Planning and Designing The Development of Software Library for Deep Learning

Oktavia Citra Resmi Rachmawati  
Electronic Engineering Polytechnic Institute of Surabaya

Ali Ridho Barakbah (✉️ ridho@pens.ac.id)  
Electronic Engineering Polytechnic Institute of Surabaya

Tita Karlita  
Electronic Engineering Polytechnic Institute of Surabaya

Research Article

Keywords: Software Library, Deep Learning, Software Design, Software Architecture, Software Process

Posted Date: July 18th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3148716/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Additional Declarations: Competing interest reported. We would like to submit a competing interest statement related to our article entitled 'Planning and Designing The Development of Software Library for Deep Learning'. The lead author, Oktavia Citra Rismi Rachmawati, received research funding from The Electronic Engineering Polytechnic Institute of Surabaya (EEPIS) for the development of a software library related to deep learning in the form of a scholarship. Although the college has no direct connection to the research, the funds are used to support research costs, such as laboratories and supporting facilities. We would like to emphasize that The Electronic Engineering Polytechnic Institute of Surabaya (EEPIS) has no influence over the research design, data analysis, or interpretation of the results. We recognize the importance of transparency in our research reports and want to ensure that readers and reviewers have this information to assess the potential influence of these competing interests on the integrity and conclusions of our research.
Planning and Designing The Development of Software Library for Deep Learning

Oktavia Citra Resmi Rachmawati¹, Ali Ridho Barakbah¹*,
Tita Karlita¹

¹Department of Information and Computer Engineering, Electronic Engineering Polytechnic Institute of Surabaya, Jl. Raya ITS, Surabaya, 60111, East Java, Indonesia.

*Corresponding author(s). E-mail(s): ridho@pens.ac.id;
Contributing authors: oktaviacitra@pasca.student.pens.ac.id;
tita@pens.ac.id;

Abstract
In recent years, deep learning has achieved noteworthy success in various applications. The existing DL libraries widely used by researchers are too complex and challenging to modify the code functions with the custom mathematical formula for research purposes. Thus, we propose a software library related to deep learning that facilitates researchers and practitioners in academia and industry to mutate the training model function with the specific equation. In this paper, we address preparation before developing a software library for deep learning to inspire the researcher who wants to develop a software library. They can utilize our research methodology step by step to conduct the software library architecture design process. We perform four main phases with different activities that execute sequentially to construct a software library intended for deep learning implementation into code practically, and we produce 11 documents to facilitate the transfer of detailed knowledge from us as the pioneer developer of a software library for deep learning that will be part of Analytical Library Intelligent-computing (ALI) to the following developers.

Keywords: Software Library, Deep Learning, Software Design, Software Architecture, Software Process
1 Introduction

Machine Learning (ML) is a subarea of Artificial Intelligence (AI) that enables computers to acquire knowledge without being explicitly programmed[1]. Machine learning has an emerging branch that has rapidly researched and become one of the most popular algorithms known as Deep Learning (DL)[2, 3]. DL is more powerful and flexible because it can handle many features at once when it works with unstructured data, thus making DL became popular due to the increase of high-performance computing facilities and overcoming the vast amount of data reliably[4].

High accuracy and advanced performance when designing to complete various tasks obtained by DL models, such as image processing[5], natural language processing[6], and disease diagnosis[7]. Thus, DL is widely employed in heterogeneous fields in practice. In recent years, despite significant progress and increased interest in DL model development, there has been an explosion of ML tools to keep up the rapid pace of innovation in research[8]. Many popular libraries, such as PyTorch[9] and Tensorflow[10], aid in the construction and efficient execution of DL. For some ML applications, the existing DL libraries are too complex, making modifying the formula in code functions difficult. As a result, the intention of researchers to modify the contents of some methods using specific formulas to enhance the model’s performance will be rendered ineffective.

By building a new software library related to the DL algorithm that is tailored to specific needs in function modification, it can avoid unnecessary features or functionality that can slow down the performance of the software and is specifically designed to work with the existing infrastructure to avoid compatibility issues and streamline the development process.

