- Derived Mathematical Calculations for AI Benchmark Dataset
  - Overview
  - Document Information
  - Derived Field Categories
  - 1. Efficiency Metrics
    - 1.1 TOPs_per_Watt
    - 1.2 FLOPS_per_Watt
    - 1.3 GFLOPS_per_Watt
    - 1.4 Performance_per_Dollar_per_Watt
    - 1.5 powerPerformance
  - 2. Performance Scaling Metrics
    - 2.1 Relative_Latency_Index
    - 2.2 Compute_Usage_Percent
  - 3. Latency and Throughput Calculations
    - 3.1 Throughput Calculations for Standard Models
    - 3.2 Avg_Throughput_fps
  - 4. Precision Performance Metrics
    - 4.1 FP16_Performance_Predicted
    - 4.2 INT8_Performance_Estimated
  - 5. Memory and Storage Calculations
    - 5.1 Memory_GB
    - 5.2 MemoryTier
  - 6. Performance Classification Categories
    - 6.1 AI_Efficiency_Tier
    - 6.2 AI_Performance_Category
    - 6.3 PerformanceTier
    - 6.4 Generation and Architecture Categories
      - GenerationCategory
      - PerformanceCategory
  - 7. Value and Economic Metrics
    - 7.1 gpuValue
    - 7.2 PricePerformanceIndex
    - 7.3 IsLegacyLowPerf
  - 8. Architecture-Based Imputation
    - 8.1 Imputed Metrics
  - 9. Data Quality and Coverage
    - Coverage Statistics

    - Quality Assurance
- 10. Usage Guidelines for AI Modeling
    - Recommended Applications
    - Model Input Features
    - Data Limitations
- Conclusion

Derived Mathematical Calculations for AI Benchmark Dataset

Overview

This document provides a comprehensive overview of all derived fields in the AI-Benchmark-Final-enhanced.csv dataset. These fields are calculated using mathematical formulas and algorithms based on existing data sources within the dataset. Each derived field includes its mathematical calculation, description, source fields, and practical interpretation for AI performance prediction modeling.

Document Information

- **Dataset**: AI-Benchmark-Final-enhanced.csv
- **Total Records**: 2,108 GPU devices
- **Original Columns**: 46 base attributes
- **Derived Columns**: 26 calculated metrics
- **Purpose**: AI hardware performance prediction and KPI modeling

Derived Field Categories

The derived fields are organized into the following categories:

1. **Efficiency Metrics** - Power and performance efficiency calculations
2. **Performance Scaling** - Precision and throughput scaling factors
3. **Latency and Throughput** - Performance speed calculations
4. **Memory Utilization** - Memory usage estimation

5. **Categorical Classifications** - Performance tier assignments
6. **Predicted Performance** - Machine learning derived estimates

---

1. Efficiency Metrics

1.1 TOPs_per_Watt

Mathematical Formula:

```
TOPs_per_Watt = (FP32_Final / 1e12) / TDP
```

Description: Tera Operations per Second per Watt - a critical AI efficiency metric measuring computational throughput relative to power consumption.

Source Fields:

- `FP32_Final`: 32-bit floating point performance (FLOP/s)
- `TDP`: Thermal Design Power (Watts)

Units: TOPs/Watt (Tera Operations per Second per Watt)

Interpretation: Higher values indicate better energy efficiency for AI workloads. This metric is crucial for data center deployments where power consumption directly impacts operational costs.

Coverage: Applied to devices with both FP32_Final and TDP data available

---

1.2 FLOPS_per_Watt

Mathematical Formula:

```
FLOPS_per_Watt = FP32_Final / TDP
```

Description: Raw floating-point operations per second per watt of power consumption. This is a fundamental efficiency metric for computational workloads.

Source Fields:

- `FP32_Final`: 32-bit floating point performance (FLOP/s)
- `TDP`: Thermal Design Power (Watts)

Units: FLOPS/Watt

Interpretation: Directly measures computational efficiency. Higher values indicate more compute capability per unit of power consumed.

1.3 GFLOPS_per_Watt

Mathematical Formula:

```
GFLOPS_per_Watt = (FP32_Final / 1e9) / TDP
```

Description: Giga Floating-Point Operations per Second per Watt. A normalized version of FLOPS_per_Watt for easier comparison and interpretation.

Source Fields:

- `FP32_Final`: 32-bit floating point performance (FLOP/s)
- `TDP`: Thermal Design Power (Watts)

Units: GFLOPS/Watt

Derivation Strategy: Convert FLOPS to GFLOPS (divide by 1e9) then normalize by power consumption.

