JOIV : Int. J. Inform. Visualization, 8(1) - March 2024 434-441

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Programming Language Selection for the Development of Deep Learning Library

Oktavia Citra Resmi Rachmawati [a,*], Aliridho Barakbah [a], Tita Karlita [a]

[a] *Department of Informatics and Computer Engineering, Electronic Engineering Polytechnic Institute of Surabaya, Surabaya, Indonesia*
Corresponding author: [*]oktaviacitra@pasca.student.pens.ac.id

***Abstract*—Recently, deep learning has become very successful in various applications, leading to an increasing need for software tools to keep up with the rapid pace of innovation in deep learning research. As a result, we suggested the development of a software library related to deep learning that would be useful for researchers and practitioners in academia and industry for their research endeavors. The programming language is the core of deep learning library development, so this paper describes the selection stage to find the most suitable programming language for developing a deep learning library based on two criteria, including coverage on many projects and the ability to handle high-dimensional array processing. We addressed the comparison of programming languages with two approaches. First, we looked for the most demanding programming languages for AI Jobs by conducting a data-driven approach against the data gathered from several Job-Hunting Platforms. Then, we found the findings that imply Python, C++, and Java as the top three. After that, we compared the three most widely used programming languages by calculating interval time to three different programs that contain an array of exploitation processes. Based on the result of the experiments that were executed in the computer terminal, Java outperformed Python and C++ in two of the three experiments conducted with 5,4047 milliseconds faster than C++ and 231,1639 milliseconds faster than Python to run quick sort algorithm for arrays that contain 100.000 integer values.**

***Keywords*— Programming language; data-driven approach; software development life cycle; software library; deep learning.**

Manuscript received 12 Dec. 2023; revised 8 Jan. 2024; accepted 10 Feb. 2024. Date of publication 31 Mar. 2024.

I. INTRODUCTION

Deep learning is extensively utilized to extract information from vast data in various fields, such as natural language comprehension, healthcare, autonomous vehicles, and others [1]. Its popularity is mainly due to its exceptional performance, which is made possible by large datasets and robust computing infrastructure [1]. To maintain the swift progress of deep learning exploration, reliable software tools are crucial to allow deep learning researchers to concentrate on their studies while eliminating the need to create neural network functions from scratch. Numerous commonly used frameworks, such as PyTorch [2] and Tensorflow [3], generate a fixed dataflow diagram that represents the computation and can be employed repeatedly to process groups of data [4].

The existing deep learning libraries have been too complex for some machine learning projects, making it difficult to modify the formula in code function. Thus, this will result from the lack of required usefulness to researchers' desire to change the contents of some functions with the specific formulas to improve the model's performance.

Thus, we propose to create a software library that fulfills the requirements of intelligent tools. This library would provide advantages to researchers and professionals in both academic and industrial settings, allowing them to implement deep learning algorithms practically into their codes by modifying the formula they want. Therefore, this can occur when a project requires specialized functionalities not readily available in existing software libraries. A specialized software library dedicated to the execution of deep learning algorithms will serve as a submodule within the Analytical Library Intelligent-computing (ALI), complementing the other modules for machine learning algorithms such as Hierarchical K-Means [5], Automatic Clustering [6], K Nearest Neighbors [7], and Neural Networks [8].

However, it is essential to carefully consider the potential costs and benefits of developing a new software library [9]. The choice of the programming language used to develop a toolkit can affect its efficiency and functionality, particularly if modifications to its internal components are expected.

Additionally, the programming language employed can affect the range of languages suitable for development [10]. Instead of, many toolkits provide alternative mechanisms or bindings that enable access from different programming languages.

This study addresses the considerations for choosing a programming language to develop a software library for deep learning algorithms. Besides finding insights into the most used programming languages by AI practitioners in the industry, we also investigate the speed comparison of programming language selection in processing and computing data. This paper contributes to data visualization as the most used programming language to build AI systems in the industry to identify the tech stack trends and the popular requirements of job shares. Also, this paper contributes to the comparison analysis of the top 3 most used programming languages in AI fields based on speed time execution by a few structured experiments conducted in the same device.

Therefore, this study ensures that a new software library related to deep learning can be produced with a reasonable resource and identifies potential risks that may arise during development. Ultimately, this paper explains a critical step in the software product development process to create a shared understanding of the programming language selection background.

In the remainder of this paper, the following section describes the material and method implemented in this study. Section 3 discusses the fundamental findings based on the experiments. Finally, Section 4 concludes the paper.

