Best Programming Language for Constructing Efficient Convolutional Neural Networks

Nathan Lucero

California State University, Stanislaus

Programming Languages

Dr. Melanie Martin

December 18, 2020

Convolutional Neural Networks are a sequence of algorithms that are set to check out underlying relationships between a group of data through a significant set up that imitates how the human brain operates. They are constructed and operated on like neurons in the human body. They operate in an evolving form, in which they can easily adjust to dynamic information and predict the best-intended solutions (Liu et al., 2015). Neural networks have become the root of artificial intelligence and used in security firms, robotics, healthcare, and economic trading platforms. For example: in financial institutions, neural networks are used in market research gathering as well as produce an algorithm for predictions of prices in the stock market. Data security firms use neural networks in the management of market risks and detection of fraud among investors and clients. Essentially, neural networks provide a foundation for deep learning which is a subfield of machine learning. These deep learning neural networks absorb data and train them to recognize patterns within the absorbed data then later predict an output for a new set of similar data (Liu et al., 2015).

More specifically, convolutional neural networks also known as CNN’s are artificial neural networks that are used in image processing. Most CNN’s are constructed to specialize in algorithms that display pattern recognition for the analysis of images. These convolutional layers contain algorithm able to receive input data, transforms it, and output its computed contents, which is later transformed into a next layer. Each of these convolutional layers comprise of a primary component known as filters. Filters vary from one image to another and work in looking for differences in textures, edges, shapes, objects, and other various recognizable patterns within images. A program will then detect/predict this inputted image and output its desired results to the user.

**Components of an Artificial Convolutional Neural Network**

Connections and weights serve as critical factors into developing an artificial neural network (Liu et al., 2015). CNN’s contain a high multitude of connections which provide the necessary input and output data schematics and share them between memory transmission processes. Corresponding weightages of data manipulations vary due to the nature of how the priority is set from the developer. It is imperative to construct CNN’s using passageways that contain multiple connections for both input and output.

Another component attributing to the creation of efficient CNN’s is a well-defined propagation function (Liu et al., 2015). This function is designed to effectively manipulate the input data and appropriately output its correct contents to the output layer. It is then sent back to a predecessor stored within the memory and recalculated to perform more computations for better prediction analysis.

Parameters are fundamental to computer diagnostics but also serve to be an important piece for creating CNN’s. These so called “hyper-parameters”, serve as a boundary setter within any neural network (Liu et al., 2015). Essentially, the values being set by the developer determine the level of learning for the machine and its data to be captured. With each iteration, the number of hidden layers together increase or decrease in size. Therefore, these parameters are set to establish contained environments for code to be analyzed and manipulated within its defined boundaries.

The last component ensuring that a CNN will be constructed, is the process of autonomous program learning. By learning what is meant is the adaptation of the network to the way it handles data. It involves adjustments of the weights in a constructed network to improve the accuracy of the output (Liu et al., 2015). The main reason behind this process is to ensure a zero error-rate which is duly maintained as it later defines the cost function of the learning process.

Establishing what convolutional neural networks are and how they operate, leads this topic to which efficient programming language should be used in the developments of a CNN to achieve a projects goals as effectively as possible.

**C++ Programming Language**

C ++ is one of the three competing programming languages that is deemed suited to solve this highly contested debate of constructing efficient CNN’s. Post 2011 C++ establishes features for library developers that will assist in designing a high-quality software solution, in which maintains a simple-to-use and stable product (Roynard et al., 2019). C++comprises of data abstraction, meta-programming, and is also a popular programming language amongst computer scientists. C++ uses a practice called generic programming, “which aims at providing more flexibility to programs” (Roynard et al., 2019). Essentially, genericity means that routines can be applied to a variety of inputs within a library handling a set of routines (Roynard et al., 2019). Developing this generic program will comprise of some techniques in which maintain: usability (simplicity from the end-user), maintainability (simplicity from the library developer stand-point), run-time availability (running routines on images whose kind is unknown until run-time), efficiency (speed and binary size tradeoff) (Roynard et al., 2019).

Among the various techniques that will assist in developing a generic program. Using polymorphisms such as ad hoc, inclusion, and parametric will better our understanding as to why C++ should serve to be used as an effective tool to constructing CNN’s. Polymorphism simply means having many forms. Typically, scientists define polymorphism as the ability of a message to be displayed in more than one form (geeksforgeeks, 2020).

