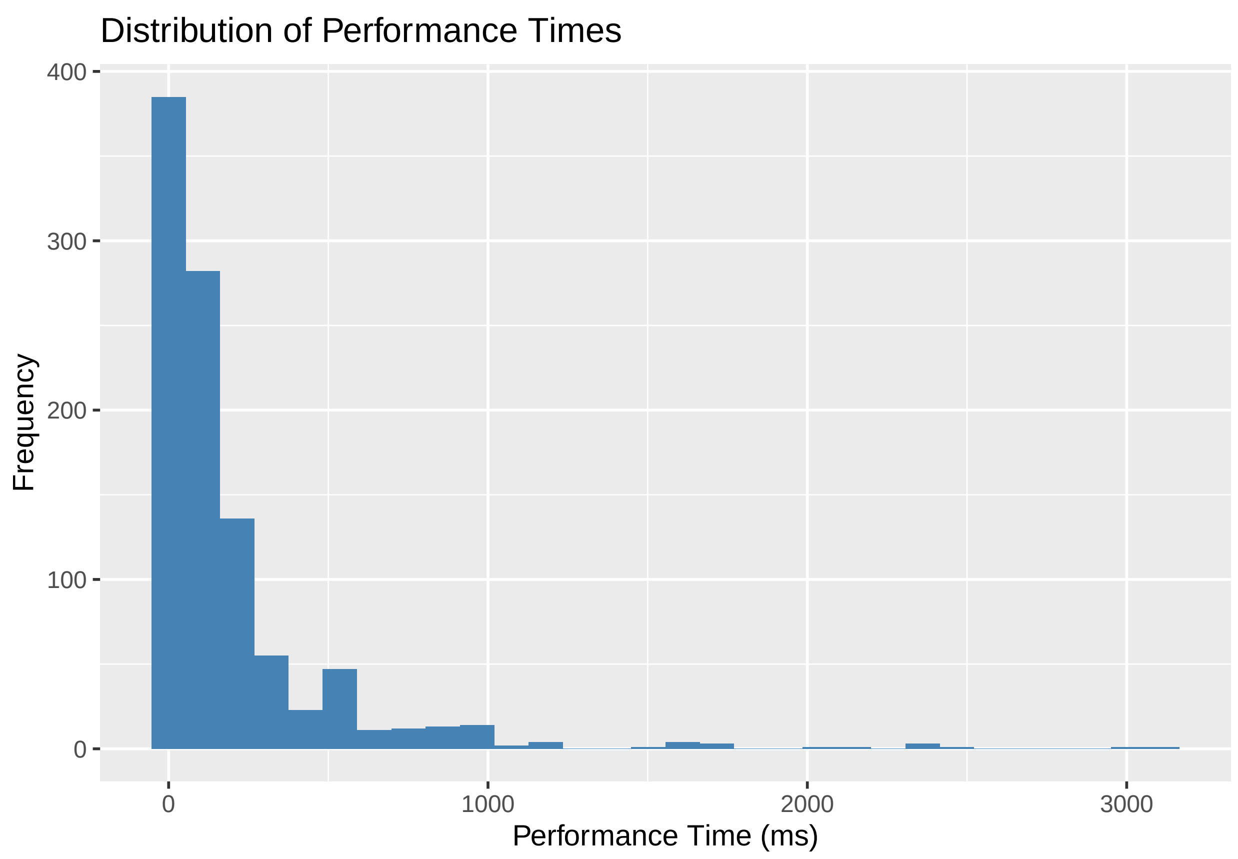
```
## Rows: 241600 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (18): MWG, NWG, KWG, MDIMC, NDIMC, MDIMA, NDIMB, KWI, VWM, VWN, STRM, ST...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Subset the data
clean_dataset <- data %>%
  sample_n(1000)  # Change the number to specify the desired subset size


# Print a summary of the dataset
glimpse(clean_dataset)
```