II. Material and Method

In this section, we present the methodology used in this study as a qualitative technique with several stages, including study literature, data-driven approach, experiments by SDLC, and paper documentation.

Fig. 1 Research Methodology

Then, we describe the definition of each existing phase in the following:

A. Study Literature

In academic research, a literature review is an organized and transparent process that aims to identify, assess, and combine the existing corpus of scholarly work and other relevant sources conducted by researchers, scholars, and professionals [11]. The process is intended to be systematic and reproducible to ensure validity and reliability [11]. Thus, we reviewed the relevant literature from the perspective of a deep learning library released and distributed openly via internet resources to compare programming language utilization with the aim of state-of-the-art for this paper.

TSFEDL [12] is a deep learning library based on Python programming language that seamlessly integrates the algorithms into advanced machine learning pipelines that gather 22 state-of-the-art methods combining convolutional and recurrent layers. The implementation is dependent on both the Keras functional API and PyTorch-Lightning. TSFEDL offers high-performance neural networks that can be easily extended and customized, demonstrating its value to users. The outcomes of the analysis substantiate that the models included in the library can be effectively deployed across diverse tasks, broadening the potential applications of this model type. Consequently, this Python module represents a practical and user-friendly option for practitioners. We can install the TSFEDL library through PyPi by issuing the command "pip install TSFEDL". Additionally, we can obtain it by cloning the library's repository from GitHub and executing the command "python setup.py install" from the root directory. Upon completion of the installation process, we can utilize the package under the name "TSFEDL." Notably, the library code complies with the PEP8 style standard for Python.

Anomalib [13] is a deep learning library based on Python programming language that offers a collection of instruments that enable users to compare a range of anomaly detection models swiftly and reliably on arbitrary datasets such as training, benchmarking, deploying, and developing anomaly detection models. Anomalib comprises a range of cutting-edge algorithms for detecting and localizing anomalies alongside a collection of modular elements that we can utilize to develop customized algorithms. Additionally, Anomalib employs several utility and helper modules to simplify the entire training and inference pipeline, such as Callbacks, Metrics, and Logging. Meanwhile, Anomalib also provides pre-processing tools that involve applying transformations to input images before training and optionally dividing the images into tiles, either overlapping or non-overlapping.

$TSFE_{DL}$ and Anomalib haven't yet described the decisive reason for Python programming language utilization in their developed library. Otherwise, we clarify the details of the programming language selection that can be a good fit with our aims to create a software library for deep learning. Furthermore, we created Table I below, which contains the research position to simplify comparative analysis between software libraries for deep learning that have published scientific papers to share their objectives.

TABLE I
Research Position

Name	Programming Language	Functionality
$TSFE_{DL}$	Python	a wide variety of customizable CNN-RNN models
Anomalib	Python	a complete collection of deep learning-based anomaly detection
Analytical Library Intelligent computing	Java	The primary module is to solve machine learning tasks optimally without supercomputing.

B. Data-Driven Approach

Software engineers have employed data-driven approaches, such as software analytics, to address software engineering issues [14]. Meanwhile, data-driven decision-making is the concept that refers to using analytics to obtain information, patterns, and insights to make decisions that are better informed and supported by factual data [15]. Thus, we utilized a data-driven decision-making approach to determine the popular programming language for developing AI systems.

Fig. 2 Data-Driven Approach

To conduct this phase of our study, we implemented all stages of a data-driven approach, and we just adopted a few steps that depend on our needs and resources. Because of that, we perform four activities sequentially, including data collection, data visualization, data analysis, and decision-making process.

C. Experiments

Software Development Life Cycle (SDLC) is a systematic approach that aims to produce software structured and efficiently, encompassing all stages of software development, including planning, coding, testing, and deployment [16]. In this study, we perform several experiments related to the performance of code execution by each programming language. These include creating flowcharts as code design, implementing them into the code lines, conducting unit tests for every function, and analyzing the result in the table format. To plan our experiments, we composed a list of the investigations that will be tested in some programming languages selected to gain information about comparing speed execution.

TABLE II
LIST OF EXPERIMENT

ID	Name	Big O Notation
1	Array Mutation	n
2	Dot Product Calculation	n^2
3	Quick Sort Algorithm	$n\log(n)$

D. Documentation

For researchers, writing a scientific paper is the most straightforward way to contribute to advancing the interested field [17]. Thus, we wrote this paper to disseminate our research findings to fellow researchers to invite feedback that motivates future works. Meanwhile, it can be the documentation that provides valuable information and encourages knowledge sharing, empowering readers to understand how processes work.

