

PERFORMANCE COMPARISON OF PROGRAMMING LANGUAGES FOR ARTIFICIAL INTELLIGENCE: PYTHON, JULIA, R

Lazar Kopanja¹

Abstract: The integration of artificial intelligence (AI) technologies across various domains has become a critical requirement in the current stage of societal development. Complex AI algorithms are being developed using advanced programming languages that enable efficient data analysis and scalable solutions. Among the most prominent languages in this field are Python, Julia, and R, each with its unique advantages and limitations. This paper analyzes the performance, ease of use, and richness of AI libraries offered by these languages. The goal of this study is to identify the most suitable language for different tasks. The conclusions are drawn based on technical specifications, practical examples, and findings from previous research.

Keywords: Artificial Intelligence, Python, Julia, R, programming.

1. INTRODUCTION

The development of artificial intelligence represents one of the most significant technological achievements of the modern era. AI is applied in various domains, including medicine, autonomous systems, education, commerce and marketing, entertainment, speech recognition, natural language processing, data analytics and other areas. The appropriate programming language enables researchers and engineers to efficiently implement complex AI algorithms, manage large datasets, and develop scalable applications. Therefore, the primary challenge in the development of these systems is the selection of a programming language that meets the optimization requirements and ensures high-performance applications [1].

Languages that stand out in this field include Python, Julia, and R. Python is a versatile language with a broad range of applications, Julia is specialized for performance and efficiency, while R is most commonly used in statistics and analytics.

The selection of a programming language in AI projects is influenced by several factors, including performance, ease of implementation, and the availability of specialized libraries. Wijek and Smolka [2] created algorithms in each language studied, working on the same test set and using algorithms from the same group. Their results included the running time of the program, the number of lines of code, and the accuracy of the trained model.

Python has become the language of choice for many AI projects due to its rich ecosystem of libraries such as TensorFlow, PyTorch, and scikit-learn [3]. These libraries facilitate the easy implementation and training of models. On the other hand, Julia, although a relatively new language, achieves exceptional performance through Just-In-Time (JIT) compilation, bringing it closer in efficiency to languages like C and Fortran [4].

In contrast, the R programming language is widely utilized in statistics, data analysis, and visualization, supported by specialized libraries such as caret and mlr3 [5]. However, its application in the field of deep learning is more limited compared to Python and Julia.

Previous research has focused on individual aspects of these languages, such as performance in specific domains or qualitative comparisons of their library ecosystems. For instance, Diehl et al. [6] assessed the capabilities of AI models in generating scientific programs and examined the performance of the resulting code in terms of compilation, runtime efficiency, and accuracy. Lubin et al. [7] explored how Julia can be utilized for implementing software and algorithms fundamental to

¹PhD, The Higher Education Technical School of Professional Studies in Novi Sad, Novi Sad, Serbia, Školska 1, email: kopanja.l@vtsns.edu.rs

operations research, with a focus on mathematical optimization. The authors demonstrate algebraic modeling for linear and nonlinear optimization and provide a partial implementation of a practical simplex code. Their benchmarks suggest that Julia is capable of achieving state-of-the-art performance. However, a comprehensive approach that combines high performance, ease of use and specialized libraries remains underexplored.

This paper aims to contribute to the field by providing a comprehensive comparison of the programming languages Python, Julia, and R, evaluated across the three aforementioned dimensions. The study draws upon contemporary research and experimental results to identify specific scenarios where each language demonstrates its strengths or reveals certain limitations. Based on the analysis, we will propose guidelines for selecting the programming language that best aligns with the specific requirements of an AI project.

