Integration of Existing HIS with Deep Neural Network for Predicting Medicine Needs by Using Scheduled Job

I Putu Arya Dharmaadi and I Made Sukarsa
Department of Information Technology, Udayana University, Bali, Indonesia
Email: {aryadharmaadi, sukarsa}@unud.ac.id

***Abstract*—Most hospitals have used a Hospital Information System (HIS) to help doctors, nurses, and staff in giving fast and excellent services. The data recorded by the system, such as diagnoses, medicines, and billing can be processed further to produce useful information. For example, medicine transaction records during a year can be analyzed to find out how many medical materials needed for tomorrow. This information can prevent the hospitals from scarcity of medicines and other medical materials. Considering its convenience and performance, in order to learn and analyze the data, most researchers used Python libraries that are very handy to run complex mathematical calculations. Therefore, this research develop an integration system to combine existing HIS implemented in hospitals, usually written in PHP code, with the need prediction module that has applied a deep neural network model using Python libraries. Rather than allowing direct call schemes, PHP and Python are interconnected through scheduled job schemes. Finally, the integration system is able to run properly and produce accurate prediction without bother HIS implementation.**

***Index Terms*—deep neural network, integration, python, scheduled job**

I. Introduction

Health is a basic need of all humanity. Due to the importance of the need for healthy living, the presence of health support facilities, start from prevention to healing, become an absolute requirement. Support from all institutions providing health facilities such as clinics, health centers, and hospitals is needed to realize the good-quality health care. In addition to providing good facilities and infrastructure, the institutions must provide maximum health care and management. Using Hospital Information System (HIS) can be a good solution nowadays because it certainly help doctors, nurses, and other medical personnel in checking and treating patients [1]. Generally, HIS is developed on PHP platform because it is cheap, reliable, handy, and able to run on various computer machines.

Although HIS has given great contribution, it has some shortcomings. One of them is the deficiency in preparing sufficient stock of drugs and other medical materials to support maximum examination and patient care. If the amount of medicines stock or medical materials is insufficient, the service to the patient will be disrupted and it may lead to fatal consequences for the patient's condition. For instance, doctors run out of disposable needles so that patients cannot get treatment quickly. Considering the importance of the existence of medicines planning, the need medicines prediction system should be provided in the information system. Unfortunately, not many developers offer this functionality on HIS because it will consume high computation resources. Therefore, we need to combine the existing HIS that has been implemented in hospitals with the prediction modules of medicines and medical materials need that run separately on different servers.

In order to create high accuracy prediction modules, this research utilizes neural network model with hidden layer architectures. The model is a complex mathematical calculation on matrix operation. Many researchers has implemented the model on Python programming language because writing mathematical code, especially matrix calculation is very simple and easy on Python. Indeed, this language is designed for writing complex logic with its clean and strict syntax rules. There are many python libraries applying neural network model that have been tested and widely used, such as *TensorFlow*, *Keras*, and so on. By using it, we can save more time to create deep neural network engine from scratch and focus more on the integration problem between HIS that run on PHP platform and prediction modules that are executed on Python platform.

II. Theoretical Background

To facilitate further discussion, we introduce several theories and related works about Hospital Information System (HIS), system integration, deep neural network, and python library.

A. Hospital Information System (HIS)

Hospital Information System (HIS) is an information system used in hospital to collect, store, retrieve, and display all data and information about patient and all hospital resources [1]. The system is an impact of information technology development that grows rapidly which can assist organizations in providing services.

Manuscript received April 24, 2020; revised September 24, 2020.

doi: 10.12720/jait.11.4.271-276

Clinic, finance, laboratory, pharmacy, and nurses are some divisions on hospital utilizing the system; the data saved into the system can be shared and used together among the divisions.

Many institutions focus on studying and improving HIS in order to make the health system become more effective and efficient. [2] tried to identify existing HIS and its problems in 8th health region covering 7 provinces in Thailand. The result shows that more than 90% of primary care units and hospitals that use many vendors to build HIS have an interconnectivity problem that makes users inconvenience. In other hand, [3] investigated cultural and environmental problems that potentially influence HIS adaptation in Kingdom of Saudi Arabia (KSA). The research made a requirement engineering process that can simplify the system adoption from South Korea into KSA.

B. System Integration

System integration is interconnectivity of computing components such as software, hardware, data, and communication so that the different components can work together [4]. The main reason why system integration is used is due to its capability to combine diverse knowledge bases and physical components [5]. Integration is usually needed in complex environment with various platforms and technologies that run alone so that these conditions complicate users when operating it. By integrating it with other systems, a system is able to collaborate and share information without boundaries. Consequently, the users will get competitive advantages without being confused by the complexity of the systems involved. For example, [6] developed an INS/GPS integration system which offered a high correction in vehicle's navigation. Another example, [7] integrated the mobile restaurant information system with the parking management system and GPS navigating technology in order to enhance competition, increase operational efficiency, and performance.