We propose a software library to realize the demands of intelligent tools that benefit researchers and practitioners in academia and industry on the implementation of deep learning algorithms practically into the codes that easily modify or add the mathematical formula in the model training to improve the model’s performance in executing the tasks. The software library that we will develop specializes in implementing deep learning algorithms, and it will be part of Analytical Library Intelligent-computing (ALI)\(^1\) as a submodule that complements the other machine learning algorithm modules including Automatic Clustering[11], Hierarchical K-Means, K Nearest Neighbors, and Neural Networks[12].

In this study, we address preparation before developing a software library for deep learning, including planning the cycles during the execution and designing the architecture inside with its details that can help us to keep the focus on a goal. We have the initiative to write this paper because we found many papers that discuss designing software applications on all platforms for various categories, such as iOS[13], Android[14], VR[15], and many others. Meanwhile, the library is also a software product same as the application. Thus, we ensure that explaining the plan and design for the software library related to deep learning will inspire the researcher who wants to develop a software library.

\(^1\)http://ridho.lecturer.pens.ac.id/ALI/
DL has an extensive scope of use in completing various tasks, but Convolutional
Neural Network (CNN) is a type of deep learning model most widely used in hetero-
genous application scenarios, especially for processing data with a grid pattern[16].
Thus, for the first iteration of software library development for deep learning imple-
mentation, we focus on constructing convolutional neural network layers and their
support attributes to solve image classification tasks. We will expand to other Deep
Learning models in the following stages.

To summarize, we offer the major contributions in this study includes: (1) providing
the scientific idea for sequentially planning the stages of software library development
needs unique treatments different from software application development in general,
(2) explain the design technique that periodically determines the software library
architecture and support attributes, and (3) generate the development process that
will be the benchmark for researchers who have the initiative to develop the software
library.

This paper runs as follows. In Section 1, we provide some necessary background for
our examination. In Section 2, we describe the related works to outline our uniqueness
from the existing DL library. Then, in Section 3, we briefly present a detailed explana-
tion of our research methodology sequentially to plan and design the software library
related to deep learning. After that, in Section 4, we show the result of the activities
that we conducted and generate 11 documentation as the comprehensive concept of
a software library that we will develop. Finally, in Section 5, we briefly outline a few
implications of the preceding conceptual analysis for designing a software library.

2 Related Works

Software library for deep learning has been developed by some famous institute either
academia or industry and discussed in several publications.

PyTorch[8] is a DL library based on Python programming language that enables
the instantaneous execution of dynamic tensor calculations with automated differ-
entiation and GPU acceleration. PyTorch resolved the global interpreter lock issue
by meticulously optimizing every part of its execution and enabling its customers to
readily implement new optimization measures. Most of PyTorch’s code is written in
C++ to achieve high performance for training the model, but it is integrated into
the Python ecosystem. PyTorch can run two operators asynchronously, either CPU
or GPU. The optimized C++ code is executed on the host CPU, while the CUDA
stream technology is utilized to queue kernel invocations for the GPU.

Torchreid[17] is a DL library written in PyTorch that implements the most recent
state-of-the-art re-ID CNNs to train a solid baseline model for future study. Torchreid
provides a uniform interface for image and video re-ID datasets, as well as optimized
workflows that enable rapid generation of deep re-ID models. Torchreid provides image
and video datasets as the top-of-base classes with essential functions for sampling,
reading, and pre-processing images. Therefore, Torchreid facilitates the users to define
the new datasets with a few options, such as the image paths, identity, and camera
view labels. On the other hand, Torchreid assists users in constructing a variety of
state-of-the-art CNN architectures created specifically for re-ID, together with model
weights trained on publicly available re-ID datasets. And then, Torchreid facilitates
the users to analyze the results qualitatively with a streamlined pipeline for training
and evaluation of deep re-ID models in the engine modules.

TorchIO[18] is a deep library based on Python programming language to enable
efficient loading, preprocessing, augmentation, and patch-based sampling of medical
imaging software platforms. It can facilitate research to code a processing pipeline
for medical images from scratch. The aim of developing TorchIO as an open-source
toolkit is to make medical researchers focus on DL experiments without building image
processing pipelines from scratch. Thus, TorchIO is compatible with other higher-level
DL frameworks for medical imaging, such as MONAI (Medical Open Network for AI).
Command-Line Interface (CLI) is a tool that can apply a transform to an image file
without using Python and assess data augmentation experimentally before network
training to ensure the preprocessing pipeline design for a given application.

