Python for Data Analytics, Scientific and Technical Applications

Abhinav Nagpal[1], Goldie Gabrani[2]

[1]*School of Computer Science and Engineering, Vellore Institute of Technology, Vellore*
[2]*School of Engineering and Technology, BML Munjal University, Gurugram*

Abstract: Since the invention of computers or machines, their capability to perform various tasks has experienced an exponential growth. In the current times, data science and analytics, a branch of computer science, has revived due to the major increase in computer power, presence of huge amounts of data, and better understanding in techniques in the area of Data Analytics, Artificial Intelligence, Machine Learning, Deep Learning etc. Hence, they have become an essential part of the technology industry, and are being used to solve many challenging problems. In the search for a good programming language on which many data science applications can be developed, python has emerged as a complete programming solution. Due to the low learning curve, and flexibility of Python, it has become one of the fastest growing languages. Python's ever-evolving libraries make it a good choice for Data analytics. The paper talks about the features and characteristics of Python programming language and later discusses reasons behind python being credited as one of the fastest growing programming language and why it is at the forefront of data science applications, research and development.

Keywords: Python, Data Analytics, Artificial Intelligence, Deep Learning, Machine Learning, Natural Language Processing, Scientific Computations, Computer Languages, Frameworks.

I. INTRODUCTION

Data Analytics mainly deals with the development of computational methods to get insights of data and get intelligent information especially through visualization tools [1]. There has been a tradition among scientists and developers of using compiled languages like C++, C and LISP, for scientific applications and data analytics. But, in recent years many compiled languages usage is decreasing, giving way to the rise of interpreted environments, such as Octave, MATLAB and R. The popularity of R, MATLAB and other interpreted languages is due to the (i) simple and clean syntax, (ii) fine integration of simulation and visualization, (iii) immediate feedback when commands are executed(iv) presence of a many of built-in functions and libraries that operates well on arrays, (v) fast numerical operations, (vi) an easily available and (vii) very good documentation and support from the community. Many scientists have feel comfortable using MATLAB than other compiled languages having a separate visualization tools.

Nowadays, the programming language Python is rising as an alternative to MATLAB, R and other related environments. Python is an interpreted language, like MATLAB. Moreover, it also has object-oriented (OO) features and therefore provides a cross-platform interface. Developers can gain more flexibility and elegance. It is easy for by writing their software designs written in C to python. Object-oriented programming present in MATLAB is less convenient than in python. This is because Python was initially made so that it could be extended using compiled code to increase its efficiency. Many tools are also available to ease this integration.

Another advantage of Python is that it is far less complicated in most other environments because its interfacing legacy software is written in C, C++ and other languages. This is due to the fact that python was initially made so that it could be extended using compiled code to increase its efficiency. Many tools are also available to ease this integration. Having features (i)–(v), and power, the ability to perform parallel programming and interfacing capabilities, Python represents a fine environment for doing computational science (CS). Many Python based frameworks have improved how we program many scientific applications. Frameworks like TensorFlow, PyTorch, Keras etc. has revolutionized the field of deep learning. Using these frameworks, even a new developer with little knowledge can make neural networks.

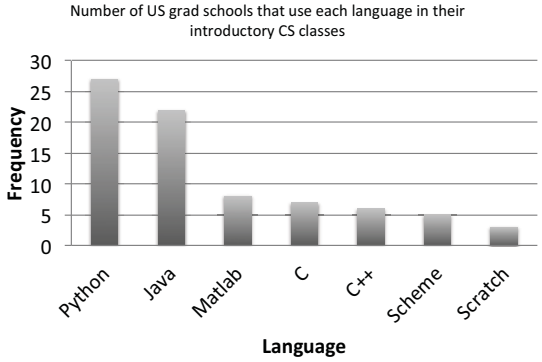

Fig. 1 Number of CS departments and the languages used to teach their introductory courses

In 1999, a proposal was submitted by Guido van Rossum [2], Python's creator, to the Defense Advanced Research Projects Agency (DARPA) stating that they require to develop a computing curriculum that is suitable for students, and to

978-1-5386-9346-9/19/$31.00 ©2019 IEEE

create tools that are more effective and simpler to use for analysis and program development. He proposed that Python meets both the needs as it is not a "toy" language and is very suitable for purposes of teaching. Now, nearly 19 years after his proposal [3], Python is widely taught as an introductory course language at many of the US computer science departments. Fig. 1 shows the number of US computer science departments and the languages they use to teach their introductory courses.