III. Results and Discussion

In this section, we undertake two main phases with several crucial activities, including a data-driven approach and performing the experiments using SDLC. The difference between those main phases is situated in the implementation. In this study, a data-driven approach constitutes a flow that performs step-by-step sequentially. Instead, the SDLC experiments are the cycles that execute the stage repeatedly in distinct iterations.

A. Data-Driven Approach:

In the context of the current era of big data, a novel methodology for managing data, which comprises a sequence of five distinct stages, has been developed. The fundamental stages involved in data processing include data collection, cleansing, storage, analysis, and subsequent exploration [18]. But, in this paper, we discovered knowledge with four distinct stages: data collection, data visualization, data analysis, and decision-making process.

The first stage of data-driven approach implementation is Data Collection, which constitutes a systematic and organized technique for gathering information from service providers and other external data sources with the objective of observations or measurements in a study that can be qualitative or quantitative to assess outcomes or derive actionable insights for further steps [19].

We manually gathered data on jobs in the Artificial Intelligence (AI) field from famous job portal platforms, such as LinkedIn, Glints, and others. Then, we gain employment information in as many as 129 data rows with locations in a few countries. Furthermore, we seek insights into the programming languages that have become many companies' needs for building AI model systems.

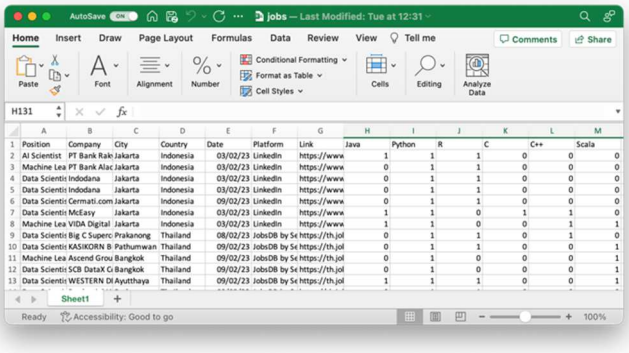

Fig. 3 Data Collection

We placed the data in Microsoft Excel format due to Efficiently carry out data manipulation, analysis, and visualization tasks without investing significant time in acquiring expertise in intricate programming languages. Furthermore, we publicly uploaded the data to the Kaggle site to provide data transparency in this study.

Then, we perform data visualization after collecting data on the internet and putting those into the spreadsheet. Data visualization is analyzing data and exhibiting it visually, graphically, or pictorially, intended to assist individuals in comprehending the meaning of data by providing a clear and straightforward summary of a substantial volume of data to communicate information effectively and lucidly [20].

To represent insight into data that have been collected, we visualize it in column chart format due to column chart is frequently utilized to compare multiple items that fall within a specific range of values and are considered optimal when a comparison is required between individual sub-items in the context of a particular data category [21].

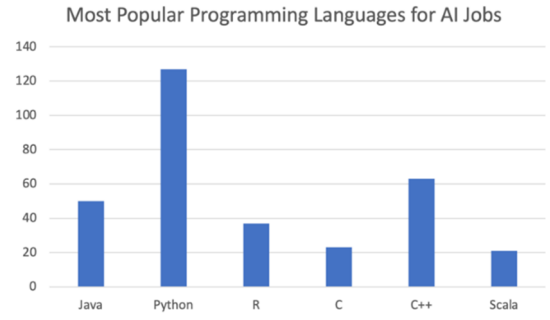

Fig. 4 Data Visualization

After performing data visualization, we conduct data analysis. We systematically apply statistical techniques to glean helpful information by inspecting, cleansing, transforming, and modeling raw data into valuable data to discover useful information, conclude relevant knowledge, and support decision-making.

TABLE III
COMPARISON OF PROGRAMMING LANGUAGES REQUIRED

Programming Language	Amount
Python	127
C++	63
Java	50
R	37
C	23
Scala	21

The activity involves the determination process and includes the last stage of a data-driven approach implemented in this study. Decision-making is the comprehensive process of determining the best option to achieve organizational objectives by evaluating situations or problems, weighing evidence, and examining alternatives among several possible so it can generate thoughtful decisions by organizing relevant information [22].

Table III shows that Python is the most demanded programming language for AI jobs in various countries. But we can't be allowed to create conclusions based on results generated from this process. Thus, we decided to conduct comparison experiments involving Python, C++, and Java as the top 3 programming in demand.