Based on the review of those related works, they have yet to describe the planning
and designing of a software library for DL sequentially with the activities that boost
the base construction. Meanwhile, the initial stage, such as plan and design, is the
critical part that can be the trailblazer for software development to ensure the results
meets the objectives initialized.

3 Material and Method

The development of software libraries involves a systematic approach, incorporating
productivity tools that facilitate progress tracking, hence ensuring that the ultimate
product aligns with the intended specifications and quality benchmarks.

3.1 Phases

In this section, we briefly present the method used in this study to plan and design the
development process of a software library intended for deep learning implementation
into code practically. There are four phases executed in this study and each phase has
different activities as shown in Tables 1.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparation</td>
<td>Idea Generation</td>
</tr>
<tr>
<td></td>
<td>Guiding Question</td>
</tr>
<tr>
<td></td>
<td>Literature Review</td>
</tr>
<tr>
<td>Identification</td>
<td>Extract all terms</td>
</tr>
<tr>
<td></td>
<td>Define the goal</td>
</tr>
<tr>
<td></td>
<td>Equipment installation</td>
</tr>
<tr>
<td>Design</td>
<td>Draw architecture</td>
</tr>
<tr>
<td></td>
<td>Describe classes and attributes</td>
</tr>
<tr>
<td></td>
<td>Work scheduling</td>
</tr>
<tr>
<td>Development</td>
<td>Explore the components</td>
</tr>
<tr>
<td></td>
<td>Construct the functionalities</td>
</tr>
</tbody>
</table>
We describe the definition of each existing phase in the following:

1. **Preparation** is the first phase, wherein matters of the activities we can perform in the following stages enable us to handle it optimally. Thus, this phase consists of the techniques that we commit to getting as much knowledge as possible related to our idea. Besides that, this allows us to generate strategies to overcome those challenges and reduces the risk of failure.

2. **Identification** is the second phase, wherein it clarifies the information sources obtained in the previous phase with the result in the form of goals and limitations. Thus, this phase consists of the ideation that provides the solution to realize our idea. Besides that, this allows us to analyze and evaluate the problem or opportunity by breaking it down into parts that can contribute to the opportunity.

3. **Design** is the third phase, wherein it translates the extracted information into some diagram or other forms of documentation to the benchmark in the next phase. Thus, this phase generates the product blueprint that will develop with the applied principles. Besides that, this allows us to identify the requirements and specifications for the software system, which serves as a blueprint for the development process.

4. **Development** is the phase that requires the longest time to execute because it consists of the actions conducted to achieve the goals that the software system meets the requirements according to the design specifications. This phase utilizes the adjustment Software Development Life Cycle (SDLC) to construct a software library for deep learning as a product to be generated.

### 3.2 Productivity Tools

During the planning and designing process, we use some productivity tools to assist us in setting up goals, assigning tasks, and tracking progress because we are inspired by the utilization of productivity tools to boost the execution of the Agile method[19].

- **Miro board**[^2] is a digital whiteboard with an infinite canvas in the form of an interface that is easy to use, which frees the user to build and develop ideas[^20]. Miro boards provide several advantages to individuals or teams, including greater collaboration, higher productivity, consolidated information, and enhanced creativity. Thus, we gained the convenience of brainstorming the idea production and design principles promulgation.

- **Flowchart Maker and Online Diagram Software**[^3] is a cross-browser web app and cross-platform desktop app to visualize the technical design of software in the form of diagrams such as flowcharts, wireframes, UML diagrams, organizational charts, and network diagrams[^21]. Diagrams.net offers several advantages for the creation of flowcharts and diagrams, including usability, compatibility, and cost-effectiveness, so that software documentation may be created quickly and professionally.

- **Notion – One Workspace**[^4] is a collaboration platform that provides a few templates related to productivity and note-taking for organizations, such as kanban boards, tasks, wikis, and databases[^22]. Notion also can be knowledge management that allows users

[^2]: https://miro.com/
[^3]: https://www.diagrams.net
[^4]: https://www.notion.so
to create a knowledge base of information related to a project. Thus, Notion serves us by tracking progress based on the existing division of tasks.