The remaining paper is divided into the following sections: Section II explains in detail the reasons why python is preferred by developers nowadays for developing scientific and technical applications and for data analytics. AI; section III discusses about the some of the most used python libraries. It also elaborates on their uses and how their use makes building applications of various domains easier. Section IV describes some of the shortcomings of the python language, followed by the speed comparison between Keras, TensorFlow and PyTorch in section V. Finally, some concluding remarks are present in section VI.

II. REASONS FOR USING PYTHON

Python has become popular for various reasons, including it's simple and a syntax that is like a pseudocode; its modularity; its object-oriented design; it's profiling, portability, testing, and self-documentation capabilities; and the presence of a Numeric library allowing the effective storage and handling of enormous amounts of numerical information.

Designed by Guido van Rossum in 1990, python is free for commercial purposes, like many other scripting languages and it can be run on most modern computers. Moreover, it provides high-level data structures such as associative arrays, lists, dynamic binding, dynamic typing modules, automatic memory management, classes, exceptions, etc. It has a small kernel and can be extended by importing external libraries. Its distribution contains a large library of standard extensions, written in python and other languages like C or C++, for operations such as Perl-like regular expressions, string manipulations, web-related utilities, operating system services, testing, and profiling tools, debugging, etc. The language can be extended by creating new modules. Some of the features of python are described as below:

A. Less Coding and High Readability

There are a lot of complex algorithms involved in AI, deep learning etc. Python involves very less coding, in terms of the lines of code, among many programming languages, which can be used for developing many such applications. This feature enables easy testing and hence developers can focus more on actual programming. Python uses as much as 1/5th code as compared to other OOPs languages to implement the same logic. Fig. 2 [4] shows the average project size in millions lines of code for many languages.

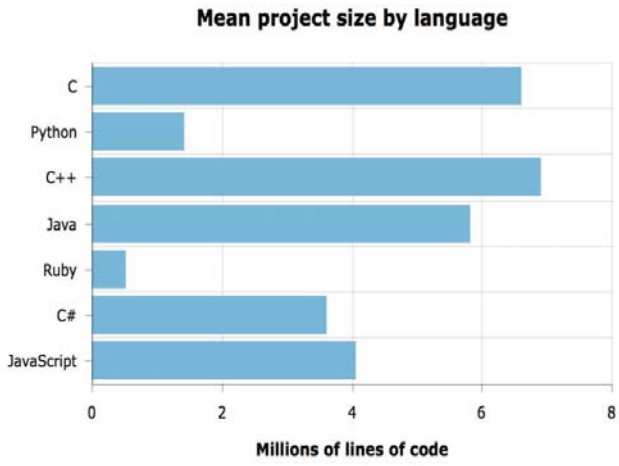

Fig. 2. Average project size by language

As seen from figure 2, python is only second after ruby with respect to the lines to code. Therefore, this is the first reason why Python is preferred by many businesses for many ML and AI-based projects. It has lesser keywords, simple structure, and a clearly defined syntax. This allows students or new programmers to pick up the language quickly. Unlike many scripting languages, readability was the main concern when python was made. Thus, python code is highly visible to the eyes. In addition, high readability encourages collaboration leading to more contribution to an open source Python project, thus leading to a rapid development cycle. It has also been seen that python users are more loyal to the language as compared to R users [5]. Fig. 3 shows the trends of loyalty in case of python and R users.

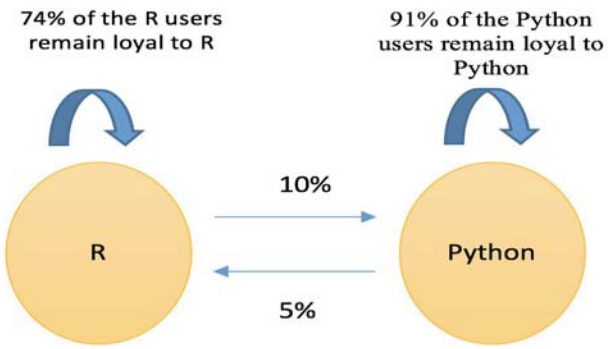

Fig. 3. Loyalty of python and R users

B. Portability and Flexibility

Python being flexible offers an option for developers to choose either to use OOPs approach and/or perform scripting and therefore, is right for developers for many purposes. It can be used to link different data structures (DS) and can be used as a backend language. Its majority of code is checked in the IDE.