2. OVERVIEW OF PROGRAMMING LANGUAGES

2.1 Python

Python is a high-level programming language characterized by its simple syntax and code readability. Developed in 1991, it has since become one of the most popular choices for artificial intelligence (AI) and machine learning (ML). Python's intuitive syntax facilitates quick adoption and application, even for beginners, making it an ideal choice for academic projects and research. Additionally, Python boasts a rich ecosystem of libraries. Libraries such as TensorFlow, PyTorch, and scikit-learn offer comprehensive solutions for various AI tasks, including data processing and deep learning [8]. In terms of integration, Python seamlessly connects with other programming languages and tools, including C++ and Fortran, thereby compensating for its relatively slower execution speed. While Python is not among the fastest languages, its flexibility and vast user community often offset this limitation. Harris et al. [9] explored the fundamental concepts, implementations, and advantages of the NumPy library, highlighting its role in enabling efficient data processing and scientific computations in Python, with significant applications in both research and industry.

2.2 Julia

Julia is an open-source programming language developed in 2012, designed to combine the performance of languages like C++ with the flexibility of high-level programming languages. Krishna et al. [10] evaluated the performance of Julia in comparison with C and Python for tasks such as one-dimensional vector operations, matrix multiplication, and solving two-dimensional equations, highlighting Julia's superior performance, especially in computationally intensive tasks, and its ability to effectively utilize parallelism, particularly in multi-core environments. Thanks to Just-In-Time (JIT) compilation, Julia is often faster than Python and can be comparable to compiled languages such as C and C++. Julia is specifically designed for working with large numbers and numerical algorithms, making it ideal for simulations and optimization problems. Julia has built-in support for parallel processing, which allows for better utilization of modern processors [10].

Recent studies focus on analyzing Julia's performance in the context of numerical simulations and algorithm implementation in real-time, with a particular emphasis on its application in engineering and scientific fields [11]. However, the small user community limits the availability of resources and practical examples.

2.3 R

R is a programming language and environment specialized for statistical analysis and data visualization. Developed in 1995, it has become the standard in the field of statistics and analytics.

R provides a rich ecosystem of libraries for statistical analysis, including caret, h2o, and mlr3, which are highly beneficial for AI tasks [12]. Data visualization is one of the language's strongest features, facilitating the creation of complex visualizations with ease. Additionally, tools such as SparkR enable the processing of large datasets efficiently [13]. Although R is less optimized for speed compared to Python and Julia, its contribution to the fields of statistics and data processing remains indispensable.

3. COMPARISON OF PROGRAMMING LANGUAGES

3.1 Performance

The performance of a programming language directly impacts the execution speed of algorithms and the ability to handle large datasets. Krishna et al. [10] showed that Julia delivers the best performance, often surpassing Python, especially in computationally intensive tasks like one-dimensional vector operations and matrix multiplication. The comparison of performance demonstrated Julia's ability to effectively leverage parallel processing. Julia's strong performance in computational tasks highlights its suitability for computational science and statistical analysis.

In contrast, Python, although slower, compensates for its performance with optimized libraries such as NumPy and TensorFlow [14]. Chard et al. [15] emphasized that Python provides strong support for parallelism. The authors introduce Parsl, a parallel programming library that enhances Python with simple, scalable, and flexible constructs for parallelism. These constructs enable Parsl to dynamically build a dependency graph of components, which can then be efficiently executed on one or more processors. Parsl is designed to scale, with a set of extensible executors tailored to different scenarios, including low latency, high throughput, and extreme scalability. They conducted experiments on the Blue Waters supercomputer revealed that Parsl allows Python scripts to execute components with minimal overhead of only 5 milliseconds, scaling up to over 250,000 workers across more than 8,000 nodes and processing more than 1,200 tasks per second. These results suggest that Parsl can support a wide range of applications, including those that require high efficiency, such as interactive applications, online systems, and machine learning, across various scientific disciplines.

While Python and Julia offer excellent support for parallelism, the programming language R significantly lags behind in handling complex data structures.

The performance of programming languages in practical applications is heavily dependent on the specific problem at hand. For instance, Julia excels in optimization tasks and simulations, delivering optimal performance, while Python tends to outperform other languages in domains such as natural language processing and deep learning-based computation.