4 Result and Discussion

This section describes the steps for planning the development and designing of the architecture of the software engineering library related to deep learning that we build. We explain the activities of each phase mentioned in the previous section. Each phase in the development of this software engineering library has different activities with significant roles.

4.1 Preparation

The preparation phase consists of the following activities:

4.1.1 Idea Generation

Idea Generation (Ideation) is a creative process aimed at looking for and producing innovation to figure out original approaches and communicate new thoughts to be solutions related to target problems [23]. We conduct creative thinking to explore the areas we are interested in. Thus, we describe the idea that we desire. Essential questions consist of the primary things to consider and the challenges to face.

At the outset, we conceived the notion of creating a deep learning software library, prompting us to ask: "How to develop a deep learning library". After that, we described what challenges we will face in the future. Through this activity, we can ascertain with clarity and precision the necessary steps to achieve the desired impact on development.
4.1.2 Guiding Questions

Guiding Question is the process of identifying the questions that outline essential knowledge to discover analysis of the idea for setting the foundation in formulating a viable solution[24]. In this activity, the questions we create will be answered by ourselves, and new questions may exist along the way as information is gathered[24].

![Guiding Resources](image)

**Fig. 2** The result of guiding questions

We answer our exploration questions by gathering various related sources so that we have extensive insight into the recent development of software libraries. Then, we will synthesize the knowledge that we get through this activity in an activity in the next phase called "Extract All Terms".

4.1.3 Literature Reviews

Literature Review is a comprehensive summary of previous research with the relevant topic by creating a synthesis and analysis based on published research documentation such as scholarly articles, conferences, journals, books, and other sources[25]. We created a table containing related works illustrating the research position to simplify comparative analysis between deep libraries with published papers to share their aims.

### Table 2 Research Position

<table>
<thead>
<tr>
<th>Name</th>
<th>Reference</th>
<th>Programming Language</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torchreid</td>
<td>[15]</td>
<td>Python</td>
<td>Person Re-Identification model</td>
</tr>
<tr>
<td>TorchIO</td>
<td>[16]</td>
<td>Python</td>
<td>Processing medical images for deep learning</td>
</tr>
</tbody>
</table>
4.2 Identification

The identification phase consists of the following activities:

4.2.1 Extract All Terms

Extract All Terms are the process of extracting the terms and concepts based on the pieces of knowledge gained by guiding questions and the literature review activity in the preparation phase. From this activity, we produced the concepts of the software library that we will develop in the form of design principles, the constraints, and why we use Java programming language after tackling the topics.

Table 3 The Software Library Concepts

<table>
<thead>
<tr>
<th>Point</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason to Utilize Java</td>
<td>(1) Enables handling large-scale data with faster execution time than Python, (2) It has strong memory management that automatically gets rid of objects, (3) One of the most popular programming languages that gain 9 million developers and 3 billion devices that use it.</td>
</tr>
<tr>
<td>Language Constraints</td>
<td>(1) Design the presentable structure and the clean architecture of the library code, thus the library can be maintainable and easier to test; (2) Develop the library with careful treatment of the codes for optimizing computational function work in the CPU;</td>
</tr>
<tr>
<td>Design Principles</td>
<td>Those are two design principles intended for library users and two design principles intended for library engineers both involved in the distribution of this library get the same benefits. Two design principles intended for library users are the following: (1) <em>Understandable</em> means that users can easily learn to exploit the library for various purposes, (2) <em>Customizable</em> means that users allow adjusting the functions provided to the needs of the study case they are working on. Besides that, there are two other design principles intended especially for library engineers are the following: (1) <em>Maintainable</em> means that engineers can modify the existing code to improve the performance of the function or debug an error on a line according to the latest version of Java programming language, (2) <em>Expandable</em> means that engineers can enlarge the coverage of the library functionalities without destroying the existing code.</td>
</tr>
</tbody>
</table>

Based on the guiding resources we obtained in the previous activity, we synthesized them into points that became the highlights in the plan to develop a software library related to deep learning. In addition, this software library concept will serve as a benchmark during the construction of the lines of code that compose the functions that support the creation of deep learning models for classification tasks.

