Python Libraries for Health Data Analysis: A Review for Apple Silicon

Utharn Buranasaksee*[1], Nadaphast Koomklang[2], Namooy Panya[2], Montri Sangthong[3], Ekachai Naowanich[1]

[1]Department of Computer Science, Faculty of Science and Technology
[2]Department of Science, Faculty of Science and Technology
[3]Department of Mathematics, Faculty of Science and Technology
Rajamangala University of Technology Suvarnabhumi
Nontaburi, Thailand
email: utharn.b@rmutsb.ac.th

Abstract— The paper presents an overview of health data analysis and the challenges associated with handling large and complex datasets. Then, the authors highlight the significance of Python as a popular language for health data analysis, emphasizing its flexible syntax and efficient libraries written in C/C++. As the data keep larger, this raises memory-related issues in current hardware platforms. While the emergence of the Arm platform, exemplified by Apple's M1 and its predecessors, the Arm platform is presented as a potential solution providing that shared memory and high-speed memory access are built into the platform. Therefore, many Python libraries for health data analysis are surveyed and analyzed for compatibility issues with Apple silicon. Finally, we conclude the migration checklist for developers transitioning to the Arm platform.

python; health; data; analysis; arm; apple; silicon;

I. INTRODUCTION

Health data analysis is one of the data types that many data analysts tried to perform exploratory analysis and deep analysis. Due to its nature that the data size is tremendously large and has many complex attributes. Many analysts find that they need a tool that supports this kind of data.

Python language is one of the most popular languages that have been used for health data analysis [1]. The language has two main advantages. First, the language has simple and flexible syntaxes that allow programmers to easily interact through the code to achieve data analysis jobs. Second, while providing a simple interface, the language has related libraries written in C/C++ which are more efficient than many other programming languages.

Since many of the libraries have been written in C/C++, they are designed and targeted only to specific platforms. One of the main platforms that are widely used around the world is the x64 platform. The x64 platform allows the system to scale up by having multiple cores and a larger level 3 (L3) cache, the main memory sticks in the x64 platform are physically located at a far distance from the CPU. As a result, the memory speed is much lower when compared to that in GPU as the memory is soldered in the GPU board. Furthermore, in the GPU domain, the low-level library that allows complex operation in GPU is largely dominated by NVIDIA [2]. Therefore, the libraries that are targeted to GPU based are likely to be developed to support the CUDA framework. Therefore, most of the libraries are targeted to both the x64 CPU platform and CUDA-based GPU platform.

However, as the chip-making technology progress, the Arm platform that has mainly dominated in mobile phone market could achieve acceptable performance in data analysis task. Many data analysis developer finds that Arm architecture is much more efficient than the x64 platform. In 2020, Apple Inc. released its silicon chip called the M1 chip [3] which is an Arm platform and is designed to have a higher clock speed when compared to the other Arm chips in the market. Therefore, Apple silicon gains more market share, especially in the mobile market [4].

In a comprehensive data analysis task, the task may involve machine learning algorithms. These algorithms are complex and time-consuming. Furthermore, the data in the machine learning algorithm are repetitively copied, transformed, and moved in the memory [5]. Therefore, the algorithm is classified as a memory bottleneck. The memory speed dominates in the performance outcome rather than the memory size. Hence, many developers are required to use CUDA-based GPU to perform those machine learning tasks.

In a machine learning algorithm, providing more data to the algorithm usually result in a more effective decision that the algorithm can make. The set of decisions that the algorithm can make after feeding the data is called the pre-trained model. A pre-trained model acts as a set of decisions that can be stored locally in the disk and used later. Therefore, a larger size of the pre-trained model usually leads to a more effective algorithm as more sets of decisions are gathered.

However, when using the pre-trained model, the model needs to be stored in the GPU memory to achieve the optimum performance. This leads to another problem in CUDA-based GPU which is an in-card memory size. Regardless of how many CUDA cores the GPU card has, the performance only holds if the data in a data analysis task

only fit into the GPU memory. However, the top-tier CUDA-based cards in the market currently have only 24GB of memory. This indicates that the maximum size of the pre-trained model cannot be exceeded 24GB. In the context of health data analysis which has a large amount of related data, the pre-trained model will likely exceed the GPU memory as we need more accuracy from the model.