This is of great help to the developers working to write different algorithms in many domains viz. AI, neural network, deep learning, computer networks algorithms. It runs on different platforms with the same interface.

C. Platform Independent

Python gives developers the flexibility to provide an API from the current programming language which turns out to be extremely flexible for the new Python developers. Moreover, it is platform independent. Developers can change the source code of their project in a small way to get their project to run on different OS, thus saving a lot of time and work.

D. Balance of Low-Level and High-Level Programming

Python has the ability to balance high-level programming with low-level. This is one of the most important feature Python. It provides high performance on its higher level objects like arrays and matrices. One such example is vectorizing in algorithms. This makes it possible to work with an entire array than a single number. As a result, the coding accuracy and coding speed increases. Such an operation is very important for good scientific coding. However, it is not helpful when during the development of new algorithms. Cython [6], a superset of python, is used to solve this problem. It first translates Python code into C code and used the Python C API to run the code. A shared library, equivalent to the original module, is created that is loaded in the form of a Python module and runs slightly faster.

Let us consider a simple example of finding the n^{th} Fibonacci term of the Fibonacci series to have a speed comparison between python and cython. Figure 4 and 5 shows the code for Fibonacci series and Table I shows the comparison between their runtimes. As seen from Table I, cython is faster than python by 38 times even when we are considering such an easy example.

```
1:   def fib(n):
2:     a=0
3:     b=1
4:     for i in range(n):
5:          a, b = a+b, a
6:     return a
```

Fig. 4. Code to find the nth term of Fibonacci series in Python

```
1:   def fib(int n):
2:     cdef int i
3:     cdef double a = 0.0
4:         cdef double b = 1.0
4:     for i in range(n):
6:          a, b = a+b, a
7:     return a
```

Fig. 5. Code to find the nth term of Fibonacci series in Cython

TABLE I: SPEED COMPARISION BETWEEN PYTHON AND CYTHON

Language	Time	Speedup
Python	6.39 ns	1x
Compiled Python	4.1 ns	1.56x
Cython	167 ns	38.26

E. Continuity

Python is nearly an ideal candidate for a first programming language. It is better to work using the language students studied in primary school. Another benefit is the fact that it was not made for educational purposes only but also to be used in later life in their professional career. It is used for example in network administration, web development, computer games programming, and many programs have an integrated support for Python scripts.

F. Data Structures

It is important for a developer to use the correct DS for an algorithm. This is particularly required in the area of research-oriented coding. The absence of such features makes a developer unable to learn and use good design patterns. Python has sets, lists, dictionaries, tuples, thread- safe queues, strings, etc. Lists are used to hold any number of data objects and can be joined, indexed, sliced, split, and used as stacks. Sets can have unique and unordered items. Dictionaries is used to map from a key, that is unique, to anything. Heaps, similar to STL heaps, are used on top of lists. Numpy provides an n-dimensional array structure with broadcasting and matrix operations. SciPy provides image objects, sparse matrices, time-series, KD-trees and much more.