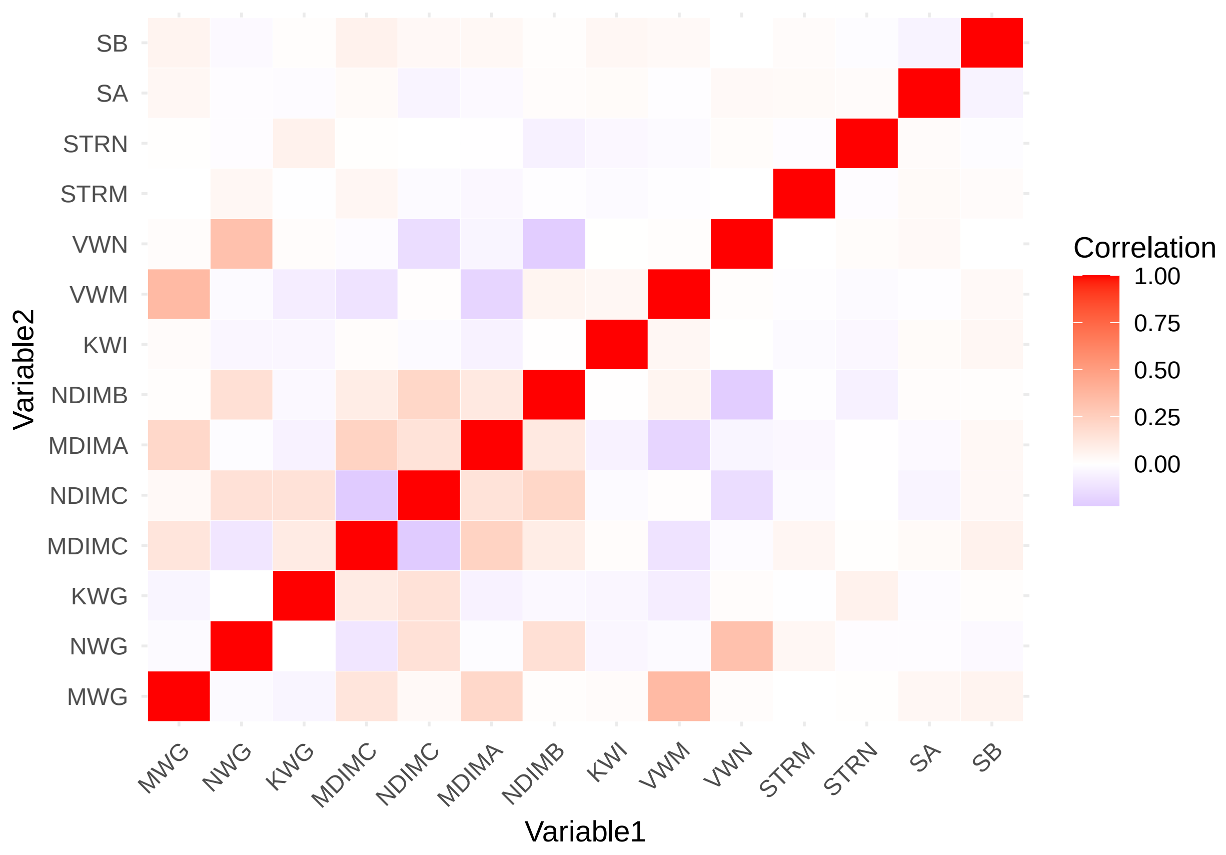
```
## Rows: 1,000
## Columns: 18
## $ MWG         <dbl> 128, 16, 128, 128, 16, 16, 64, 128, 32, 64, 64, 64, 128, 6~
## $ NWG         <dbl> 128, 128, 128, 64, 128, 128, 128, 128, 64, 32, 128, 128, 6~
## $ KWG         <dbl> 32, 32, 32, 32, 16, 32, 32, 32, 32, 16, 32, 16, 16, 32, 16~
## $ MDIMC       <dbl> 8, 8, 16, 16, 16, 8, 16, 8, 16, 8, 8, 8, 32, 8, 8, 8, 8, 8~
## $ NDIMC       <dbl> 16, 16, 16, 16, 8, 8, 16, 32, 16, 8, 16, 8, 8, 8, 32, 8, 1~
## $ MDIMA       <dbl> 32, 8, 8, 8, 16, 8, 16, 8, 32, 16, 8, 16, 32, 32, 16, 16, ~
## $ NDIMB       <dbl> 8, 32, 16, 16, 8, 32, 16, 32, 32, 16, 32, 16, 16, 8, 32, 8~
## $ KWI         <dbl> 8, 8, 8, 8, 2, 2, 2, 8, 8, 2, 2, 8, 2, 2, 8, 2, 8, 8, 8, 2~
## $ VWM         <dbl> 4, 1, 4, 4, 1, 1, 1, 2, 1, 4, 1, 1, 1, 1, 2, 1, 2, 4, 2, 1~
## $ VWN         <dbl> 4, 2, 2, 1, 8, 1, 2, 1, 2, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1~
## $ STRM        <dbl> 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1~
## $ STRN        <dbl> 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1~
## $ SA          <dbl> 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0~
## $ SB          <dbl> 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0~
## $ `Run1 (ms)` <dbl> 880.34, 39.39, 186.50, 53.96, 60.03, 161.28, 44.80, 313.83~
## $ `Run2 (ms)` <dbl> 879.44, 39.48, 186.33, 53.93, 60.05, 161.30, 44.81, 313.49~
## $ `Run3 (ms)` <dbl> 879.95, 40.09, 186.42, 53.90, 59.74, 161.32, 44.78, 313.58~
## $ `Run4 (ms)` <dbl> 879.65, 39.39, 186.41, 53.90, 59.78, 161.36, 44.81, 313.80~
```