1.4 Performance_per_Dollar_per_Watt

Mathematical Formula:

```
Performance_per_Dollar_per_Watt = (FP32_Final / 1e9) / (price * TDP)
```

Description: Economic efficiency metric combining performance, cost, and power consumption. Measures GFLOPS delivered per dollar invested per watt consumed.

Source Fields:

- `FP32_Final`: 32-bit floating point performance (FLOP/s)
- `price`: Device price in USD
- `TDP`: Thermal Design Power (Watts)

Units: GFLOPS/(USD × Watt)

Interpretation: Higher values indicate better value proposition considering both purchase cost and operational power costs.

1.5 powerPerformance

Mathematical Formula:

```
powerPerformance = G3Dmark / TDP
```

Description: Graphics performance per watt, measuring rendering efficiency relative to power consumption.

Source Fields:

- `G3Dmark`: 3D graphics benchmark score
- `TDP`: Thermal Design Power (Watts)

Units: Score/Watt

Interpretation: Particularly relevant for graphics-intensive AI applications such as computer vision and rendering workloads.

---

2. Performance Scaling Metrics

2.1 Relative_Latency_Index

Mathematical Formula:

```
Relative_Latency_Index = max(FP32_Final) / FP32_Final
```

Description: Normalized latency estimation using inverse relationship with FLOPS performance. Higher computational performance typically correlates with lower inference latency.

Source Fields:

- `FP32_Final`: 32-bit floating point performance (FLOP/s)

Units: Dimensionless ratio

Derivation Strategy:

- Calculate maximum FLOPS across all devices
- Divide maximum by each device's FLOPS
- Results in relative latency where 1.0 represents the fastest device

Interpretation: Lower values indicate faster (lower latency) performance. A value of 2.0 suggests approximately twice the latency of the fastest device.

2.2 Compute_Usage_Percent

Mathematical Formula:

```
Compute_Usage_Percent = (powerPerformance / max(powerPerformance)) × 100
```

Description: Estimated compute utilization percentage based on power efficiency normalization.

Source Fields:

- `powerPerformance`: Graphics performance per watt

Units: Percentage (%)

Derivation Strategy:

- Normalize powerPerformance by maximum value in dataset
- Scale to percentage (0-100%)
- Represents relative compute unit utilization

Interpretation: Higher percentages indicate more efficient utilization of available compute resources.

---

3. Latency and Throughput Calculations

3.1 Throughput Calculations for Standard Models

Mathematical Formula:

```
Throughput_ModelName_fps = FP32_Final / Model_FLOPS_per_inference
```

Standard Model Complexities:

- `Throughput_ResNet50_ImageNet_fps`: FP32_Final / 4.1e9
- `Throughput_BERT_Base_fps`: FP32_Final / 22.5e9
- `Throughput_GPT2_Small_fps`: FP32_Final / 1.5e9
- `Throughput_MobileNetV2_fps`: FP32_Final / 0.3e9
- `Throughput_EfficientNet_B0_fps`: FP32_Final / 0.39e9

Description: Estimated inference throughput (frames/inferences per second) for standard AI models based on their computational complexity.

Source Fields:

- `FP32_Final`: 32-bit floating point performance (FLOP/s)

Units: Frames per second (fps) or Inferences per second

Model Complexity Constants:

- ResNet50 (ImageNet): 4.1 GFLOPS per inference
- BERT Base: 22.5 GFLOPS per inference

- GPT-2 Small: 1.5 GFLOPS per inference
- MobileNetV2: 0.3 GFLOPS per inference
- EfficientNet-B0: 0.39 GFLOPS per inference

3.2 Avg_Throughput_fps

Mathematical Formula:

```
Avg_Throughput_fps = mean(Throughput_ResNet50_fps, Throughput_BERT_fps,
                          Throughput_GPT2_fps, Throughput_MobileNetV2_fps,
                          Throughput_EfficientNet_fps)
```

Description: Average throughput across all standard model benchmarks, providing a generalized performance indicator.

Source Fields:

- All individual throughput calculations

Units: Average frames per second (fps)

---

4. Precision Performance Metrics

4.1 FP16_Performance_Predicted

Mathematical Formula:

```
FP16_Performance_Predicted = LinearRegression(FP32_Final).predict()
```

Description: Machine learning predicted FP16 (half-precision) performance based on FP32 performance using linear regression.