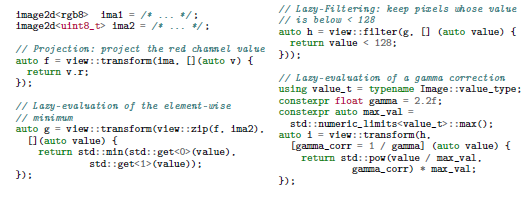
Ad hoc polymorphism is the brutal force solution in C++ programming that enumerates parameters with each of its combinations (Roynard et al., 2019). This means there contains one implementation for each pair of object types. Whether it be it images, structuring elements, or whichever pair of process, inevitably the process results in the duplication of code. The implementation of ad hoc polymorphism requires both a compile-time selection and a run-time selection which is accomplished through a routine overload of its parameters (Roynard et al., 2019). The difference between compile-time and run-time is that compile-time is addressed with routine overloading, which is automatically handled by a computer’s compiler. Once the parameters are known at run-time, the user has to select the right implementation by hand (Roynard et al., 2019). However, this would require duplicating many routines and lead to maintenance issues (Roynard et al., 2019). Image compilation times will be very much dependent on the amount of duplicated data and whether the processed data will be accessed at run-time or compile-time.

Inclusion polymorphic type programs are developed at run-time and relied on by using a strategy called Object-Oriented Programming. Object-oriented programming has several advantages. One advantage is that it is faster and easy to compile on compatible integrated designed environments meant to handle C++ code (w3schools, 2020). It also provides an organized structure for the programs, it helps with minimizing redundancies, and speeds up development time by using shorter lines of code to write programs (w3schools, 2020). Lastly, OOP makes the written code easier to maintain, modify and debug (w3schools, 2020).

Inclusion C++ polymorphism can provide a program’s changing environment with a profound amount of versatility. Whereby the types of images handled, remain unknown until the execution of the program. This flexibility comes at the cost of degraded performances due to the dynamic dispatches of data (Roynard et al., 2019). On the contrary, parametric polymorphism provides better performance due to its static features, which remain known at compile-time.

After encapsulating ourselves with how C++ handles the data. We can thereby explain two algorithms in C++ that will help construct a CNN. The algorithms provided consist of a dilation algorithm and a gamma correction algorithm. In a dilate algorithm there contains a function that simply converts image pixels to either black or white. This function more specifically detects nearby pixels and converts them depending on their pixel color value (Roynard et al., 2019). This color value is registered by the fundamental RGB (Red, Green, Blue) color model. With (0,0,0) being complete black and (255, 255, 255) being completely white. The gamma correction algorithm is used to control the overall brightness of an image’s pixels. Gamma encoding of images is used to optimize the usage of bits when encoding an image (Poynton, 2003). Developers utilize gamma correction by adjusting pixel colorations that are responsible for the output an actual image, with different tones of color.

Below is figure one, which will demonstrate the implementation of the two previously described algorithms. Within the image contains a transform function, which applies a transformation “f” on each pixel of the inputted image (Roynard et al., 2019). These transformations either rotate, translate, or scale and image to specified dimensions which will allow the two algorithms to function properly. Lazy fileting contains the necessary code which keeps the pixels to be converted in an array (Roynard et al., 2019). These values are kept within a domain to be used in the later stages of the program. Lazy evaluation will simply correct the image’s coloration which will allow the outputted image to become more easily visible. This example demonstrates a generic algorithm with little to zero overhead. Here also, neither dense memory allocations nor intricate computations are performed; the image has just recorded all the operations required to compute its values (Roynard et al., 2019). This example also shows the complex methodologies C++ has to offer along with its library components, making this programming language appropriate for handling image processing solutions such as as the CNN.



*(Figure 1: A Demonstration of the Dilate Algorithm and Gamma Correction Algorithm )*

**Java**

People will use Java for its efficient implementation, easy-to-learn language, and fast computation times when building complex convolutional neural networks. This is because it is simple to run, extremely understandable, and to top it all up is it is an object-oriented programming language. This OOP language deals with the high-end features that are needed in most neural network projects. It is also easily portable and is inclusive in in-built garbage collection (Sage and Unser, 2003). The running of problems and queries on it is also simple and easy to understand in case of a change of the workers handling it. To top it all up, Java as a programming language allows the scalability of its programs and is among the best languages used in featuring Al projects. The widespread use of imaging, facilitates an ever-pressing need to train engineers who are proficient with this new technology. The extensive use of imaging promotes an ever-pressing need to prepare engineers with this new technology. This trend is destined to continue as personal computing power continues to grow. Hence, making these complex image processing algorithms more convenient to a broader user base should increase the number of imaging programs (Sage and Unser, 2003).