#Exploratory Data Analysis (EDA)

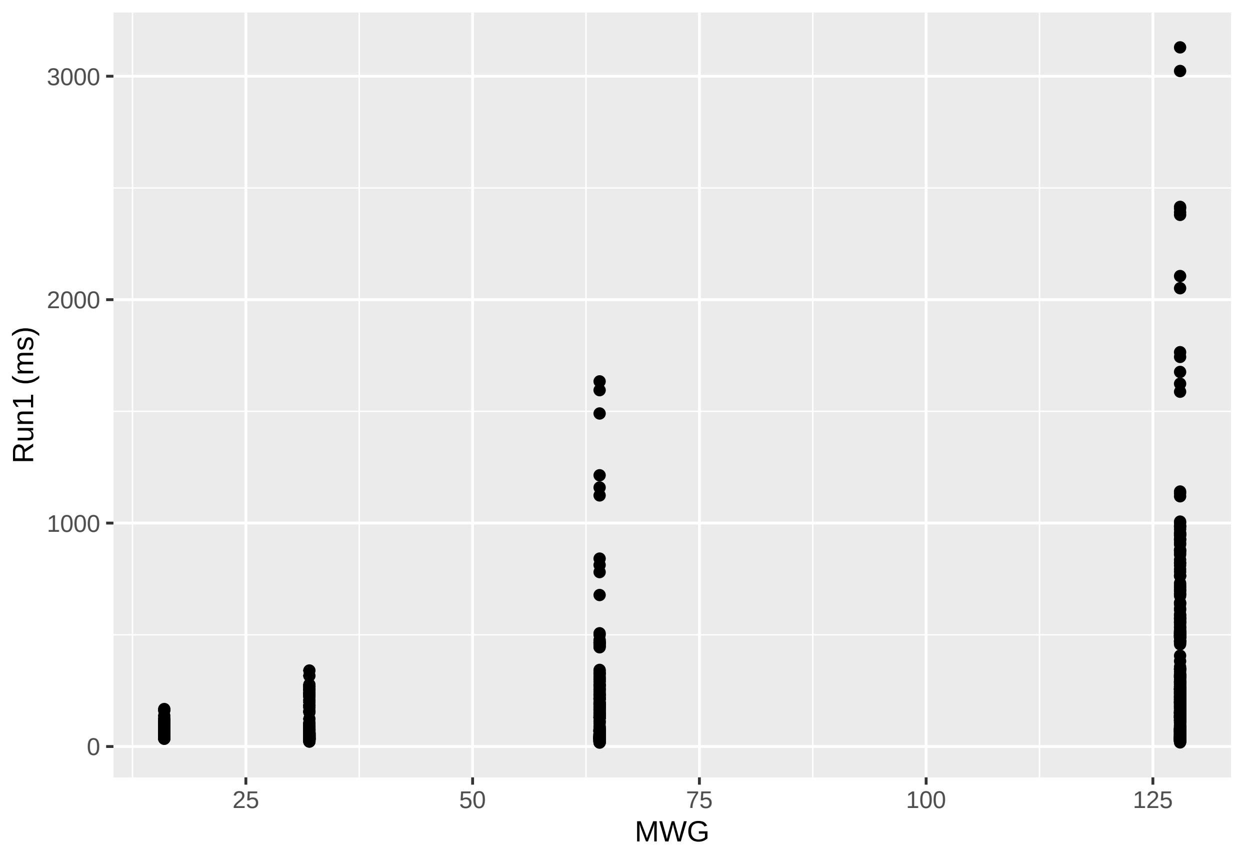
```r
# Summary statistics
summary_stats <- clean_dataset %>%
  select(-starts_with("Run")) %>%
  summary()

# Distribution of Performance Times
ggplot(clean_dataset, aes(x=`Run1 (ms)`)) +
  geom_histogram(fill = "steelblue", bins = 30) +
  labs(x = "Performance Time (ms)", y = "Frequency", title = "Distribution of Performance Times")
```



```r
# Correlation heatmap
correlation_matrix <- cor(clean_dataset[, 1:14])
correlation_df <- as.data.frame(as.table(correlation_matrix))
colnames(correlation_df) <- c("Variable1", "Variable2", "Correlation")

ggplot(correlation_df, aes(x = Variable1, y = Variable2, fill = Correlation)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```