3.2 Ease of use

Ease of use is critical for the rapid implementation of solutions and the efficient learning of a programming language. Python is widely preferred by researchers and engineers due to its intuitive syntax and extensive educational materials, making it an accessible and efficient tool for various computational tasks. Python's compatibility with platforms like Jupyter Notebook makes it particularly well-suited for educational and research environments. In comparison, although Julia offers high performance, its more intricate syntax and smaller user base may pose difficulties for those just starting out. In contrast, R is particularly tailored for statistical analysis, which can make it challenging for learners unfamiliar with statistics. However, its specialized libraries make it a powerful tool for data visualization.

3.3 Libraries for AI

The availability of libraries directly influences the choice of programming language for AI projects. Python offers a wide range of libraries, including TensorFlow, PyTorch, and scikit-learn, which are considered standards for machine learning and deep learning [3]. El Aoun et al. [16] observed an increasing trend in the usage of Python's deep learning libraries.

Julia, while still a relatively new language, has developed a rapidly expanding ecosystem, with libraries like Flux.jl and MLJ.jl that provide impressive performance and user-friendly features [2]. It is often highlighted that Julia excels particularly in model optimization and large dataset analysis. In contrast, R remains the dominant tool in statistical analysis, supported by libraries like caret and mlr3, although it is less frequently applied in deep learning and real-time scenarios [5], [12].

4. DISCUSSION

While Python, Julia, and R serve different purposes, the decision of which one to use depends on the specific problem at hand and the user's requirements. Python is ideal for research that demands flexibility, scalability, and a vast selection of available resources. Julia is recommended for applications where performance is paramount, such as large-scale data processing and simulations. R continues to be the preferred tool for statistical analysis and advanced data visualization tasks [10],[17]-[21].

Bezanson et al. [17] highlights that Julia is designed for high-performance technical computing, enabling efficient data processing and simulations. The paper [18] emphasizes Julia's suitability for developing machine learning algorithms, balancing efficiency and simplicity, making it ideal for applications in this field. Godoy et al. [19] compares the performance of Julia, Python, and Numba, demonstrating that Julia offers competitive performance compared to C/OpenMP on CPUs and CUDA/HIP on GPUs, which is crucial for applications requiring high performance. The study [10] shows that Julia often outperforms Python, especially in tasks demanding computational power, such as matrix and vector operations.

Nordmann et al. [20] discussed the advantages of using R for data visualization, noting that it benefits from the same advantages as statistical analysis when writing code rather than using point-and-click software. Ayyagari et al. [21] examines Python's integration with cloud platforms, highlighting its performance and efficiency in scalable data processing.

5. CONCLUSION

Python, Julia, and R represent three significant programming languages for artificial intelligence, each with its unique advantages and limitations. Python dominates due to its flexibility, community support, and extensive array of tools, while Julia excels in performance for numerical computations and parallel applications. R, although limited in terms of performance, remains essential for statistical analysis and data visualization.

Based on the analysis, Python is recommended for most applications, particularly those requiring rapid implementation and a broad range of tools. Julia is the best choice for projects where performance is critical, especially in scientific and technical domains. R is ideal for data analysis and projects focused on statistics.

Future research should focus on integrating these languages to leverage their strengths. For instance, using Python for data preprocessing, Julia for simulations, and R for result analysis can provide an optimal solution for complex AI projects.