Source Fields:

- `FP32_Final`: 32-bit floating point performance (FLOP/s)

- `FP16 (half precision) performance (FLOP/s)`: Training data for model

Units: FLOP/s (half-precision)

Derivation Strategy:

1. Train linear regression model on devices with both FP32 and FP16 data
2. Validate model with $R^2 > 0.3$ threshold
3. Predict missing FP16 values using FP32 performance
4. Model equation: FP16_pred = a × FP32_Final + b

Model Quality: R^2 coefficient reported for validation

4.2 INT8_Performance_Estimated

Mathematical Formula:

```
INT8_Performance_Estimated = FP32_Final × Architecture_Speedup_Factor
```

Architecture-Specific Speedup Factors:

- Turing: 4.0× (Tensor cores)
- Ampere: 5.0× (Improved tensor cores)
- Ada Lovelace: 6.0× (Latest tensor cores)
- RDNA 2: 2.0× (AMD INT8 support)
- RDNA: 1.8× (Limited INT8)
- GCN: 1.5× (Basic INT8)
- Pascal: 2.0× (DP4A instruction)
- Maxwell: 1.2× (Limited INT8)
- Unknown: 2.0× (Conservative estimate)

Description: Estimated INT8 (8-bit integer) performance based on architecture-specific acceleration capabilities.

Source Fields:

- `FP32_Final`: 32-bit floating point performance (FLOP/s)
- `Architecture`: GPU architecture family

Units: Operations per second (OP/s)

---

5. Memory and Storage Calculations

5.1 Memory_GB

Mathematical Formula:

```
Memory_GB = Memory size per board (Byte) / 1e9
```

Description: GPU memory capacity converted from bytes to gigabytes for easier interpretation.

Source Fields:

- `Memory size per board (Byte)`: Memory capacity in bytes

Units: Gigabytes (GB)

Conversion Factor: 1 GB = 1e9 bytes

5.2 MemoryTier

Mathematical Formula:

```
MemoryTier = categorical_bins(Memory_GB)
```

Category Ranges:

- "Minimal (<4GB)": Memory_GB < 4
- "Low (4-8GB)": 4 ≤ Memory_GB < 8
- "Medium (8-16GB)": 8 ≤ Memory_GB < 16
- "High (16-24GB)": 16 ≤ Memory_GB < 24
- "Ultra (24GB+)": Memory_GB ≥ 24
- "Unknown": Missing data

Description: Categorical classification of memory capacity for tier-based analysis.

Source Fields:

- `Memory_GB`: GPU memory in gigabytes

---

6. Performance Classification Categories

6.1 AI_Efficiency_Tier

Mathematical Formula:

```
AI_Efficiency_Tier = categorical_bins(TOPs_per_Watt)
```

Tier Ranges:

- "Entry": 0 < TOPs_per_Watt ≤ 0.01
- "Mid-Range": 0.01 < TOPs_per_Watt ≤ 0.03
- "High-End": 0.03 < TOPs_per_Watt ≤ 0.06
- "Premium": 0.06 < TOPs_per_Watt ≤ 0.1
- "Ultra": TOPs_per_Watt > 0.1

Description: AI efficiency classification based on TOPs per Watt performance.

Source Fields:

- `TOPs_per_Watt`: Calculated efficiency metric

6.2 AI_Performance_Category

Mathematical Formula:

```
AI_Performance_Category = conditional_classification(FP32_Final, TOPs_per_Watt)
```

Classification Conditions:

- "AI_Flagship": FP32_Final ≥ 20e12 AND TOPs_per_Watt ≥ 0.05
- "AI_High_End": FP32_Final ≥ 15e12 AND TOPs_per_Watt ≥ 0.03
- "AI_Mid_Range": FP32_Final ≥ 10e12 AND TOPs_per_Watt ≥ 0.02
- "AI_Entry": FP32_Final ≥ 5e12 AND TOPs_per_Watt ≥ 0.01
- "AI_Basic": FP32_Final ≥ 1e12
- "AI_Legacy": FP32_Final < 1e12

Description: Comprehensive AI performance classification combining raw performance and efficiency.

Source Fields:

- `FP32_Final`: 32-bit floating point performance (FLOP/s)
- `TOPs_per_Watt`: AI efficiency metric

6.3 PerformanceTier

Mathematical Formula:

```
PerformanceTier = categorical_classification(G3Dmark)
```

Description: Graphics performance tier based on 3DMark benchmark scores.