The following is an explanation us regarding the three programming languages that we chose as experimental objects:

Python is a programming language that is interpreted and considered high-level, which is utilized extensively for web development, data analysis, scientific computing, and artificial intelligence applications [23]. It was first released in 1991 and is famous for its simplicity, readability, and ease of use, becoming one of the most widely used programming languages. Python has a sizable standard library and a broad range of third-party packages, making it appropriate for various tasks. It emphasizes code readability and is frequently employed as a first language for novices to learn to program. The language is open source, with many developers contributing to its growth and development.

Java is an independent, popular, high-level, and object-oriented programming language platform. Sun Microsystems created it and was first released in 1995. We can use Java for various applications, including web development, mobile app development, and desktop applications. The language emphasizes the "write once, run anywhere" principle, which means that we can run Java code on any platform with a Java Virtual Machine (JVM) installed without recompilation. Java is known for its sizeable standard library, third-party packages, and security features, making it a favored language for many developers [24].

C++ is a general-purpose, high-level programming language developed in the 1980s as an extension of the C programming language. It is an object-oriented language extensively used for system software, device drivers, game development, and more. C++ is renowned for its power, flexibility, and efficiency, allowing developers to write low-level code while providing high-level abstractions. C++ is a compiled language, meaning the source code is translated into machine code that a computer can execute. C++ includes a large standard library and an extensive ecosystem of third-party packages, making it adaptable to various applications. Its performance, portability, and compatibility with C code make it popular among developers [25].

TABLE IV
COMPARISON OF PARADIGM PROGRAMMING LANGUAGE

Paradigm	Programming Language		
	Java	Python	C++
Generic	Yes	No	Yes
Object-oriented	Yes	Yes	Yes
Functional	Yes	Yes	Yes
Imperative	Yes	Yes	Yes
Reflective	Yes	Yes	No
Concurrent	Yes	No	No
Structured	No	Yes	No
Procedural	No	Yes	Yes
Modular	No	No	Yes

A programming paradigm is a methodology or approach to computer programming that entails a collection of principles, concepts, and techniques to design and develop software [26]. It involves a particular way of thinking about how to write code and solve problems using specific methodologies and techniques. Various programming paradigms exist, such as procedural, object-oriented, functional, and logical. The selection of a programming paradigm is influenced by factors such as the project's requirements, the problem domain, and the personal preferences of the developer or development team.

Table IV above clarifies that three programming languages, Java, Python, and C++, have various characteristics that will be the future goals and contributions to software product development. The existence of substantial programming languages aims to complement each other.

B. Experiments Using SDLC Concept

To enforce a few experiments purely involving three programming languages, we compile and run the program files in MacOS Terminal without implicating Integrated Development Environment (IDE) applications because different programming languages need other IDE for execution processes, so the results will be invalid because it can't say "apple to apple."

The MacOS Terminal is a command-line interface with the macOS operating system [27]. It enables users to interact with the computer's file system and execute various commands and utilities through a text-based interface. The Terminal facilitates multiple tasks such as navigating directories, creating and modifying files, installing software, and managing system settings. Advanced users and developers who prefer working with command-line interfaces rather than graphical user interfaces find the Terminal a potent tool.

We describe the device we utilize in these experiments due to the transparency of the experimental result if there is dissimilarity in the execution time length on other devices.

TABLE V
DEVICE INFORMATION

Device Information	Value
Manufacture	Apple
Handset Model	MacBook Pro (15-inch, 2019)
Operating System	MacOS Monterey 12.5
Processor	2,4 GHz 8-Core Intel Core i9
RAM	16 GB 2400 MHz DDR4
Storage	SSD 256 GB

Identical to the data-driven approach that not all steps are commonly used, we implemented them in this study; we also utilize several stages according to our needs. There are four main stages of SDLC that we perform in these experiments.

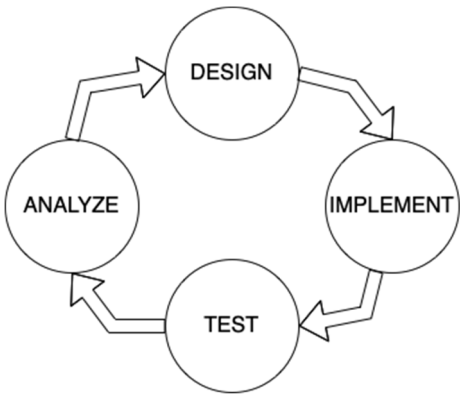

Fig. 5 Experiment Process

The Design Phase in SDLC is a pivotal stage that defines the software system's technical, functional, and architectural requirements. Ultimately, the design phase produces a detailed plan for software developers to follow while coding the system. We created a flowchart for every experiment done. A flowchart is a diagram representing a step-by-step process using different symbols and containing a short description to present the flow of algorithms [28]. Hence, visualize complex processes or make explicit the structure of tasks.

