Contribution ID: **46** Type: **Poster**

# Performance Comparison of WebAssembly and JavaScript

*Thursday 24 April 2025 11:30 (30 minutes)*

JavaScript remains the dominant language for client-side scripting, while WebAssembly offers near-native execution speeds, making it a compelling choice for computationally intensive tasks. This study provides a comprehensive analysis of the performance differences between WebAssembly and JavaScript across various computing environments, including different browsers (Firefox, Chrome, Edge) and platforms (desktop and mobile) [1], [2].

To evaluate computational efficiency, we conducted a series of benchmark tests, including integer operations (Sieve of Eratosthenes, sorting algorithms) [3], floating-point calculations (numerical integration, Monte Carlo method) [4], and recursive computations (Fibonacci sequence, matrix multiplication) [5]. Additionally, we investigated the impact of WebAssembly on machine learning workloads by utilizing minimalist implementations such as TinyDNN for digit classification on the MNIST dataset [6]. Our findings indicate that WebAssembly consistently outperforms JavaScript in CPU-bound tasks, particularly in integer operations and recursive computations. However, JavaScript's just-in-time (JIT) compilation allows it to remain competitive in some floating-point calculations.

One focus of our study was the application of WebAssembly in browser-based machine learning. We examined the performance of lightweight neural network implementations, emphasizing WebAssembly's ability to accelerate tensor computations directly in the browser. This capability is crucial for deploying AI models on the client side, reducing reliance on cloud-based services, improving privacy, and minimizing latency. Our analysis also explores WebAssembly's potential for real-time applications such as image classification, object detection, and natural language processing.

Beyond raw performance metrics, this study assesses the broader implications of WebAssembly's adoption in web development. One of its key advantages is its ability to support multiple programming languages, including Rust, C, and C++, allowing developers to leverage high-performance libraries within the browser environment [7]. Additionally, WebAssembly's sandboxing mechanisms enhance security by isolating execution, reducing potential attack vectors compared to traditional JavaScript-based applications.

Despite its advantages, WebAssembly has limitations. Execution performance varies across browsers, and its integration with JavaScript-based applications presents challenges due to data serialization overhead. Moreover, WebAssembly lacks direct access to the DOM, necessitating JavaScript as an intermediary for UI interactions.

As WebAssembly continues to evolve, its role in performance critical web applications is expected to expand, particularly in fields such as cryptography, data processing, and real-time

machine learning.
Keywords—WebAssembly, JavaScript, Performance Compar
ison, Machine Learning, Convolutional Neural Networks.

**Authors:** CISZEWSKI, Jakub (PW EE); MYK, Kaja

**Presenters:** CISZEWSKI, Jakub (PW EE); MYK, Kaja

**Session Classification:** Session C (Poster)