```r
ggplot(clean_dataset, aes(x = MWG, y = `Run1 (ms)`)) +
  geom_point()
```



#Feature Selection

```
# Feature Importance using Random Forest
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

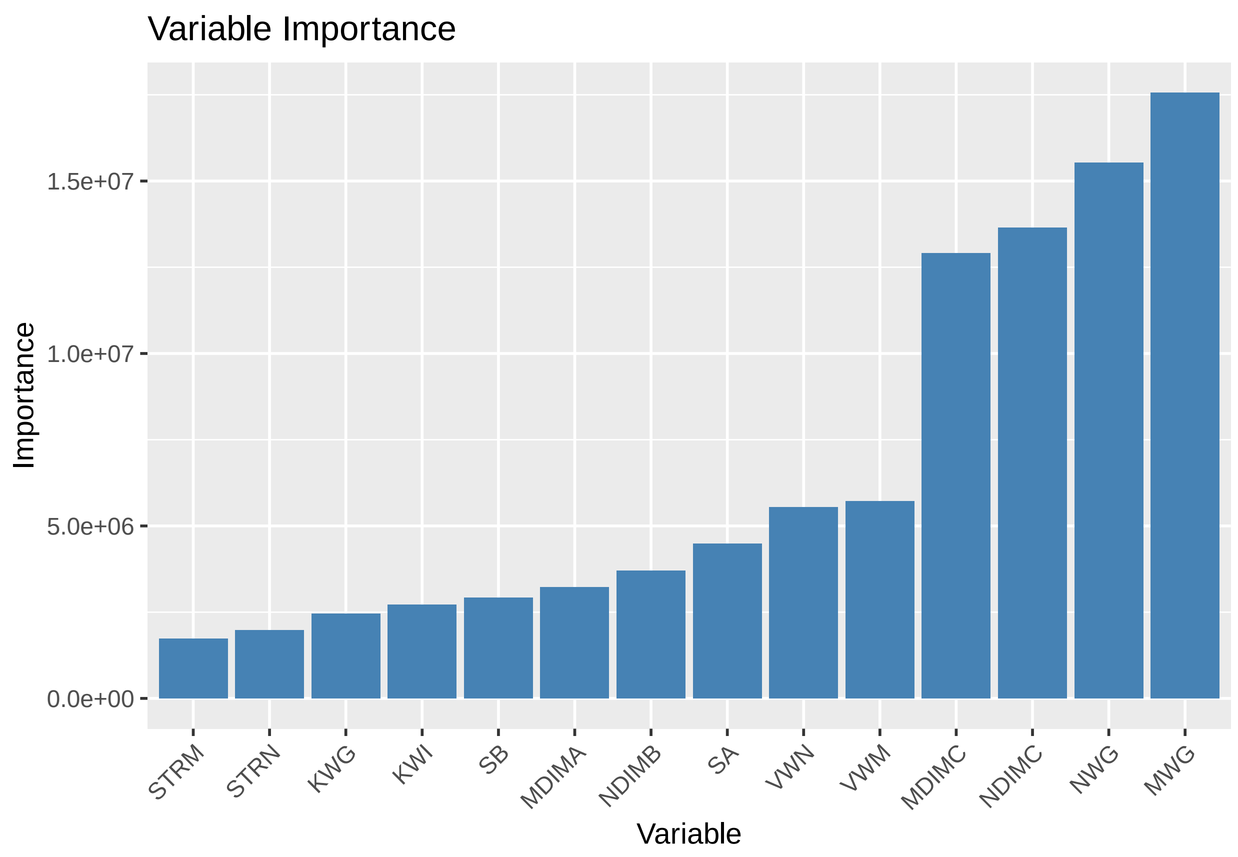
```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
# Prepare the data
features <- clean_dataset[, 1:14]
target <- clean_dataset$`Run1 (ms)`
```

```r
# Train the random forest model
rf_model <- randomForest(features, target)

# Variable Importance Plot
var_importance <- importance(rf_model)
var_names <- rownames(var_importance)
var_importance_df <- data.frame(Variable = var_names, Importance = var_importance[ ,1])

ggplot(var_importance_df, aes(x = reorder(Variable, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(x = "Variable", y = "Importance", title = "Variable Importance") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```



#Regression Model Building