After the design phase, we conduct the coding or implementation phase in SDLC is a stage in the software system's design that converts into a fully functioning program that utilizes the chosen programming languages and development tools to write the code that will make the software operational. Then, there is the testing phase in SDLC, a crucial stage where the software system undergoes a rigorous assessment to determine its quality, functionality, and performance. It involves executing a series of tests designed to identify any defects or issues in the software.

In this stage, we use Unit Tests to check small pieces of code to deliver information and validate that each unit of the software works as intended and meets the requirements. Then, inspecting performance makes the building flow more vulnerable to environmental issues. To conduct the tests of the three experiments that we have decided on before, we prepare the Terminal and command line that we can use to compile and run the program files from different programming languages.

In the context of programming, "compile and run" refers to transforming human-readable source code into machine-readable code that we can execute on a computer. This process begins with compilation, translating the source code into a format the computer's processor can understand. This resulting code is often stored in a binary or executable file. Once this compilation process is complete, the computer can execute the program and perform the actions specified in the source code. Some popular programming languages requiring compilation before execution include C, C++, and Java.

TABLE VI
DIFFERENT COMMAND LINES FOR PROGRAMMING LANGUAGES

Programming Language	Command Line	
	Compile	Run
C++	`g++ -std=c++17 -g` *<file name>* `-o` *<output name>*	`./`*<output name>*
Java	`javac` *<file name>*	`java` *<file name>*
Python	-	`python` *<file name>*

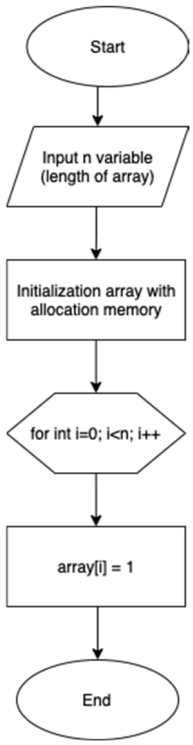

Fig. 6 Flowchart of Array Mutation Experiment

Python is categorized as an interpreted language since the interpreter executes its source code without requiring a compilation process. During runtime, the interpreter reads and executes the code line by line instead of converting it into machine code, as with compiled languages [29]. This approach provides a more dynamic and interactive programming experience as developers can promptly view the output of their code without waiting for a compilation phase. Nonetheless, it might lead to relatively lower performance as the code has to be interpreted each time it is executed.

The last stage in every iteration of implementing SDLC for the comparison programming languages experiments is The Analysis Phase, which constitutes a critical stage where the software system's requirements are examined thoroughly and evaluated to ensure that the specific features and functionalities are fully understood and defined before moving forward to the next iteration.

In this stage, we need data understanding that involves accessing the experimental result to find benchmarks and set performance goals to gain general insights that will

potentially be helpful for further steps in the form of a table or graphic.

We explain the process of the experiments that are executed using SDLC in the following:

1) Array Mutation

The process of array mutation involves altering the values of an array that already exists. Arrays are data structures that store elements of the same data type [30]. We can modify their values by changing one or more elements or performing operations on multiple array elements simultaneously.

The ability to mutate arrays is a fundamental aspect of many programming languages and a necessary part of building complex applications and algorithms. Nevertheless, it is imperative to ensure that array mutation executes accurately and efficiently, as errors or inefficiencies in this process can result in program failures or suboptimal performance.

TABLE VII
COMPARISON OF PROGRAMMING LANGUAGE EXECUTE ARRAY MUTATION

Programming Language	Time Execution (milliseconds)
C++	2.830,52
Java	1.878,73
Python	63.614,47

Table VII above demonstrates that Java has the fastest performance for processing array mutation with the length of array 1.000.000.000 compared to other programming languages. Java can be the most rapid programming language in this experiment because Java compilers can optimize array access and iteration to reduce the number of instructions, leading to better performance.

2) Dot Product Calculation

The dot product is a mathematical operation used in convolution functions, which involves the computation of the sum of the outcomes of the corresponding elements of two matrices. The dot product is executed precisely between a kernel matrix, also known as a filter, and a portion of the input matrix referred to as the receptive field. The dot product is performed by element-wise multiplication of the two matrices, followed by the sum of the resulting products [31]. This process is an essential part of the convolution process, commonly utilized in deep learning and image processing applications to extract crucial features from input data. The dot product enables the kernel matrix to be applied to the input matrix in a sliding window manner, allowing the convolution operation to be executed at various locations of the input matrix.