C. Deep Neural Network

Artificial Neural Network, or so called Neural Network (without Artificial), is an engine that works based on the way of brain does a certain job [8]. For example, the engine is asked to classify objects into some categories or predict the next value based on the past values. The basic processing unit of a neural network that produce output based on its inputs is neuron [9]. In the network, output of a neuron is used as the input by other neurons so that it forms the interconnected neurons [10].

In mathematical terms, every neuron is a sum result of every input value multiplying with its weight. The result is then entered into an activation function to produce the output value. Thus, the function to calculate the output value of neuron *k* is:

$$y_k = \varphi\left(\sum_{j=1}^{m} w_{kj} x_j\right) \quad (1)$$

where *x* is the input signal, *w* is the input weight, and *m* is the number of input [8]. The activation function widely used is *tanh(x)* function, but based on latest studies, researchers found that using deep neural network with Rectified Linear Units (*ReLU*) train is several times faster than their equivalents with *tanh* units [11]. The *ReLU* activation function is formulated as follow.

$$f(x) = \max(0, x)$$

Neural network developed basically consist of three layers, namely input layer, hidden layer, and output layer where each layer has a various number of node or neuron [12]. Today, many researchers add the number of hidden layer in order to increase the accuracy of the network. The neural network with many hidden layers is called a deep neural network, as illustrated in Fig. 1. By utilizing many hidden layers, the network is able to analyze complex data to produce high accuracy prediction [13].

D. Phyton Libraries

Python is a very popular programming language used for computational science calculations, as in the field of machine learning [14]. Writing code in python is quite easy and simple, especially in matrix calculations, because reliable libraries have been provided but the writing is very practical. For example, there is a *Numpy* library that is used for array calculations and operations or matrices, the *Scipy* library which is used for linear algebra and statistical calculations [15].

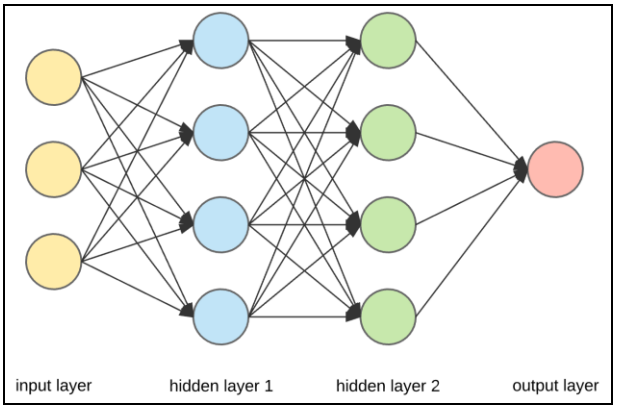

Figure 1. Deep neural network model [16]

To simplify the use of deep neural network model with hidden layer schema, Google's researchers presented *TensorFlow* and *Keras* library [17], [18]. The libraries have become very popular tools for deep learning enthusiasts because they give fast deployment of state-of-the-art deep learning models along with state-of-the-art training algorithms [19].

III. Proposed System

HIS need to call the prediction module written on Python code in order to run and show the prediction results. Generally, developers use the direct call in the form of "*exec()*" function that is usually not active and forbidden from PHP code to execute python file. However, forcing to allow the use of that function probably make our system become vulnerable. For example, an unknown code that was successfully injected by hackers into a PHP file could execute malicious scripts

[20]. Considering the problems, we have to deactivate "*exec()*" function and avoid executing python file directly from PHP. As an alternative solution, we use the database system as a connector media to give information to the Python application about when the modules must be run. We propose a python scheduled job to always check the database to find out if there are new commands created and stored in it.

A. System Overview

In the cloud server running HIS application on PHP Platform, we propose 4 modules for adding the medicines need prediction function. They are *prediction setting module* and *prediction result module* that run on PHP platform, and *scheduled job module* and *prediction module* that run on Python platform. For more details, see the system architectures in Fig. 2 below.

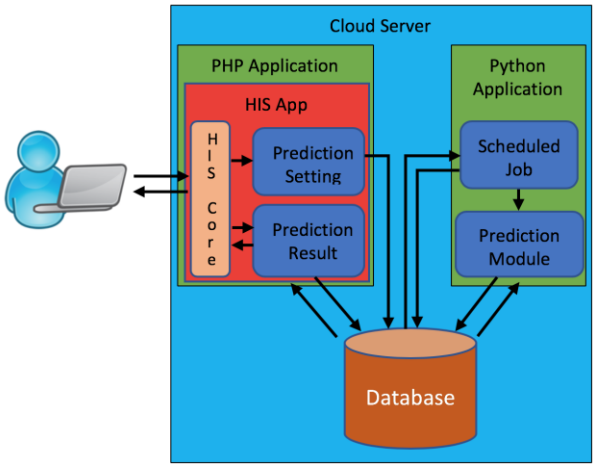

Figure 2. System architectures

The *prediction settings module* serves to get setting data from users, such as at what time the prediction module will be run every day and how much data that will be trained. The setting data is then stored in the database. In order to fetch the outcome of the prediction of medicines need, the user calls the *prediction result module* that runs database queries, get the results, and return it to the user. Both modules are written on PHP code and saved in the HIS project application.