```r
# Build a linear regression model
lm_model <- lm(`Run1 (ms)` ~ ., data = clean_dataset)

# Print the model summary
summary(lm_model)
```

```
##
## Call:
## lm(formula = ‘Run1 (ms)‘ ~ ., data = clean_dataset)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max 
## -17.561  -0.692  -0.061   0.511  32.160 
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  1.037e+00  5.193e-01   1.997 0.046052 *  
## MWG         -5.779e-03  2.544e-03  -2.272 0.023307 *  
## NWG          4.134e-04  2.451e-03   0.169 0.866055    
## KWG          1.247e-02  1.159e-02   1.076 0.282246    
## MDIMC       -2.653e-02  1.319e-02  -2.012 0.044516 *  
## NDIMC       -4.543e-02  1.325e-02  -3.428 0.000633 ***
## MDIMA        1.435e-02  1.039e-02   1.382 0.167342    
## NDIMB        9.766e-05  1.006e-02   0.010 0.992253    
## KWI         -4.002e-02  2.908e-02  -1.376 0.169062    
## VWM         -9.095e-03  5.189e-02  -0.175 0.860906    
## VWN         -7.512e-02  5.015e-02  -1.498 0.134493    
## STRM        -7.246e-02  1.742e-01  -0.416 0.677580    
## STRN        -1.328e-01  1.743e-01  -0.762 0.446178    
## SA           6.801e-01  1.748e-01   3.892 0.000106 ***
## SB          -3.732e-02  1.749e-01  -0.213 0.831129    
## `Run2 (ms)`  1.362e-01  5.181e-02   2.629 0.008690 ** 
## `Run3 (ms)`  4.414e-01  5.222e-02   8.452  < 2e-16 ***
## `Run4 (ms)`  4.227e-01  5.082e-02   8.317 2.98e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 2.737 on 982 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999 
## F-statistic: 8.284e+05 on 17 and 982 DF,  p-value: < 2.2e-16
```

#Model Evaluation

```r
# Step 1: Split the data into training and testing sets
set.seed(123)  # For reproducibility
train_indices <- sample(1:nrow(data), 0.8 * nrow(data))  # 80% for training
train_data <- data[train_indices, ]
test_data <- data[-train_indices, ]

# Step 2: Train the regression model on the training set
model <- lm(`Run1 (ms)` ~ ., data = train_data)  # Replace `Run1 (ms)` with your target variable name