Java is capable of constructing a simple data processing chart table called a computer vision application (CVI). The CVI runs as follows: first, it builds the network, then loads the weights, acquires the images, detects the edges, converts raw data, shows the output, and finally closes down (Sheridan, 2002). The first stage building the network refers to the building of the neural network that will later be used in the sixth stage of presenting to the net. The third stage used the TWIAN API from Java for it to get to the information from a Webcam using Universal Service Bus that is connected to the system (Sheridan, 2002). Stage four’s work is to extract an image and perform edge analysis/computations from the acquired image that was detected in the third stage. The image will later be converted back to raw data after production in stage 5. Once it is produced, it will be written into the ‘edge.trn’, which is the necessary file to see if the stages were successfully executed accurately.

The sixth stage is where the information on the ‘edge.trn’ file will be acquainted to the network. This operating network will strictly be dealing with one image at a time. It will take each one of the inputted values then load it with the appropriate input unit to the appropriate value (Sheridan, 2002). Later the system will feed the values that were inputted forward and then indicate them on the output nodes.

The last and final of these stages is the seventh stage where the network is completed and it will feed-forward cycles the program, ensure each output node has to correct value, and see the highest. The moment the highest value gets detected, the running program will notify the programmer of the value result, so that they can choose to either test the network once again by getting another image or quit the application (Sheridan, 2002).

Java utilizes JAI (Java Advanced Imaging). JAI is a collection of classes that authorize useful image processing features that are implemented into Java applications. More specifically, JAI incorporates image processing features such as: logical image operators, filtering, edge extraction, image interpolation, transformations, I/O of the image, statistical program operators, tiling of images, and more (Sheridan, 2002). Employing filters to an image allows developers to change the image’s pixel colors, change the image contrast, brighten or darken the image; additionally, they can blur an image. Blurring is vital since it can benefit the recognition phase results by reducing the amount of "noise" in an image (Sheridan, 2002). Another particular type of filtering is edge extraction, which can be applied to an image to find sharp changes in pixel intensity values (Ireyuwa, 2013).

Filtering, edge extraction, and the other features work together in developing and processing of any raw data into the information you may require effortlessly (Sheridan, 2002). It is also in the best interest of the developers to note that there are limited parameters in which they can manipulate and edit the gathered data when it comes to generating convolutional neural networks created within a Java application. These limitations include: adjustments of Webcam parameters, color analysis boundaries, establishing better raw data conversion metrics, the addition of more complete training sets, input image orientation schematics, implementing various types sophisticated neural networks, optimization of learning parameters, dealing with multiple images, and experimenting of programs with other domain avenues (Sheridan, 2002).

In Sage and Unser’s 2003 publishing *Teaching image-processing programming in Java* (Sage and Unser, 2003)*.* They insist that Java has the greatest potential in mass learning of image processing. They extend this claim by providing a “student-friendly” graphical software user interface in which they call, “ImageAccess”. Integrated with a freely available comprehensive image processing library called, “ImageJ”. ImageAccess “simplifies and robustifies the access to the pixel data without having to worry about technicalities and the interfacing with ImageJ” (Sage and Unser, 2003). The primary reasons they utilized the Java programming language was because it was established to be neutral (well adapted to diverse communities), free, provides convenience for remote learning, great at finding and fixing bugs or crashes, easy to learn, and exceptionally fast when handling convolutional filters in which students get an immediate feedback from their program (Sage and Unser, 2003). After installing this developed lab for their students, they found out that their image processing application written in Java kept their students engaged. Conversely, their students found the implementations, guides, and tutorials to be organized, understandable, and convenient.

**Python**

Python is the last of the three functional programming languages for developers whose goals align with developing efficient, professional, and flexible programs for convolutional neural networks. To illustrate Python's effectiveness, a 2019 article titled, Efficient development of high-performance data analytics in Python Future Generation Computer Systems (Álvarez et al., 2019) will help us understand why Python proves to be a powerful programming language which can be applied to not only image processing but to other big data projects.