4.2.2 Define The Goal

Define The Goal describes a measurable and actionable future state of the product that will be developed[26]. We plan a software library for deep learning as the product that can deliver value through the long-term objective. We described the goal of this
software library by mentioning the numerical value to measure the success of the product that will distribute to the community precisely. Besides that, it can facilitate tracking the progress of development easily.

Table 4  The Goal of Software Library for Deep Learning

<table>
<thead>
<tr>
<th>Product Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>We develop a software library for deep learning that will be part of Analytical Library Intelligent-computing (ALI) with the aim to facilitate Indonesian researchers and practitioners in academia and industry for implementing deep neural networks into the codes. We ensure that our software library development can be a successful product if the package has been downloaded by 1000 on 100 days after release</td>
</tr>
</tbody>
</table>

4.2.3 Equipment Installation

Equipment Installation is the process of composing the list of requirements and preparing the tools that need during the development of the software library for deep learning. Most of the pieces of equipment utilized in this study are software. Table 5 demonstrates the data that mention the basic tool needs with their criteria during the research conducted, from the preparation phase until the development defined done.

Table 5  The tools that needed

<table>
<thead>
<tr>
<th>Name</th>
<th>Platform</th>
<th>Type</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miro board</td>
<td>Web</td>
<td>Productivity</td>
<td>No installation required</td>
</tr>
<tr>
<td>Notion</td>
<td>Web</td>
<td>Productivity</td>
<td>No installation required</td>
</tr>
<tr>
<td>Diagrams.net</td>
<td>Web</td>
<td>Productivity</td>
<td>No installation required</td>
</tr>
<tr>
<td>Intellij IDEA CE</td>
<td>Desktop</td>
<td>Programming</td>
<td>Installation required</td>
</tr>
<tr>
<td>Git</td>
<td>Desktop</td>
<td>Programming</td>
<td>Installation required</td>
</tr>
<tr>
<td>StackEdit</td>
<td>Web</td>
<td>Programming</td>
<td>No installation required</td>
</tr>
<tr>
<td>Git</td>
<td>Desktop</td>
<td>Cloud Storage</td>
<td>Installation required</td>
</tr>
</tbody>
</table>

4.3 Design

The design phase consists of the following activities:

4.3.1 Draw Architecture

The library is one of many software products. To build the library must draw a diagram that describes the components inside and the arrangement of those during the design phase, called Architecture Diagram. The architecture of the software is the form that its engineers have given the system[27]. The way that system is broken down into its parts, how they are put together, and how they interact give that structure its
A system with good architecture is simple to comprehend, simple to create, simple to manage, and straightforward to deploy. The ultimate objective is to enhance programmer productivity while reducing system lifetime costs.

Fig. 3 Architecture Diagram

Four main components arranged from top to bottom build ALI architecture. Computation is the component arranged in the lowermost, and it is described as the minor component because computation consists of the functions that calculate the formula in a neuron. Meanwhile, Data is the component arranged in the uppermost, and it is described as the most prominent component because this allows the data to be processed a vast amount.

The following are explanations of the existing components in the architecture diagram: Data is often used to refer to computer-encoded records or recordings, but it is generally used to refer to statistical observations and other recordings or evidence collections. Typically, data are gathered from monitoring a process, evaluating a condition, or establishing a reference for phenomena.

A machine learning model is a mathematical representation of the outcome of a training procedure that has been trained to recognize specific patterns based on prior experience or current data. A computer will train a model on data collection by executing an algorithm that uses the data to reason and learns.

A layer of deep learning is a container that normally receives weighted input, modifies it with a set of often nonlinear functions, and then sends the altered values to the succeeding layer. In a network, the starting and terminal levels are referred to as input and output layers, respectively, while all intermediate levels are referred to as hidden layers.

Computation is a method algorithms use to “learn” information directly from data to adaptively improve their performance as the number of samples available for learning increases. The library provides accelerated computation to facilities that support numerical optimization on generic mathematical expressions, which it uses for training.

4.3.2 Describe Classes and Attributes

We create a class diagram to expand the architecture diagram, which is more detailed with mentioned the classes and the interactions between each class in every component. Class diagram is a visual representation that illustrates the classes and interactions.
between classes that do not change over time in a system\textsuperscript{[32]}. The class diagram illustrates classes, which encompass behaviors and states and their interactions\textsuperscript{[32]}. Class diagram is one Unified Model Language diagram that can be directly mapped to object-oriented programming languages\textsuperscript{[32]}. Therefore, it is utilized extensively by the developer community\textsuperscript{[32]}.