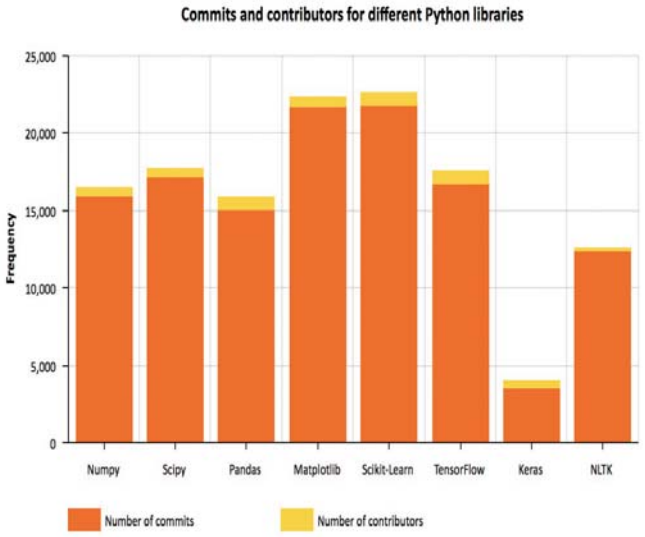

Fig. 6. Number of commits and contributors for different Python libraries

G. Available Open Source Libraries

Python, an open source (OS) language, has many libraries for almost every need of an AI project. Few of these are SciPy for advanced computing, Pybrain for ML and numpy for scientific computation. AIMA, implemented from "Artificial Intelligence: A Modern Approach" is one of the best performing library available for AI even today. Fig. 6 shows the number of commits and contributors for different Python libraries on GitHub. Dedicated libraries help save a developer's time from coding basic algorithms. Moreover, there is a huge developer community of Python developers who are willing to help other Python developers in various stages of their development lifecycle. The next section describes some of the most popular Python libraries.

III. POPULAR PYTHON LIBRARIES

A. NumPy

In order to make hash tables by enumerating a collection of objects and dictionaries, Python contains some high-level data structures like lists. However, they are not able to provide high-performance numerical computation. Numpy [7], one of the most used python library, deriving its name from two words - Numerical Python, is used by most developers as it provides high performance numerical and scientific computations on multidimensional arrays. It consists of many library functions and operations for the purpose of numerical computation. It maintains its good performance even while providing a high-level abstraction for numerical computation. It helps to cut down on python loops and instead, the main time is spent on the vectorized array operations. When a python code is written in this manner, it is able to cut down on the computation time by a large amount. More performance is achieved by using compilers that are optimized. One example is Cython. It allows a good control over cache effects. Numpy also consists of many smaller sub-packages for various mathematical tasks like linear algebra, FFTs, generating a random value, and polynomial manipulation. SciPy, another python library is built on numpy.

B. Pandas

To perform real-world data analysis, pandas [8], a python library is being developed since 2008. It aims to be the building block for data analysis. Being open-source, it is being developed by a community of users. It gives high-performance and provides tools and data structures that are simple and easy to use. Developers use it for loading data, preparing data, manipulating it, followed by its modeling, and then to analyze it. Figure 7 [9] shows that numpy is better performing that pandas. Despite its lower performance, it is preferred among users because one is able to perform many time-consuming data science tasks such as Boolean indexing, checking for NaN's in a dataset, selecting and dropping a column from a dataset etc. All these tasks does not require loops.

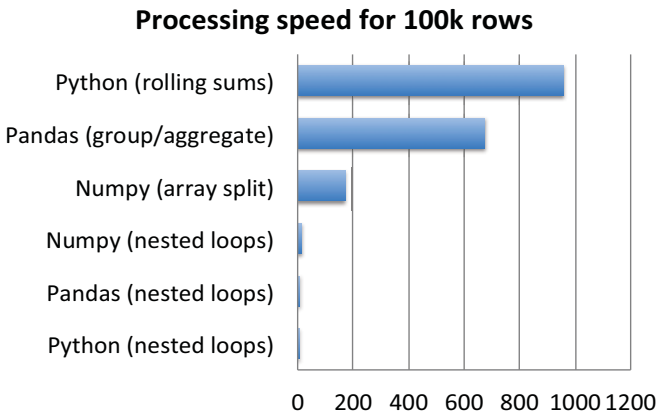

Fig. 7. Performance comparison between python, numpy and pandas

C. TensorFlow

TensorFlow [10], an open source software library, offers a strong support for ML and deep learning and the flexible numerical computation core. Using its front-end API, we can make applications while it executes the applications in C++ giving high performance.

TensorFlow is used for training and running deep neural nets for natural language processing (NLP), image recognition, sequence-to-sequence models for machine translation, recurrent neural networks (RNNs), etc. Data flow graph, a structure that describes how data will move a series of nodes, can also be created by developers. All nodes present in the graph is one of mathematical operation while edges between the nodes is a multi-dimensional array of data known as a tensor. Instead of worrying about implementing algorithms and connecting different functions together, developers can concentrate on the main logic.

The math operations in TensorFlow are not performed in Python. These libraries provided by TensorFlow are written in the form of C++ binaries. Python is used to directs traffic occurring between the pieces and then hook them together. The models on TensorFlow can be deployed by the developers across various platforms like desktops, clusters of servers, mobile etc.