“The method of obtaining resourceful information for huge volumes of data is known as big data analytics (BDA)” (Álvarez et al., 2019). BDA includes transforming data by using different operations, such as sorting, aggregating, or filtering. In addition to these operations, BDA also utilizes machine learning algorithms to obtain new information and discover data patterns within data (Álvarez et al., 2019). Big data analytics include some challenges, including eliminating data redundancies, representing structured data efficiently, communicating large quantities of data effectively, storing the data, and building vast amounts of data in a scalable or fault-tolerant manner (Álvarez et al., 2019).

The 2019 article claims to remedy some of these BDA challenges with PyCOMPS. PyCOMPS is a task-based programming model, compatible with high-performance cluster (HPC) infrastructures. HPC infrastructures enable developers to write machine learning algorithms from scratch and utilize the Python programming language to parallelize the composed code (Álvarez et al., 2019). The term parallelism refers to techniques that make programs run faster by performing several computations simultaneously. Hence, PyCOMPS applications are standard Python applications with specific annotations that support the exploit of run-time parallelism (Álvarez et al., 2019).

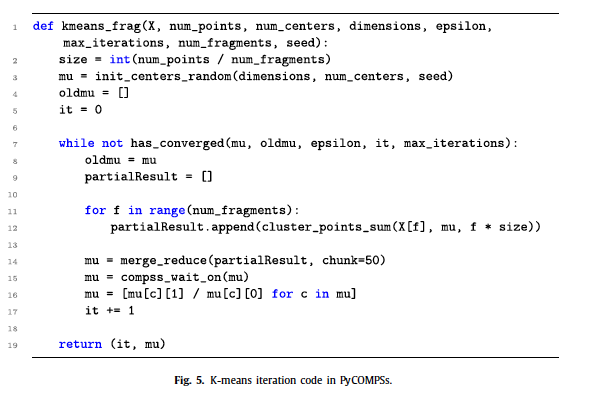
The advantages of using PYCOMPS include: its encompassed by a plethora of accessible mathematical libraries, promotes a highly productive language that is extensively used within the scientific society, organizing a programming model for rapid development within analytic data algorithms, offers live monitoring, proficient at managing different infrastructures without excessive code modifications, relevant for parallel data applications, serializes data to a disk which allows PyCOMPS to handle big collections of data, and lastly, it uses a first-in-first-out scheduling policy that maximizes data locality, (Álvarez et al., 2019).

In order to test the validly of Python being an efficient programming language for CNN’s the article incorporates two machine learning algorithms called, “The K-Mean Algorithm” and the “C-SVM algorithm” or “SVM algorithm”. The K-Mean Algorithm is a popular clustering centroid model. A cluster refers to the process of collecting data points which are grouped together because of their similarities (Garbade, 2018). Using a K-Mean algorithm is typical to define a target number k, which refers to the number of centroids you need in the dataset (Garbade, 2018). The centroid is a dimension representing the center of the cluster. Essentially, the K-means algorithm identifies a K number of centroids and then allocates every data point to the nearest cluster while preserving the centroids as small as possible (Garbade, 2018). The ‘means’ in the K-means refers to averaging of the data; that is, finding the centroid (Garbade, 2018).

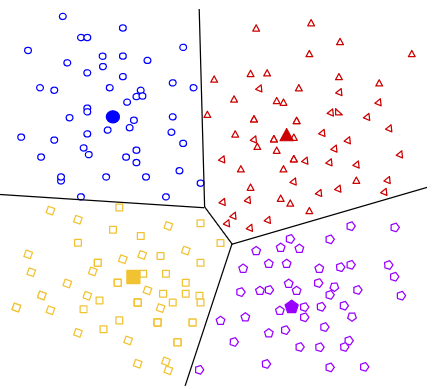
Álvarez and his fellow colleagues explain how the K-Means algorithm is deployed in Python. Figure 2 provides the demonstration of the algorithm at work. In their paper Efficient development of high-performance data analytics in Python Future Generation Computer Systems (2019):

“Variable X represents the input dataset. In order to generate the input data in an organized manner the code uses the “generate\_fragment” tasks, X contains a list of future objects that represent these sectioned objects. On line 3 the application iterates until a fixed number of iterations or until convergence on line 7. In each iteration, the application computes the partial sums of each partition given the current centers on lines 11 and 12 (calculates cluster points). To achieve this, the application on line 12 then generates a “cluster\_points\_sum” (add up all the cluster points) task for each partition. Then, the application merges together the result of the “cluster\_points\_sum” tasks in a reduction process on line 14. The “merge\_reduce” function iteratively calls the “merge\_reduce\_task” task until a single dictionary of centers, number of vectors, and sum of these vectors is left. Finally, the application divides the sum of the vectors by the number of vectors to get the new mean of each center on line 16” (Álvarez et al., 2019).