Fig. 4 Class Diagram

Class diagram has been drawn in Figure 4 above shows that it is an expansion of the architecture diagram from Figure 3. It is more detailed with mentioned the classes and the interactions between each class in every component. On the other hand, the directory structure of the library can illustrate clearly in the class diagram.

There is an interface, namely Use Case, in the class diagram drawn in Figure 4. Meanwhile, the use case has a comprehensive definition. Thus, use case is the methodology to gather the user requirements of a system\textsuperscript{[33]}. It must espouse behavior to clarify and expose that behavior to support the system’s intent\textsuperscript{[34]}. To represent that map out the user’s possible interactions with a system, a use case diagram has been drawn.
4.3.3 Work Scheduling

We create work scheduling in the form of a product roadmap and Kanban board to be self-explanatory, ensuring specific tasks that must be completed at a particular period. The vision and direction of product development are mapped out in a product roadmap, which also outlines how a product will achieve a certain set of goals and the effort necessary to get there[35].

Fig. 5 Use Case Diagram

Fig. 6 Product Roadmap

The library is one of the software products that need a roadmap during the development to track the progress and provide time boxing for controlling the deadline of every task. The figure below shows five stages, each task at a specific period. It is sorted by priority based on impact and effort. Development stages have the most extended
A Kanban Board is a visual representation of a work management system that maximizes efficiency by restricting work in progress to drive improvements in throughput\cite{36}. Usually, the Kanban Board has primary states such as “to-do,” “doing,” and “done”\cite{36}.

![Kanban Board](image)

**Fig. 7** Kanban Board

### 4.4 Development

In this phase, we conduct Software Development Life Cycle (SDLC) to build the library, but we modify the steps according to the needs of this study. Due to our limited resources, we just adopted three steps that are the most necessary. Besides that, we also decided on the actions that represent optimal productivity during the development process.

![SDLC in This Study](image)

**Fig. 8** SDLC in This Study

#### 4.4.1 Explore The Components

We conduct the exploration of the components owned by Java due to discovering the components that have high performance. Thus, we can optimize the speed of the functions that we will build by we keep off the components that process slower than the...
others. Processors in a laptop or computer affect the speed performance in executing the computations. Thus, we describe the device that we use to do these experiments because the time that needs to run the programs can be different for various devices.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacture</td>
<td>Apple</td>
</tr>
<tr>
<td>Handset model</td>
<td>MacBook Pro (15-inch, 2019)</td>
</tr>
<tr>
<td>Operating System</td>
<td>MacOS Monterey 12.5</td>
</tr>
<tr>
<td>Processor</td>
<td>2.4 GHz 8-Core Intel Core i9</td>
</tr>
<tr>
<td>RAM</td>
<td>16 GB 2400 MHz DDR4</td>
</tr>
<tr>
<td>Storage</td>
<td>SSD 256 GB</td>
</tr>
</tbody>
</table>

The loop is the fundamental lesson to learn when studying computer programming. Meanwhile, the loop contains the instructions continually repeated until an exact condition is achieved. Thus, the loop becomes important in various tasks when the computation of deep learning is run, such as indexing and calculating cross-correlation. We hold the experiments that compare the speed performance of loop types owned by Java using IntelliJ IDEA CE, which also provides JUnit to facilitate us counting time of the execution time from each loop type. In this experiment, we set 100 as the number of looping limitation.

<table>
<thead>
<tr>
<th>Types of loops</th>
<th>Execution Time (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>For</td>
<td>866</td>
</tr>
<tr>
<td>Do While</td>
<td>820</td>
</tr>
<tr>
<td>While</td>
<td>833</td>
</tr>
</tbody>
</table>

On the other hand, we organize the experiments that involve all types of collections owned by Java because deep learning needs multi-high dimensional when the computation is running. Thus, we try while looping a two-dimensional collection that defines n as the number of rows and m as the number of columns, then we set 10 as n and 10 as m variable.