D. NLTK

NLTK [11] is a collection of libraries or modules for natural language processing (NLP) in python for English. It was developed for research and teaching purpose of NLP and fields like cognitive science, information retrieval, AI, ML, empirical linguistics. Implemented as a collection of interdependent modules, it consists of a set of core modules that defines the basic data types. The other modules are used to perform a specific NLP task.

For e.g. NLTK's parser module is used for parsing or to derive the syntactic structure of any given sentence, and NLTK's tokenizer module is used for tokenizing i.e. breaking the text into parts. It provides full support to tasks such as stemming, classification, tagging, semantic reasoning functionalities etc.

E. Scikit-Learn

Scikit-learn [12], a ML library was developed as a Google Summer of Code project. It was designed to work along with NumPy and SciPy. It is written partly in python and cython to get an effective performance. Built upon SciPy, it provides a large number of unsupervised and supervised learning algorithms. Some group of models present in Scikit-learn includes clustering, Datasets, Ensemble methods, cross-validation, Feature selection, Dimensionality Reduction, Parameter Tuning, Feature extraction and Manifold Learning.

F. Matplotlib:

Matplotlib [13], a 2D plotting library, is used for embedding plots of various types across platforms. Used in IPython shells, web application servers, python scripts and Jupyter notebook, it produces figures of publication quality. It allows us to generate, by using a few lines of code, figures like bar charts, pie charts, scatterplots, histograms, error charts, etc.

IV. DISADVANTAGES OF PYTHON

Even though python has many advantageous features, and many programmers prefer to use this language over other programming languages, it has its some disadvantages also.

A. Speed

High or low, the speed of a programming language can be a big issue. Most of the compiled languages are faster than Python. Python has some benchmarks that run faster than in C or for other programming languages. Many optimized python packages in the recent few years were able to address the issue of Python's slow speed of execution. However, Python still is slower in many ways to programming languages like C++ and C, and newer ones like Go.

B. Low Usage in Mobile Applications and in Browsers

Python is being used for desktop and server platforms. However, it is weak in platforms like mobile. There have not been many smartphone apps made using it. Moreover, it is hardly seen in web applications, on the client side. Additionally, it is not being used in web development browsers. One of the major reasons for its less usage in these areas is that it is difficult to secure. While, the programmers find it very difficult for CPython, even for python, there are not enough good and secure sandbox.

C. Restrictions in design

Even the most loyal developers of Python know that because Python is dynamically typed, it has some design restrictions. Consequently, more errors arise only during runtime and thus, it requires more testing. This can be frustrating, especially for scientists that are testing different models and want to focus more on the algorithm than the actual code. We can use only one thread in Python because of its global interpreter lock.

D. Less Mature and Maintained Packages

There are many toolboxes that are present in MATLAB that are absent in the case of python. Since most of Python's libraries are developed only recently, they may still have some bugs and are being updated with all the documentation. The major reason for its under supported libraries is that most of these libraries are made volunteers that have very little time to spent on supporting and documenting a library. It is better to see if a library is supported actively before we use it for making an application or else we have to spend time debugging the code.

E. Problems in Matplotlib

Matplotlib, the most used plotting library in python, also has some limitations. One of its limitations is the absence of a uniformity for some functions in interfaces. For instance, when one creates a text box using pyplot.annotate or using annotate of the axis, we can use the xycoords keyword to mention that whether the text location is addressed using data or axis fractional coordinates. However, this keyword is absent in the case of the pyplot.text function where only data coords can be mentioned.

F. High Memory Consumption

Python was never a great option for tasks that are memory intensive. Mainly, because of the flexibility of its data-types, its memory consumption is large.

V. AN EXAMPLE

Apart from TensorFlow, there are many frameworks to choose from for deep learning. For e.g. Keras provides a higher-level API, making it easy to experiment, while Pytorch gives the user full control over entire process of model designing and training. There are some cases when ease-of-use will be more important and others, where we would require full access to model design. However, in both the cases, the speed at which the framework trains a model is also very important. To check the performance of the three frameworks, a convolutional neural network (CNN) is used. Each of the three frameworks was trained on VGG16, VGG19, and ResNet50. Table II shows the library and their version we used for comparison.