Figure 3 provides an accurate illustration of what the program is calculating and generating.



*Figure 2: K-Means iteration code in PyCOMPS*



*Figure 3: Centroid Based Clustering (DOES NOT PERTAIN TO CODE ABOVE!)*

*(*[*https://developers.google.com/machine-learning/clustering/clustering-algorithms*](https://developers.google.com/machine-learning/clustering/clustering-algorithms)*)*

The SVM algorithm is a complex classification algorithm. Classification algorithms include an administered machine learning procedure in which the computer program acquires data from the user's input and then manages this information data to analyze new recognizable patterns (Sidana, 2017). Building the decision function is called training, and the process of categorizing unknown vectors is called prediction (Álvarez et al., 2019). Thus, the SVM algorithm is constructed to mathematically formulate a hyperplane that classifies all training vectors in two classes (Álvarez et al., 2019). The actual written code implementation of the algorithm can be found within the article. What is important is the extraction and manipulation of large quantities of data using Python in an effective and efficient way.

This 2019 article finally derives the code complexity for the K-Means algorithm and the SVM algorithm using three different methods: source lines of code, cyclomatic complexity, and NPath complexity. The source lines of code method provide an estimate of a given program's complexity by simply counting its number of code lines. More specifically, they counted the program's number of text lines, excluding the comments (Álvarez et al., 2019). The cyclomatic complexity method was computed by totaling the number of logical conditions, such as "if" and "while" statements, plus one (Álvarez et al., 2019). The last method for determining a program's complexity was the NPath method. NPath defines the number of tests required to cover all possible outcomes (Álvarez et al., 2019). K-means is a highly efficient algorithm, with the best and worst case of θ(m·n) complexity, where m is the number of centers and in the number of input vectors (Álvarez et al., 2019). In contrast, the SVM algorithm's time complexity follows either O(n2) or O(n3).

This extensive evaluation of the PyCOMPSs programming model presents developers the knowledge on how to write big data analytic algorithms with more functionality and less complexity by using Python. Their analysis showed that PyCOMPSs establishes a useful programming model, achieved successful performance metrics in high-performance cluster environments, is an exceptional option for producing large and sophisticated big data analytic programs, and produces excellent trade-offs in performance, flexibility, and productivity (Álvarez et al., 2019).

**Conclusion**

After conducting an extensive overview on the C++, Java, and Python. Java is the best-suited programming language for constructing a CNN for beginners. This process of machine learning is synonymous with deep learning which allows the creation for any neural network. Hence, constructing neural networks with the Java supports an easy to learn library database, a graphical user interface tool that majorly supports creating, saving, and training neural networks. It is mini, skillfully documented, user friendly, and very flexible neural network framework (Ott et al., 2018).

This neural network framework supports many types of facets, including a multilayer perceptron. This is where every layer has potentially a different or fixed number of neurons in it. Java application programming interfaces (APIs) are straightforward for constructing various types of programs. Java APIs provide developers easy to read code which will maintain a higher quality program since understandability is a critical aspect of coding any program especially more complex one such as convolutional neural networks. Java APIs assist in the production stages of development which highly comprise of detecting syntactical errors for a program to build and compile effectively. Java APIs also assist in the modification of written code to maintain an efficient running program.

Java is an object-oriented programming language that aims at enabling the provision of all the quality specs that are very essential to work on AI executions. It's easily movable, and provides in-built garbage collection. The popularity of Java is spread widely around the world. This diverse Java community provides an advantage since there is plenty of documentation that can be found. Here also there are plenty of individuals that can be of assistance to help in any future project in Java.

Based on its capabilities, Java is well suited for convolutional neural networks since it provides the most suitable libraries to use. Predefined libraries such as JAI and ImageAccess offer a very effective graphical user interface for creating, saving, and training convolutional neural networks.

However, in terms of efficiency Python should be the recommended programming language for constructing efficient convolutional neural networks. Like Java, Python’s has a clearly defined and understandable syntax, which serves well for faster developments and allows the developer to effectively test algorithms. Python (like Java) has a strong community of individuals that are willing to share their code as well as help those who are in need of assistance. These communities list freely available source code to online databases as well as answer questions that aspiring developers or even established developers may have.