TABLE VIII
COMPARISON OF PROGRAMMING LANGUAGE EXECUTE DOT PRODUCT

Programming Language	Time Execution (milliseconds)
C++	0,27856
Java	0,38393
Python	16,756

Table VIII above demonstrates that C++ has the fastest performance for processing the function of dot product calculation with the length of array 100.000 compared to other programming languages. C++ can be the most rapid programming language in this experiment due to its ability to write highly optimized code through low-level memory manipulation and control.

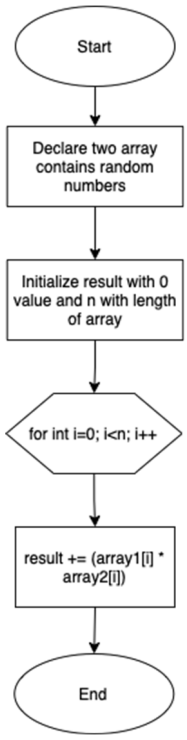

Fig. 7 Flowchart of Dot Product Calculation Experiment

3) Quick Sort Algorithm

Quick sort is a sorting algorithm widely used in computer science that follows a divide-and-conquer strategy. The algorithm selects a pivot element from the array, divides the remaining elements into two sub-arrays based on whether they are more significant than or less than the pivot, and recursively applies the same process to each sub-array. Typically, the last element of the array is selected as the pivot. The algorithm rearranges the array such that all elements less than the pivot are placed before it, and all elements more outstanding than the pivot are set after it, a process called partitioning. The quick sort algorithm continues recursively partitioning the sub-arrays until the sub-arrays contain only one element [32].

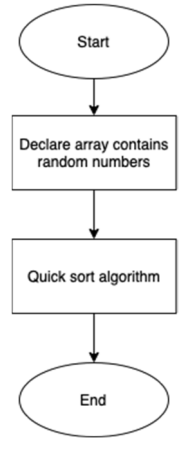

Fig. 8 Flowchart of Quick Sort Experiment

The time complexity of quick sorting is $O(^{n}\log_{n})$ on average, which makes it one of the fastest sorting algorithms in most cases. However, in the worst-case scenario, where the pivot is the largest or smallest element, it is much slower than the average case. To avoid the worst-case scenario, we can use various techniques, such as selecting a random pivot or using a median-of-three method.

TABLE IX
COMPARISON OF PROGRAMMING LANGUAGE EXECUTE QUICK SORT

Programming Language	Time Execution (milliseconds)
C++	17,6858
Java	12,2811
Python	243,445

Table VIII above demonstrates that Java has the fastest performance for processing the quick sort algorithm with the length of array 100.000 compared to other programming languages. Java can be the most rapid programming language in this experiment because Java arrays are stored in contiguous blocks of memory, enabling efficient traversal of array elements using a short loop. The Java Virtual Machine (JVM) optimizes the loop for better performance. Moreover, using primitive data types in arrays makes it possible to process large datasets more quickly [33].

C. Discussions

From a data-driven approach to finding the most demanding programming language, we can synthesize that Java is widely used in various machine learning projects. Therefore, engineers who have some experience with Java are needed in many job opportunities. Furthermore, the experiments that adjust to the basics of deep learning intend to compare the most capable programming language, indicating that Java has the fast-executing code capability to compute the high number array.

TABLE X
COMPARISON OF PROGRAMMING LANGUAGE POSITION IN THIS STUDY

Aspects	Python	Java	C++
The most demanding programming for the requirement of job opportunities	1st position	3rd position	2nd position
The most fast-executing for array mutation experiment	3rd position	1st position	2nd position
The most fast-executing for dot-product calculation experiment	3rd position	2nd position	1st position
The most fast-executing for quick sort experiment	3rd position	1st position	2nd position

Java came first in two aspects compared to Python and C++, which only came first in one aspect. Besides that, Java is a programming language that has been used since the early development of Android. Thus, we selected Java to develop a deep learning library that can be embedded in mobile applications.

IV. CONCLUSIONS

Deep Learning applications typically involve multidimensional data arrays as inputs and multidimensional parameter arrays, known as kernels, that are adjusted by the training algorithm. Thus, the performance of array processing is a crucial part of considering the software library development for Deep Learning. Meanwhile, the existing programming languages have different treatments to execute the array that has been initialized. From the experiments conducted, we can conclude that Java is superior in processing arrays compared with Python and C++. Java is a programming language still in demand in several countries for jobs in artificial intelligence departments. We will use Java programming language to build a deep learning library from scratch for further research.