While there seems to be a problem in CUDA-based GPU, it is not the problem in the Arm context. In Arm architecture, the main memory used by the CPU and GPU are shared. For instance, the M1 chip and its predecessors have both CPU and GPU acting as a system-on-chip (SoC). Furthermore, the memory and the chip are soldered closely on the same board providing high-performance memory access from the chip. In addition, the system with the latest Apple silicon chip called M2 Ultra can have the maximum of high-speed memory for 192GB [6] which is significantly greater than those CUDA-based GPUs can achieve at the same level of system cost. Therefore, the system with the Apple silicon chip can be used to achieve data analysis task that requires a tremendous amount of main memory.

In this paper, we first surveyed the Python libraries used in health data analysis. Then, we analyze the compatibility issues in both the source code level and library level as Python is the interpreted language. This is because it has been 3 years since Apple released its silicon. Many libraries have adapted their supports to cover the Apple silicon through the Apple proprietary application programming language (API) that can access its GPU called Metal. Finally, we conclude the checklist if the developer wants to migrate from the x64 and CUDA-based platform to the Apple silicon arm platform.

II. PYTHON LIBRARIES

In this section, we discuss the Python core and its libraries that are related to health data analysis about the dependencies and compatibility status.

A. Python Core

Since Python is one of the most popular languages used in data analysis, many studies surveyed the libraries used in data mining and big data analysis[7], machine learning and artificial intelligence [8], and specific tasks [9]. However, none of them has reviewed the library and the compatibility issues with the hardware platform. Therefore, we have surveyed the most popular libraries that are suitable for health data analysis. Then, we discuss the purpose of the library, the support status of the Apple silicon, and its progress.

For Python itself, Python 3.8.2 and Python 3.91 are the first two versions that natively support the Apple silicon. Installing the version stated above causes the performance to hurt from emulation x64 CPU instructions to arm CPU instructions.

B. Numpy

Numpy is a Python library for numerical and mathematical computations. It supports many data structures including arrays, matrices, and various mathematical functions, making it as a fundamental library that many libraries depend on. Numpy started supporting Python 3.8 from version 1.21.0 [10]. The library is based on NumPy which starts to support the Apple silicon from version 1.20.1.

C. Pandas

Pandas is a library that offers data structure and operations for manipulating numerical tables and time series. Since Pandas depends on Numpy, Pandas started supporting the Apple silicon in version 2.12.1 [11]. At first, developers need to download the source and build it for themselves. Finally, Pandas has had a universal package that supports both x64 and arm platforms since January 2022 [12]. Though the library utilizes neither CUDA-based GPU nor Apple GPU, the library can take full parallelism in CPU both x64 and arm platforms.

D. Matplotlib and Seaborn

Matplotlib and Seaborn are the libraries that help visualize data to graphs, charts, plots, and other graph representations. The Matplotlib started supporting the Apple silicon in version 3.4. However, the user needs to install via the conda command only [13]. By version 3.5, Matplotlib started rolling a universal package that can support both x64 and arm platforms [14]. Seaborn is the library that does the same but is built on top of matplotlib to support more complex graphs. Since Seaborn is a pure Python library, the compatibility issues were solved once the upstream package supported the arm platform.

E. Scipy and Scikit-learn

SciPy is a Python library that is used for scientific and technical computing. It is built on top of NumPy and provides functionalities for tasks related to data analysis. SciPy is often used in conjunction with other libraries like NumPy, Matplotlib, and pandas to perform complex scientific and mathematical tasks efficiently. SciPy first has supported the Apple silicon via nightly build since November 2021. However, the performance in the arm platform is much slower than that in the x64 platform. The patch was released in version 1.7.2 on February 2022 [15].