Source Fields:

- `G3Dmark`: 3D graphics benchmark score

Categories: "Entry-Level", "Mid-Range", "High-End", "Flagship"

6.4 Generation and Architecture Categories

GenerationCategory

Formula:

```
GenerationCategory = year_range_classification(testDate)
```

Categories:

- "Legacy (<2016)": testDate < 2016
- "Older Gen (2016-2017)": 2016 ≤ testDate < 2018
- "Previous Gen (2018-2019)": 2018 ≤ testDate < 2020
- "Recent Gen (2020-2021)": 2020 ≤ testDate < 2022
- "Current Gen (2022+)": testDate ≥ 2022

PerformanceCategory

Formula:

```
PerformanceCategory = performance_tier_classification(G3Dmark)
```

Categories: Based on G3DMark score percentiles

---

7. Value and Economic Metrics

7.1 gpuValue

Mathematical Formula:

```
gpuValue = G3Dmark / price
```

Description: Performance value per dollar - graphics performance relative to purchase price.

Source Fields:

- `G3Dmark`: 3D graphics benchmark score
- `price`: Device price in USD

Units: Score per USD

Interpretation: Higher values indicate better price-performance ratio for graphics workloads.

7.2 PricePerformanceIndex

Mathematical Formula:

```
PricePerformanceIndex = gpuValue
```

Description: Alias for gpuValue, representing the price-performance index.

Source Fields:

- `gpuValue`: Calculated performance per dollar

Units: Score per USD

7.3 IsLegacyLowPerf

Mathematical Formula:

```
IsLegacyLowPerf = (Generation == "Legacy (<2016)") AND (PerformanceTier == "Entry-Level")
```

Description: Boolean flag identifying legacy, low-performance devices.

Source Fields:

- `Generation`: Device generation category
- `PerformanceTier`: Performance classification

Units: Boolean (True/False)

---

8. Architecture-Based Imputation

8.1 Imputed Metrics

Mathematical Formula:

```
Field_Imputed = median(Field, groupby=Architecture)
```

Description: Missing values filled using architecture-specific median values.

Imputed Fields:

- `TDP_Imputed`: Missing TDP values
- `FP32_Final_Imputed`: Missing FLOPS values
- `FLOPS_per_Watt_Imputed`: Missing efficiency values

Source Fields:

- Original field values
- `Architecture`: GPU architecture family

Strategy:

1. Group devices by architecture
2. Calculate median for each metric within architecture groups
3. Fill missing values with architecture-specific medians
4. Only apply when architecture has sufficient data (≥5 devices)

---

9. Data Quality and Coverage

Coverage Statistics

- **TOPs_per_Watt**: ~95% coverage (2,000+ devices)
- **Throughput Metrics**: ~95% coverage (calculated from FP32_Final)
- **Efficiency Metrics**: ~90% coverage (dependent on TDP availability)
- **Predicted Metrics**: Variable based on training data quality
- **Classification Categories**: ~100% coverage for categorizable fields

Quality Assurance

- Linear regression models validated with $R^2 > 0.3$ threshold
- Architecture-based estimates use proven speedup factors from literature
- Outlier detection applied to prevent unrealistic derived values
- Missing value imputation only applied with sufficient group sample sizes

---

10. Usage Guidelines for AI Modeling

Recommended Applications

1. **Performance Prediction Models**: Use derived throughput and efficiency metrics
2. **Hardware Selection Optimization**: Leverage tier classifications and value metrics
3. **Power Budget Planning**: Utilize TOPs_per_Watt and efficiency calculations
4. **Precision Optimization**: Apply FP16/INT8 predictions for mixed-precision modeling
5. **Economic Analysis**: Incorporate cost-performance metrics for ROI calculations

Model Input Features

Primary derived features for machine learning models:

- `TOPs_per_Watt` (efficiency)
- `Relative_Latency_Index` (speed)
- `Avg_Throughput_fps` (general performance)
- `AI_Performance_Category` (classification)
- `GFLOPS_per_Watt` (computational efficiency)

Data Limitations

- Derived metrics inherit uncertainty from source data quality
- Predicted values should be validated against actual measurements when available

- Architecture-based estimates may not account for driver optimizations
- Throughput calculations assume ideal conditions without memory bottlenecks

---

Conclusion

This documentation provides a complete reference for all 26 derived fields in the AI-Benchmark-Final-enhanced.csv dataset. These calculated metrics enable comprehensive analysis of AI hardware performance, efficiency, and value propositions. The mathematical formulations ensure reproducibility and transparency in the data enhancement process, supporting robust AI performance prediction modeling and hardware selection optimization.

For technical implementation details, refer to the source ETL scripts in the `scripts/utils/` directory, specifically `derive_missing_metrics.py` and `create_final_enhanced_matrix.py`.