ACKNOWLEDGMENT

We appreciate the reviewers for their insightful feedback to improve the quality of this paper. We want to thank the Electronic Engineering Polytechnic Institute of Surabaya for facilitating and supporting us in researching and developing a product that we believe can significantly contribute to our campus and country.

REFERENCES

[1] S. E. Whang, Y. Roh, H. Song, and J.-G. Lee, "Data Collection and Quality Challenges in Deep Learning: A Data-Centric AI Perspective," *The VLDB Journal*, vol. 32, pp. 791–813, 2023.

[2] E. Stevens, L. Antiga, and T. Viehnam, *Deep Learning with PyTorch*. Manning, 2020.

[3] A. Kapoor, A. Gulli, and S. Pal, *Deep Learning with TensorFlow and Keras*. Packt Publishing, 2022.

[4] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Vancouver, 2019, pp. 8024–8035.

[5] N. Ramadijanti, A. Barakbah, and F. A. Husna, "Automatic Breast Tumor Segmentation using Hierarchical K-means on Mammogram," 2018 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC), Oct. 2018, doi:10.1109/kcic.2018.8628467.

[6] A. R. Barakbah, T. Harsono, and A. Sudarsono, "Automatic Cluster-oriented Seismicity Prediction Analysis of Earthquake Data Distribution in Indonesia," International Journal on Advanced Science, Engineering and Information Technology, vol. 9, no. 2, pp. 587–593, Apr. 2019, doi: 10.18517/ijaseit.9.2.7269.

[7] M. Subhan, A. Sudarsono, and A. R. Barakbah, "Classification of Radical Web Content in Indonesia using Web Content Mining and k-Nearest Neighbor Algorithm," EMITTER International Journal of Engineering Technology, vol. 5, no. 2, pp. 328–348, Jan. 2018, doi:10.24003/emitter.v5i2.214.

[8] M. N. Shodiq, D. H. Kusuma, M. G. Rifqi, A. R. Barakbah, and T. Harsono, "Neural Network for Earthquake Prediction Based on Automatic Clustering in Indonesia," JOIV : International Journal on Informatics Visualization, vol. 2, no. 1, pp. 37–43, Feb. 2018, doi:10.30630/joiv.2.1.106.

[9] R. Cordingly et al., "Implications of Programming Language Selection for Serverless Data Processing Pipelines," 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), Aug. 2020, doi:10.1109/dasc-picom-cbdcom-cyberscitech49142.2020.00120.

[10] S. A. Abdulkareem and A. J. Abboud, "Evaluating Python, C++, JavaScript and Java Programming Languages Based on Software Complexity Calculator (Halstead Metrics)," IOP Conference Series: Materials Science and Engineering, vol. 1076, no. 1, p. 012046, Feb. 2021, doi: 10.1088/1757-899x/1076/1/012046.

[11] H. Snyder, "Literature review as a research methodology: An overview and guidelines," Journal of Business Research, vol. 104, pp. 333–339, Nov. 2019, doi: 10.1016/j.jbusres.2019.07.039.

[12] I. Aguilera-Martos *et al.*, "TSFEDL : A python library for time series spatio-temporal feature extraction and prediction using deep learning," *Neurocomputing*, vol. 517, pp. 223–228, Jan. 2023, doi:10.1016/j.neucom.2022.10.062.

[13] S. Akcay, D. Ameln, A. Vaidya, B. Lakshmanan, N. Ahuja, and U. Genc, "Anomalib: A Deep Learning Library for Anomaly Detection," 2022 IEEE International Conference on Image Processing (ICIP), Oct. 2022, doi: 10.1109/icip46576.2022.9897283.

[14] F. Cunha, T. Rique, M. Perkusich, K. Gorgônio, H. Almeida, and A. Perkusich, "A Data-driven Framework to Support Team Formation in Software Projects," Anais do II Workshop Brasileiro de Engenharia de Software Inteligente (ISE 2022), Oct. 2022, doi:10.5753/ise.2022.227029.

[15] N. Elgendy, A. Elragal, and T. Päivärinta, "DECAS: a modern data-driven decision theory for big data and analytics," Journal of Decision Systems, vol. 31, no. 4, pp. 337–373, Mar. 2021, doi:10.1080/12460125.2021.1894674.