Scikit-learn, which was previously packaged as sklearn, is a Python module for machine learning built on top of SciPy. Scikit-learn tries to provide a simple interface to SciPy for machine learning tasks. Since Scikit-learn depends on both NumPy and SciPy, the project compatibility issues were fixed by February 2023 [16].

F. Statmodels

Statsmodels is a Python library that extends the capability of SciPy for statistical computations including descriptive statistics, and estimation and inference for statistical models. The library still has overflow bugs in Apple silicon [17]. The maintainers plan to address this issue in the version 0.14 milestone. However, as of now, version 0.14 was released on 5 May 2023, and the issue remains open.

G. Biopython

Biopython [18] is a set of tools written in Python and specifically designed for computational biology and bioinformatics applications. It supports functionalities for tasks related to DNA, RNA, protein sequences, molecular structures, phylogenetics, and other biological data analysis tasks. The status of Apple silicon support is not officially supported. Though power data analysts can pull the source code and compile it for themselves, Biopython heavily depends on the tools that need different compiled versions including psycopg2 [19] which depends on the Postgresql database, and mysqlclient [20] which depends on MySQL database.

H. Pydicom

Pydicom [21] is a Python package for working with DICOM [22] files. Pydicom allows developers to read, modify and write DICOM (Digital Imaging and Communication in Medicine) data. Since the library is purely written in Python. Pydicom has no compatibility issue and does not require a separate installing package.

I. Nilearn

Nilearn is a Python library designed for the analysis of brain imaging data. It specifically focuses on functional magnetic resonance imaging (fMRI) data [23], which is a common technique used to record brain activity. Nilearn itself depends on Intel one API Math Kernel Library (Intel MKL) which is not available to another platform except the x64 platform. As a result, data analysis developers cannot install Nilearn unless they use the x64 platform [24]. Therefore, Nilearn would not support Apple silicon due to one of its dependencies.

J. Tensorflow

TensorFlow is a machine learning framework developed by Google. It is designed to facilitate the development and deployment of machine learning models, particularly deep learning models, across a variety of platforms and devices. Tensorflow mainly supports CUDA-based GPU to achieve its highest performance. Though Apple releases its Metal plugins to support Tensorflow [25] and Tensorflow is a large framework, some functions remain unoptimized for the Apple silicon [26].

K. PyTorch

PyTorch is a deep learning framework developed by Facebook's AI Research Lab (FAIR). It is designed to provide a platform for building and training various deep-learning models. Many developers may choose either Tensorflow or PyTorch depending on their preferences. To support the Apple silicon, a new Metal programming framework called Metal Performance Shaders (MPS) has been released by Apple [27]. After that, the MPS backend has been integrated into PyTorch without installing additional packages [28]. However, the performance of using MPS lacks behind that of using CUDA-based which has been developed for many years [29], [30].

From the discussion above, we have concluded a comparison among the libraries, their dependencies, compatibility issues, and performance issues in Table 1.

TABLE I. PYTHON LIBRARIES AND THEIR DEPENDENCIES

Library	Dependencies	Compatibility Issues	Performance Issues
Python Core	Supports from 3.8 or above	No	No
NumPy	None (Purely written in Python)	No	No
Pandas	NumPy	No	No
Matplotlib	NumPy	No	No
Seaborn	Matplotlib, Pandas	No	No
SciPy	NumPy	No	No
Scikit-learn	NumPy, SciPy	No	No
Statsmodels	NumPy, SciPy	Yes	No
Biopython	None (Purely written in Python)	Yes	No
Pydicom	None (Purely written in Python)	No	No
Nilearn	NumPy, SciPy, Scikit-learn, Matplotlib	Yes	Yes
TensorFlow	NumPy, Google Abseil	Yes	Yes
PyTorch	None (built on C/C++ libraries)	Yes	Yes

III. CHALLENGES AND LIMITATIONS

Despite 3 years since Apple released its silicon and the performance claimed by Apple was surprisingly well. In health data analysis, the Apple silicon still supports only basic Python libraries. The x64 platform is still required for a specific use case and advanced machine learning frameworks; TensorFlow and PyTorch.