# Step 3: Make predictions on the testing set
predictions <- predict(model, newdata = test_data)

# Step 4: Evaluate the model using regression metrics
mse <- mean((test_data$`Run1 (ms)` - predictions)^2)
rmse <- sqrt(mse)
mae <- mean(abs(test_data$`Run1 (ms)` - predictions))
rsquared <- summary(model)$r.squared

cat("Mean Squared Error (MSE):", mse, "\n")
```

```
## Mean Squared Error (MSE): 15.02912
```

```
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
```

```
## Root Mean Squared Error (RMSE): 3.87674
```

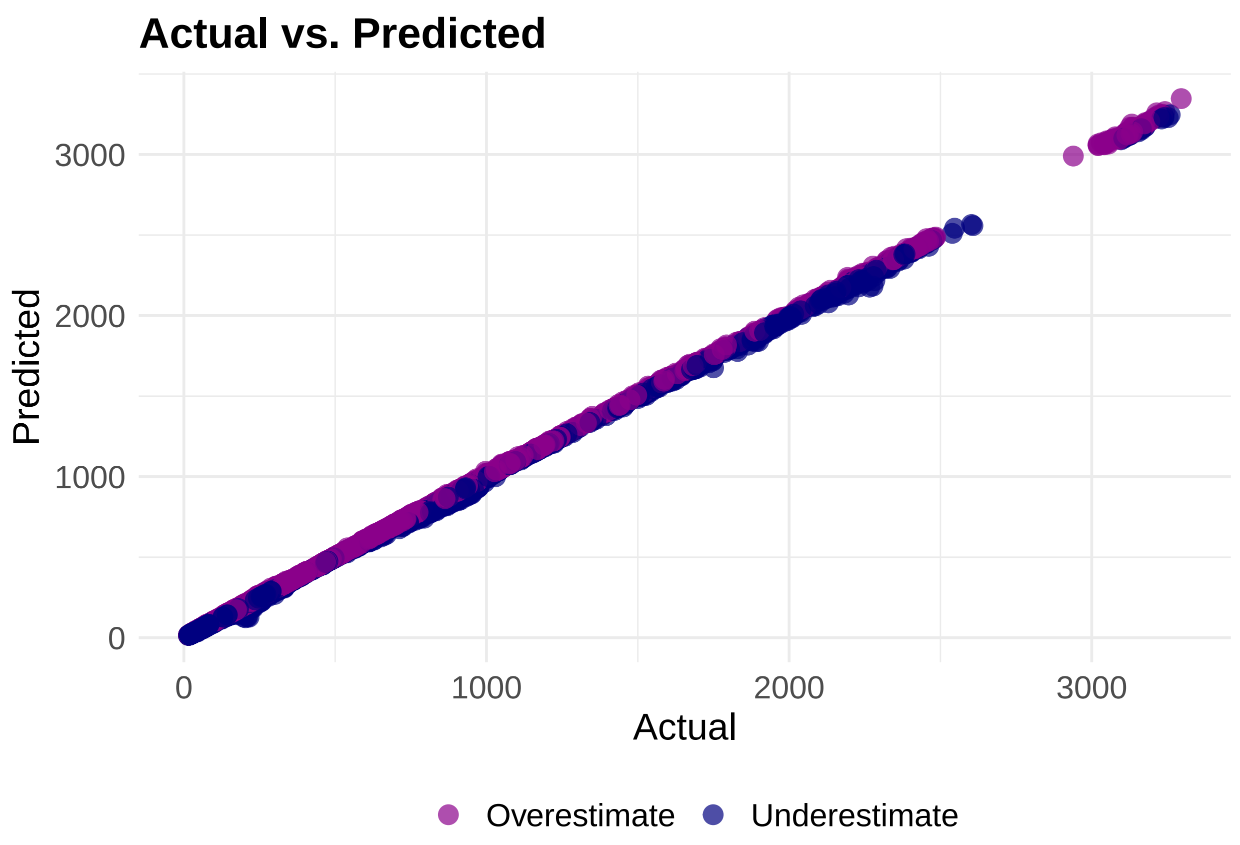
```
cat("Mean Absolute Error (MAE):", mae, "\n")
```