TABLE II: FRAMEWORK AND THEIR VERSIONS USED

Framework	Version
TensorFlow	1.4.0
Keras	2.1.1
Pytorch	0.2.0

The training was performed using GTX1080 GPU on Kaggle's Dog Breed Identification dataset [14] for 10 epochs. All models were trained on the exact same data, where the same method of data loading & preprocessing is applied. Each model is trained with Adam and SGD optimizers with batch size = 4 and batch size = 16 respectively. Table III and table IV shows the training time for the three frameworks on the three models for Adam and SGD. We can see that there is not much difference in time in case of TensorFlow and PyTorch. However, Keras takes a significantly more time. When the dataset is not very big and the focus is on rapid experimentation, then Keras is preferred due to its simplicity. On the other hand, it is preferable to use TensorFlow or PyTorch when high performance is required.

TABLE III: FRAMEWORK AND THEIR SPEED FOR ADAM OPTIMIZER

Framework	Time (in seconds)		
	VGG 16	VGG 19	RESNET50
TensorFlow	21.41	24.49	10.39
Keras	27.9	31.56	20.2
PyTorch	22.28	24.58	11.35

TABLE IV: FRAMEWORK AND THEIR SPEED FOR SGD OPTIMIZER

Framework	Time (in seconds)		
	VGG 16	VGG 19	RESNET50
TensorFlow	14.41	16.29	7.3
Keras	14.12	16.51	12.41
PyTorch	11.28	12.81	7.38

VI. CONCLUSION

In this paper, we discussed the major reasons for the increasing popularity and demand of python. Moreover, we also saw how the performances of different deep learning frameworks on training a CNN. Even though Python is slower in runtime and has some design restrictions as compared to compiled languages like C or C++, it is preferred by scientists and developers in the field of data analytics, numerical computations and almost all technical domains like, Machine learning, AI, Deep learning, etc. Its open source deep learning libraries such as TensorFlow, Keras and PyTorch have made the process of training a deep learning model very easier. That is why it is first choice of even the topmost companies in the world such as Amazon, Facebook, Spotify and Instagram, that have to deal with enormous amounts of data, for their needs of data processing and its analysis.

REFERENCES

[1] X. Cai, H. Langtangen and H. Moe, "On the Performance of the Python Programming Language for Serial and Parallel Scientific Computations," Scientific Programming, vol. 13, no. 1, pp. 31-56, 2005.

[2] G. V. Rossum, "Computer Programming for Everybody-A Scouting Expedition for the Programmers of Tomorrow," CNRI Proposal 90120-1a, Corporation for National Research Initiatives, 1999.

[3] P. J. Guo. (2014) "Python is now the most popular introductory teaching language at top U.S. universities". [Online]. Available: http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext, 2014

[4] R. Sand. (2012) The slideshare website. [Online]. Available: https://www.slideshare.net/blackducksoftware/open-source-by-the-numbers/

[5] G. Piatetsky. (2017) Kdnuggets website. [online]. Available: https://www.kdnuggets.com/2017/08/python-overtakes-r-leader-analytics-data-science.html

[6] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. Seljebotn and K. Smith, "Cython: The Best of Both Worlds," Computing in Science & Engineering, vol. 13, no. 2, pp. 31-39, 2011.

[7] S. V. D. Walt, S. Colbert and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," Computing in Science & Engineering, vol. 13, no. 2, pp. 22-30, 2011.

[8] W. McKinney, "Pandas: a foundational Python library for data analysis and statistics," Python for High Performance and Scientific Computing, pp. 1-9, 2011.

[9] KM Lukas. (2017) The machinelearningexp website. [Online]. Available:http://machinelearningexp.com/data-science-performance-of-python-vs-pandas-vs-numpy/

[10] Abadi, Martín, P. Barham, J. Chen, Z. Chen, A. Davis et al. "Tensorflow: A system for large-scale machine learning," in OSDI, 2016, p. 265-283.

[11] Bird, Steven, and E. Loper, "NLTK: the natural language toolkit," in Proc. of the ACL 2004 on Interactive poster and demonstration sessions, 2004, p. 31.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort et al. "Scikit-learn: Machine Learning in Python," Journal of Machine Learning, vol. 12, pp. 2825-2830, 2011.

[13] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," in Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, May-June 2007.

[14] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for Fine-Grained Image Categorization: Stanford dogs," in Proc. Computer Vision and Pattern Recognition workshop on Fine-Grained Visual Categorization (FGVC), vol. 2, p. 1, 2011.