However, we can see the progress of the libraries as the developers try to embrace Apple silicon support these years. Therefore, it might take some more time for the arm platform to catch up with x64 and CUDA-based platforms.

IV. CONCLUSION

In conclusion, this paper offers a comprehensive exploration of the realm of health data analysis. Then, the authors underscore the pivotal role of Python as a preferred programming language for health data analysis, drawing attention to its adaptability in syntax and the efficacy of its C/C++ libraries. Then, as the volume of data continues to expand, the paper poignantly highlights the growing

concern of memory-related issues within existing hardware platforms. While the Arm platform has become increasingly popular among data analysts. The paper illuminates the Arm platform's potential to address these memory-related challenges. However, the libraries need to embrace their codes to support the Arm platform. As of now, only basic libraries are cross-platform. This is because many advanced libraries are written in C/C++ which are required to rewrite the code to utilize and optimize the code for new hardware. Therefore, in its current state, the Apple silicon may be only suitable for only beginner of health data analysts. Serious developers would better remain in a CUDA-based platform.

REFERENCES

[1] K. J. Millman and M. Aivazis, "Python for scientists and engineers," *Comput Sci Eng*, vol. 13, no. 2, pp. 9–12, Mar. 2011, doi: 10.1109/MCSE.2011.36.

[2] A. Ilievski, V. Zdraveski, and M. Gusev, "How CUDA Powers the Machine Learning Revolution," *2018 26th Telecommunications Forum, TELFOR 2018 - Proceedings*, 2018, doi: 10.1109/TELFOR.2018.8611982.

[3] Apple Inc., "Apple unleashes M1 - Apple," Nov. 10, 2020. https://www.apple.com/newsroom/2020/11/apple-unleashes-m1/ (accessed Aug. 08, 2023).

[4] D. L. F. Lam, "Study of iPhone's Big Data, Market Share, Usage and Their Relationships," *ACM International Conference Proceeding Series*, pp. 7–10, Aug. 2020, doi: 10.1145/3421537.3421542.

[5] X. Wu, P. Brazzle, and S. Cahoon, "Performance and Energy Consumption of Parallel Machine Learning Algorithms," 2023.

[6] Apple Inc., "Apple unveils M2 Pro and M2 Max: next-generation chips for next-level workflows - Apple," Jan. 17, 2023. https://www.apple.com/newsroom/2023/01/apple-unveils-m2-pro-and-m2-max-next-generation-chips-for-next-level-workflows/ (accessed Aug. 08, 2023).

[7] I. Stancin and A. Jovic, "An overview and comparison of free Python libraries for data mining and big data analysis," *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2019 - Proceedings*, pp. 977–982, May 2019, doi: 10.23919/MIPRO.2019.8757088.

[8] S. Raschka, J. Patterson, and C. Nolet, "Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence," *Information 2020, Vol. 11, Page 193*, vol. 11, no. 4, p. 193, Apr. 2020, doi: 10.3390/INFO11040193.

[9] J. Siebert, J. Groß, and C. Schroth, "A systematic review of Python packages for time series analysis," Apr. 2021, Accessed: Aug. 08, 2023. [Online]. Available: https://arxiv.org/abs/2104.07406v2

[10] NumPy Developers, "NumPy 1.21.1 Release Notes — NumPy v2.0.dev0 Manual." https://numpy.org/devdocs/release/1.21.1-notes.html (accessed Aug. 09, 2023).

[11] Github Inc., "Can't install Pandas on Mac M1 · Issue #40611 · pandas-dev/pandas," Mar. 24, 2021. https://github.com/pandas-dev/pandas/issues/40611 (accessed Aug. 09, 2023).

[12] Github Inc., "ENH: Please provide 'universal2' wheels for macOS · Issue #39053 · pandas-dev/pandas," Jan. 21, 2022. https://github.com/pandas-dev/pandas/issues/39053 (accessed Aug. 09, 2023).