[16] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, "A Literature Review of Using Machine Learning in Software Development Life Cycle Stages," IEEE Access, vol. 9, pp. 140896–140920, 2021, doi:10.1109/access.2021.3119746.

[17] X. Bai, M. Wang, I. Lee, Z. Yang, X. Kong, and F. Xia, "Scientific Paper Recommendation: A Survey," IEEE Access, vol. 7, pp. 9324–9339, 2019, doi: 10.1109/access.2018.2890388.

[18] G. Sun, H. Z. Lv, W. D. Jiang, and F. H. Li, "General process of big data analysis and visualisation," International Journal of Computational Science and Engineering, vol. 23, no. 2, p. 177, 2020, doi: 10.1504/ijcse.2020.110543.

[19] J. Liu et al., "Data Mining and Information Retrieval in the 21st century: A bibliographic review," Computer Science Review, vol. 34, p. 100193, Nov. 2019, doi: 10.1016/j.cosrev.2019.100193.

[20] K. Börner, A. Bueckle, and M. Ginda, "Data visualization literacy: Definitions, conceptual frameworks, exercises, and assessments," Proceedings of the National Academy of Sciences, vol. 116, no. 6, pp. 1857–1864, Feb. 2019, doi: 10.1073/pnas.1807180116.

[21] K. Citra and F. Wahyuni, "Exploring Demographic Variations of Freshmen to Online Learning Anxiety: A Data Visualization Analysis Based Approach," 2021 International Research Symposium On Advanced Engineering And Vocational Education (IRSAEVE), Sep. 2021, doi: 10.1109/irsaeve52613.2021.9604012.

[22] G. O. Odu, "Weighting methods for multi-criteria decision making technique," Journal of Applied Sciences and Environmental Management, vol. 23, no. 8, p. 1449, Sep. 2019, doi:10.4314/jasem.v23i8.7.

[23] A. Nagpal and G. Gabrani, "Python for Data Analytics, Scientific and Technical Applications," 2019 Amity International Conference on Artificial Intelligence (AICAI), Feb. 2019, doi:10.1109/aicai.2019.8701341.

[24] L. Ardito, R. Coppola, G. Malnati, and M. Torchiano, "Effectiveness of Kotlin vs. Java in android app development tasks," Information and Software Technology, vol. 127, p. 106374, Nov. 2020, doi:10.1016/j.infsof.2020.106374.

[25] A. Gyen and N. Pataki, "Comprehension of Thread Scheduling for the C++ Programming Language," 2021 International Conference on Data and Software Engineering (ICoDSE), Nov. 2021, doi:10.1109/icodse53690.2021.9648489.

[26] S. Krishnamurthi and K. Fisler, "Programming Paradigms and Beyond," The Cambridge Handbook of Computing Education Research, pp. 377–413, Feb. 2019, doi: 10.1017/9781108654555.014.

[27] A. Adekotujo, A. Odumabo, A. Adedokun, and O. Aiyeniko, "A Comparative Study of Operating Systems: Case of Windows, UNIX, Linux, Mac, Android and iOS," International Journal of Computer Applications, vol. 176, no. 39, pp. 16–23, Jul. 2020, doi:10.5120/ijca2020920494.

[28] J. A. Fabro, E. Teixeira Paula, A. F. G. P. Dias, and L. E. Skora, "Programming Teaching Using Flowcharts in a Simulated Environment Focused on Introducing Practical OBR," 2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE), Oct. 2019, doi: 10.1109/lars-sbr-wre48964.2019.00086.

[29] A. Javed, M. Zaman, M. M. Uddin, and T. Nusrat, "An Analysis on Python Programming Language Demand and Its Recent Trend in Bangladesh," Proceedings of the 2019 8th International Conference on Computing and Pattern Recognition, Oct. 2019, doi:10.1145/3373509.3373540.

[30] J. Pivarski, D. Lange, and P. Elmer, "Nested data structures in array frameworks," Journal of Physics: Conference Series, vol. 1525, no. 1, p. 012053, Apr. 2020, doi: 10.1088/1742-6596/1525/1/012053.

[31] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*. Cambridge University Press, 2023.

[32] S. K. Gill, V. P. Singh, P. Sharma, and D. Kumar, "A comparative study of various sorting algorithms," *International Journal of Advanced Studies of Scientific ResearchInternational Journal of Advanced Studies of Scientific Research*, vol. 4, no. 1, 2019.

[33] Y. Chen, T. Su, and Z. Su, "Deep Differential Testing of JVM Implementations," 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), May 2019, doi:10.1109/icse.2019.00127.