Some Python libraries include: TensorFlow (used heavily in developing neural networks), Pandas, Scikit learn, Keras and more. Pandas offers fast execution speeds and vast amounts of data engineering features for performing basic data analysis. NumPy is a fundamental package needed for high-performance data analysis and scientific computing. Lastly, scikit-learn supports plenty of machine learning algorithms like decision trees, linear and logistic regression, k-means, clustering, and more. It makes the implementation of complex tasks like feature selection, transforming data, ensemble methods etc., possible within a few lines. These Python libraries provide a foundation in which developers don’t have to code them from scratch.

Python also offers a wide variety of platforms in which allow for the programming language to be handled. These systems include: Macintosh, Windows, Linux, Solaris, and many more systems. What makes Python even more resourceful is its extensible nature which allows the written code to be integrated with other languages such as C/C++, Java, .NET components, and more.

Modern society is generating an increasing amount of data and accelerating its production at unprecedented volumes, variations, and speeds. This pertains to the numerous types of research areas, such as robotics, physics, astronomy, economics, and medicine. All of which contain crucial large-scale data processing. With the evaluation of PyCOMPSs as a task-based programming model for Python, has shown to be an attractive model for organizing big data processes in high performance cluster infrastructures. Since Convolutional Neural Networks are considered to be big data problems. Python has shown to be more than capable at constructing, modifying, and managing efficient CNN’s.

**References**

Álvarez Cid-Fuentes, Javier & Álvarez, Pol & Amela, Ramon & Ishii, Kuninori & Morizawa, Rafael & Badia, Rosa M. (2019). Efficient development of high-performance data analytics in Python. Future Generation Computer Systems. 111.10.1016/j.future.2019.09.051.

Cubuk, E. D., Zoph, B., Schoenholz, S. S., & Le, Q. V. (2017). Intriguing properties of adversarial examples. *arXiv preprint arXiv:1711.02846*.

Charles A. Poynton (2003). [Digital Video and HDTV: Algorithms and Interfaces](https://books.google.com/books?id=ra1lcAwgvq4C&q=gamma-encoding&pg=RA1-PA630). Morgan Kaufmann. pp. 260, 630. [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [1-55860-792-7](https://en.wikipedia.org/wiki/Special:BookSources/1-55860-792-7).

C++ OOP (Object-Oriented Programming). (2020). Retrieved 11 November 2020, from <https://www.w3schools.com/cpp/cpp_oop.asp>

Garbade, M. (2018). Understanding K-means Clustering in Machine Learning. Retrieved 13 November 2020, from <https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1>

Igbinosa, Ireyuwa. (2013). Comparison of Edge Detection Technique in Image Processing Techniques. 25. 2012-13.

Liu, Tianyi & Fang, Shuangsang & Zhao, Yuehui & Wang, Peng & Zhang, Jun. (2015). Implementation of Training Convolutional Neural Networks.

Ott, J., Atchison, A., Harnack, P., Best, N., Anderson, H., Firmani, C., & Linstead, E. (2018, May). Learning lexical features of programming languages from imagery using convolutional neural networks. In *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)* (pp. 336-3363). IEEE.

Polymorphism in C++ - GeeksforGeeks. (2020). Retrieved 13 November 2020, from https://www.geeksforgeeks.org/polymorphism-in-c/

Roynard, Michaël & Carlinet, Edwin & Géraud, Thierry. (2019). An Image Processing Library in Modern C++: Getting Simplicity and Efficiency with Generic Programming.

Sage, D., and Unser, M. (2003) "Teaching image-processing programming in Java," in IEEE Signal Processing Magazine, vol. 20, no. 6, pp. 43-52, doi: 10.1109/MSP.2003.1253553.

Sheridan, Stephen (2002) "A Java Framework for Computer Vision," The ITB Journal: Vol. 3: Iss. 2, Article 6. doi:10.21427/D7845Z

Sidana, M. (2017). Intro to types of classification algorithms in Machine Learning. Retrieved 13 November 2020, from <https://medium.com/sifium/machine-learning-types-of-classification-9497bd4f2e14#:~:text=In%20machine%20learning%20and%20statistics,learning%20to%20classify%20new%20observations.&text=Linear%20Classifiers%3A%20Logistic%20Regression%2C%20Naive,Support%20Vector%20Machines>