[13] Github Inc., "matplotlib 3.4.2 using on M1 Mac, with python 3.9.4 · Issue #20261 · matplotlib/matplotlib," May 20, 2021. https://github.com/matplotlib/matplotlib/issues/20261 (accessed Aug. 09, 2023).

[14] Github Inc., "Build wheels for Apple Silicon. by QuLogic · Pull Request #20970 · matplotlib/matplotlib," Sep. 02, 2021. https://github.com/matplotlib/matplotlib/pull/20970 (accessed Aug. 09, 2023).

[15] "scipy · PyPI." https://pypi.org/project/scipy/1.7.3/ (accessed Aug. 09, 2023).

[16] Github Inc., "Unable to install dependencies on Mac M1 · Issue #22 · GokuMohandas/mlops-course," Feb. 13, 2023. https://github.com/GokuMohandas/mlops-course/issues/22 (accessed Aug. 09, 2023).

[17] Github Inc., "BUG: optimize.brentq triggers an overflow error · Issue #14851 · scipy/scipy," Oct. 14, 2021. https://github.com/scipy/scipy/issues/14851 (accessed Aug. 09, 2023).

[18] P. J. A. Cock *et al.*, "Biopython: Freely available Python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, Jun. 2009, doi: 10.1093/bioinformatics/btp163.

[19] F. Di Gregorio and D. Varrazzo, "Psycopg – PostgreSQL database adapter for Python — Psycopg 2.9.7 documentation." https://www.psycopg.org/docs/ (accessed Aug. 09, 2023).

[20] Github Inc., "PyMySQL/mysqlclient: MySQL database connector for Python (with Python 3 support)." https://github.com/PyMySQL/mysqlclient (accessed Aug. 09, 2023).

[21] Github Inc., "pydicom/pydicom: Read, modify and write DICOM files with python code." https://github.com/pydicom/pydicom (accessed Aug. 09, 2023).

[22] Mislav. Grgić, Sonja. Grgić, Institute of Electrical and Electronics Engineers. Region 8., Institute of Electrical and Electronics Engineers. Croatia Section., and S. and I. Processing. European Association for Speech, "Proceedings ELMAR-2008 : 50th International Symposium ELMAR-2008, 10-12 September 2008, Zadar, Croatia," Croatian Society Electronics in Marine, 2008, p. 654.

[23] R. A. Poldrack *et al.*, "Towards open sharing of task-based fMRI data: The OpenfMRI project," *Front Neuroinform*, vol. 7, no. JUNE, Jun. 2013, doi: 10.3389/FNINF.2013.00012.

[24] Github Inc., "[BUG] cannot install mkl (doc dependency) on Mac M1 Sillicon chip · Issue #3849 · nilearn/nilearn," Jul. 20, 2023. https://github.com/nilearn/nilearn/issues/3849 (accessed Aug. 09, 2023).

[25] Apple Inc., "Tensorflow Plugin - Metal - Apple Developer." https://developer.apple.com/metal/tensorflow-plugin/ (accessed Aug. 09, 2023).

[26] Github Inc., "Lack of Documentation for GPU Use, Especially in Metal GPUs in MacBook M1, M1 Max and M2 Chips · Issue #61092 · tensorflow/tensorflow," Jun. 27, 2023. https://github.com/tensorflow/tensorflow/issues/61092 (accessed Aug. 09, 2023).

[27] Apple Inc., "Accelerated PyTorch training on Mac - Metal - Apple Developer." https://developer.apple.com/metal/pytorch/ (accessed Aug. 09, 2023).

[28] The Linux Foundation, "MPS backend — PyTorch 2.0 documentation." https://pytorch.org/docs/stable/notes/mps.html (accessed Aug. 09, 2023).

[29] Lambda Inc., "GPU Benchmarks for Deep Learning | Lambda." https://lambdalabs.com/gpu-benchmarks (accessed Aug. 09, 2023).

[30] The Linux Foundation, "Introducing Accelerated PyTorch Training on Mac | PyTorch." https://pytorch.org/blog/introducing-accelerated-pytorch-training-on-mac/ (accessed Aug. 09, 2023).