6. REFERENCES

- [1] Rao VS, Satish MA, Prasad MB. Artificial intelligence: Principles and applications. Leilani Katie Publication; 2024.
- [2] Wiekak T, Smółka J. Performance of machine learning tools. Comparative analysis of libraries in interpreted and compiled programming languages. Journal of Computer Sciences Institute. 2024;33:339-45.
- [3] Liu YH. Python Machine Learning by Example: Build Intelligent Systems Using Python, TensorFlow 2, PyTorch, and Scikit-Learn. Packt Publishing Ltd; 2020.
- [4] Verdugo F, Badia S. The software design of Gridap: a finite element package based on the Julia JIT compiler. Computer Physics Communications. 2022;276:108341.
- [5] Bischl B, Sonabend R, Kotthoff L, Lang M, editors. Applied machine learning using mlr3 in R. CRC Press; 2024.
- [6] Diehl P, Nader N, Brandt S, Kaiser H. Evaluating AI-generated code for C++, Fortran, Go, Java, Julia, Matlab, Python, R, and Rust. arXiv preprint arXiv:2405.13101. 2024.
- [7] Lubin M., Dunning I. Computing in Operations Research using Julia. arXiv preprint arXiv:1312.1431. 2013.
- [8] Lamaazi H, Mathew E. Comprehensive comparative analysis of artificial intelligence, machine learning, and deep learning. In Artificial Intelligence for Intelligent Systems 2024 (pp. 51-68). CRC Press.
- [9] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). <https://doi.org/10.1038/s41586-020-2649-2>
- [10] Krishna SV, Jadhav O, Ardhapurkar P, Valmiki M, Agrawal S, Bulusu R, Dinde P, Wandhekar S. Performance Comparison of Julia with C and Python for Solving Computational Problems. In International Conference on Advances in Computing and Data Sciences 2024 May 9 (pp. 35-45). Cham: Springer Nature Switzerland.
- [11] Rackauckas C, Gwozdz M, Jain A, Ma Y, Martinuzzi F, Rajput U, Saba E, Shah VB, Anantharaman R, Edelman A, Gowda S. Composing modeling and simulation with machine learning in Julia. In 2022 Annual Modeling and Simulation Conference (ANNSIM) 2022 Jul 18 (pp. 1-17). IEEE.
- [12] Park Y. Automated machine learning with R: AutoML tools for beginners in clinical research. Journal of Minimally Invasive Surgery. 2024 Sep 15;27(3):129.
- [13] Ryu WS. Distributed Processing of Big Data Analysis based on R using SparkR. The Journal of the Korea institute of electronic communication sciences. 2022;17(1):161-6.
- [14] Yapıcı MM, Topaloğlu N. Performance comparison of deep learning frameworks. Computers and Informatics. 2021;1(1):1-11.
- [15] Chard K, Babuji Y, Woodard A, Clifford B, Li Z, Hategan M, Foster I, Wilde M, Katz DS. Productive Parallel Programming with Parsl. ACM SIGAda Ada Letters. 2021 Apr 27;40(2):73-5.
- [16] El Aoun MD, Tidjon LN, Rombaut B, Khomh F, Hassan AE. An Empirical Study of Library Usage and Dependency in Deep Learning Frameworks. arXiv:2211.15733. 2022.
- [17] Bezanson J, Karpinski S, Shah VB, Edelman A. Julia: A Fast Dynamic Language for Technical Computing. arXiv:1209.5145. 2012.

- [18] Gao K, Mei G, Piccialli F, Cuomo S, Tu J, Zenan Huo. Julia Language in Machine Learning: Algorithms, Applications, and Open Issues. arXiv:2003.10146. 2020.
- [19] Godoy WF, Valero-Lara P, Dettling TE, Trefftz C, Jorquera I, Sheehy T, Miller RG, Gonzalez-Tallada M, Vetter JS, Churavy V. Evaluating performance and portability of high-level programming models: Julia, Python/Numba, and Kokkos on exascale nodes. arXiv:2303.06195. 2023.
- [20] Nordmann E, McAleer P, Toivo W, Paterson H, DeBruine LM. Data Visualization Using R for Researchers Who Do Not Use R. *Advances in Methods and Practices in Psychological Science*. 2022;5(2). doi:10.1177/25152459221074654
- [21] Ayyagari A, Jain A, Goel O. Utilizing Python for Scalable Data Processing in Cloud Environments. *DIRA* [Internet]. 2024;12(2):183-98. DOI: <https://doi.org/10.36676/dira.v12.i2.78>