<table>
<thead>
<tr>
<th>Types of Collection</th>
<th>Execution Time (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>806</td>
</tr>
<tr>
<td>ArrayList</td>
<td>830</td>
</tr>
<tr>
<td>LinkedList</td>
<td>820</td>
</tr>
<tr>
<td>Stack</td>
<td>1010</td>
</tr>
<tr>
<td>HashSet</td>
<td>850</td>
</tr>
</tbody>
</table>
From two experiments, we can conclude that “while” is the fastest loop and “array” is the fastest collection to process. Because of that, to construct the functionalities that build the library, we will utilize “while” and “array” to operate the process of deep learning implementation.

4.4.2 Construct The Functionalities

Recently, we just started the library development by constructing the pieces of the class that has comprehensive functions coherently, such as Convolution. Thus, we still demonstrate the detail of the class named “Convolution” in which two design patterns have been applied as the treatments to write better codes.

The implementation of singleton and builder design patterns not only for Convolution1DLayer class but also for the other class applied, especially the classes indicated as part of the Layer group.

Convolution is an operation on two functions of a real-valued argument: the first argument as the input and the second argument as the kernel [37]. Furthermore, the output is the feature map [37]. To transform the intuition of recognizing patterns corresponding to objects into mathematical form, the library calculates the weighted sum of a pixel with its near neighbors rather than with all other pixels in the image [9].

Design patterns are best practices that have been documented and become standard by experienced software engineers [38]. The design pattern focuses on a particular object-oriented design problem or issue to identify the participating classes and instances, their roles and collaborations, and the distribution of responsibilities [39].

The Singleton Pattern is a programming technique that ensures a class instantiates only one object and provides a global point to access the class [39, 40]. Whereas the Builder Pattern is a programming technique that encapsulates the construction of a class and allows it to be constructed by attributes in steps to create different representations [39, 40]. Due to the implementation of two design patterns mentioned
above in the writing of the codes, calling the function from this library will be legible, as shown in Figure 10 below.

```java
Convolution1DLayer
    .getInstance(new float[][]{
        Filter1D.Kernel1x3.IDENTITY,
        new float[][]{0, 1, 1, 0}
    })
    .usePadding(true)
    .useBias(false)
    .activation(ActivationType.RELU)
    .stride(2)
    .dilation(2);
```

Fig. 10 Example of calling Convolution1DLayer class

5 Conclusions and Future Work

We organized four main phases with different activities in developing a software library for deep learning, which must be executed sequentially. All phases have different aims and results. Preparation, identification, and design include phases that define the software library’s terms. Meanwhile, the development phase implements modified SDLC to construct the functionalities into the code. Thus, we created the table below to summarize the throughput of each phase.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparation</td>
<td>• Idea outline</td>
</tr>
<tr>
<td></td>
<td>• Guiding resources</td>
</tr>
<tr>
<td></td>
<td>• The table of research position</td>
</tr>
<tr>
<td>Identification</td>
<td>• Design principles</td>
</tr>
<tr>
<td></td>
<td>• The constraints</td>
</tr>
<tr>
<td></td>
<td>• Product goal</td>
</tr>
<tr>
<td></td>
<td>• List of tools</td>
</tr>
<tr>
<td>Design</td>
<td>• Architecture diagram</td>
</tr>
<tr>
<td></td>
<td>• Class diagram</td>
</tr>
<tr>
<td></td>
<td>• Use case diagram</td>
</tr>
<tr>
<td></td>
<td>• Product roadmap</td>
</tr>
<tr>
<td></td>
<td>• Kanban board</td>
</tr>
<tr>
<td>Development</td>
<td>• Guaranteed codes</td>
</tr>
</tbody>
</table>

By writing the process of planning and designing the development of a software library for deep learning in this paper, we propose to produce the documentation that contains all activities with their throughput so we can evaluate the process to find the
things that can improve in the next iteration of development and perform the efficient steps going forward.

The software library for deep learning we are currently developing still focuses on constructing convolutional neural network layers and their support attributes to solve image classification tasks. This paper still discusses the design process that we conduct before building it. Because of that, we have yet to explain the adjustment SDLC in detail. Thus, we will update the progress of this development and add what has yet to be explained in this paper because the process has yet to be applied in the forthcoming paper.

References