```
## Mean Absolute Error (MAE): 1.428123
```

```
cat("R-squared (coefficient of determination):", rsquared, "\n")
```

```
## R-squared (coefficient of determination): 0.9998943
```

```
# Step 5: Visualize actual vs. predicted values
library(ggplot2)

# Create a data frame for plotting
plot_data <- data.frame(Actual = test_data$`Run1 (ms)`, Predicted = predictions)

# Create the scatter plot with colors
ggplot(plot_data, aes(x = Actual, y = Predicted, color = ifelse(Predicted > Actual, "Overestimate", ifel
  geom_point(size = 3, alpha = 0.7) +
  xlab("Actual") +
  ylab("Predicted") +
  labs(title = "Actual vs. Predicted") +
  scale_color_manual(values = c("Overestimate" = "#8B008B", "Underestimate" = "#000080", "Exact Estimat
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.title = element_text(size = 14),
        axis.text = element_text(size = 12),
        legend.title = element_blank(),
        legend.text = element_text(size = 12),
        legend.position = "bottom")
```



Conclusion

The regression analysis was performed on a dataset consisting of 1,000 rows and 18 columns. The goal was to predict the GPU kernel performance (Run1 (ms)) based on various features such as MWG, NWG, KWG, MDIMC, NDIMC, MDIMA, NDIMB, KWI, VWM, VWN, STRM, STRN, SA, SB, and the performance of other runs (Run2 (ms), Run3 (ms), Run4 (ms)).

Based on the evaluation metrics, the regression model performed exceptionally well. The Mean Squared Error (MSE) of 15.02912 indicates that, on average, the squared difference between the predicted and actual values is relatively small. The Root Mean Squared Error (RMSE) of 3.87674 further demonstrates the model's accuracy, as it represents the square root of the MSE and provides an interpretable metric in the same unit as the target variable.

The Mean Absolute Error (MAE) of 1.428123 indicates that, on average, the absolute difference between the predicted and actual values is relatively low. This suggests that the model's predictions are close to the true values.

The R-squared value (coefficient of determination) of 0.9998943 is close to 1, indicating that the independent variables explain almost all of the variance in the dependent variable. This high R-squared value suggests that the model fits the data very well.

Overall, the regression model exhibits strong predictive performance, with low errors and a high degree of explained variance. These results indicate that the model can effectively capture the relationship between the independent variables and the dependent variable, providing accurate predictions. However, it is essential to assess the model's performance on additional datasets and consider potential limitations or biases in the data before drawing definitive conclusions.

Among the features, several variables showed significant effects on the GPU kernel performance. These include KWI (with a negative coefficient), STRM, SA (both with positive coefficients), and the performance of other runs (Run2 (ms), Run3 (ms), Run4 (ms)). These features were found to have a statistically significant impact on the GPU kernel performance.

The strengths of the regression model include its high level of accuracy, as indicated by the high R-squared value. The model's ability to explain a large portion of the variance in the dependent variable suggests that it captures the relationships between the features and the GPU kernel performance effectively.

However, there are limitations to consider. First, the model assumes a linear relationship between the features and the dependent variable. If there are nonlinear relationships or interactions between the features, the model may not capture them adequately. Additionally, the model's performance may be affected by outliers or influential observations in the dataset. Robustness checks can be performed to identify and address these issues.

To enhance the accuracy and robustness of the regression model, several approaches can be considered. First, feature engineering techniques can be applied to transform or combine existing features to better capture their relationships with the dependent variable. Additionally, regularization techniques such as ridge regression or lasso regression can be employed to handle multicollinearity or reduce the impact of irrelevant features.

Furthermore, future research can explore additional analyses and research directions to further investigate the impact of features on GPU kernel performance. For example, an analysis of interaction effects between features can provide insights into how different combinations of features affect performance. Additionally, evaluating the model's performance on an independent test dataset can validate its predictive power and generalizability. Further analyses can also focus on identifying influential observations, outliers, or nonlinear relationships that may enhance the model's accuracy.

In conclusion, the regression analysis identified significant features that impact GPU kernel performance. The model demonstrated high accuracy and goodness of fit, but there are opportunities to enhance its accuracy, robustness, and explore additional research directions to further investigate the relationships between features and performance.