In the other side, in the Python platform, there is a module named the *scheduled job module*, acting to retrieve the setting information from the database and call the *prediction module* based on the setting information. Then, the prediction module is operated by getting the training data (and last weighting values) from the database, processing it on the neural network model, and saving the result in the form of the last weighting values and prediction values. Because there are various types of medicines, the module process it one at a time based on their names instead of simultaneously.

B. Python Scheduled Job

At first, the *scheduled job module* has default setting that has been stored in its variables, such as at 12 am for calling *prediction module* daily and 1 year for training data amount. Before running the *prediction module*, the *scheduled job module* checks the database whether there are new changes for data setting. If so, the module will update its variables with the new setting and wait for a new execution time. If there is no change and the execution time has come, the module will call the prediction module.

C. How It Works

The Python application that runs alongside with PHP-based HIS has a privilege to access and modify the database. When opening HIS through a browser application, the user can open the prediction result page. There the user will see the prediction of medicines need of today and tomorrow. Those prediction results are obtained from the *prediction_result* table in the database. Initially, the table is empty and then it is filled with the output of the prediction module that is called and executed by the scheduled job module. The execution of the prediction module and the writing of its results into the database is run automatically without being commanded directly by the user. Before customized by the user, the setting data has default values, those are "*running_time*: 00" and "*training_data_amount*: 12" which means that the scheduled module will invoke the prediction module every 12 o'clock at night and the prediction module will use the transaction data from yesterday until 1 year ago as the training data. If user want to change the setting, he can open the prediction setting page and update the initial values. The user can stop the operation of the scheduled job by setting the *running_time* value become *null*.

IV. Implementation

In order to give a better understanding, we implement the proposed concept above on simple programming code.

A. Scheduled Job Implementation

This module is the main part of the integration schema. In order to ease the implementation, we utilize the *Schedule* library created by Daniel Bader [21]. The simple implementation written in Python code is as follow in Fig. 3.

```
import ...

#global variables
running_time = "00:00"

def getSettingDataFromDB():...

# create a job function
def scheduled_job():

    # getting the setting data from the database
    running_time = getSettingDataFromDB()

    # create object from PredictionModule class
    p = prediction_module.PredictionModule()

    # run the prediction process
    p.predict()

    # stop scheduled job with old running_time
    schedule.clear('daily-tasks')

    # set scheduled job with new running_time
    schedule.every().day.at(running_time).do(scheduled_job).tag('daily-tasks')

#set scheduled job with initial running_time
schedule.every().day.at(running_time).do(scheduled_job).tag('daily-tasks')

while True:
    schedule.run_pending()
    time.sleep(4)
```

Figure 3. Source code of scheduled job

B. Prediction Implementation

This module is the main part of prediction system. Utilizing the *TensorFlow*, *Keras*, and *Numpy* libraries, we create the simple implementation written in Python code as shown in Fig. 4 below.

```
import ...

class PredictionModule:

    def __init__(self):
        self.num_inputlayer = 14
        self.num_hiddenlayer1 = 15
        self.num_hiddenlayer2 = 15
        self.training_data_amount = self.getSettingDataFromDB()

    def getMedicinesListFromDB(self):...

    def getMedicinesHistoryFromDB(self, idMedicines):...

    def saveDataIntoDB(self,idMedicines,result):...

    def predict(self):
        medicinesList = self.getMedicinesListFromDB()
        for i in range(len(medicinesList)):
            medicinesHistory = self.getMedicinesHistoryFromDB(medicinesList[i][0])
            self.generate_train_data(medicinesHistory)
            X_train = self.normalizeData(self.X_train)
            Y_train = self.normalizeData(self.Y_train)

            model = tf.keras.models.Sequential()
            model.add(tf.keras.layers.Dense(self.num_hiddenlayer1, activation=tf.nn.relu))
            model.add(tf.keras.layers.Dense(self.num_hiddenlayer2, activation=tf.nn.relu))
            model.add(tf.keras.layers.Dense(1, activation=tf.nn.relu))

            model.compile(optimizer="adam", loss="mean_squared_error")
            model.fit(X_train, Y_train, epochs=500)

            prediction_input = self.getLatestTrainData()
            X_predict_result = self.denormalizeData(model.predict(prediction_input))

            self.saveDataIntoDB(medicinesList[i][0],X_predict_result)
```

Figure 4. Source code of prediction module

At first, we fetch all medicines data from the database and save it in an array named *medicinesList*. Because the list contains many medicines, we iterate it for processing every medicine one by one. In every iteration, basically the process consists of 5 stages, namely Data Retrieval, Data Preparation, Model Creation, Model Training, and Result Prediction. Each iteration is run to predict one type of medicine. Hence, the number of medicines variety determines how many loops will be done.

1) Data retrieval

First, we set up data training by taking the daily usage data of certain medications from the database, transform it into a *Numpy* array, and store it in *medicinesHistory* variable. The amount of data that we retrieve depends on the initial setting that we have saved in database.

2) Data preparation

After getting the raw data, we set it up to comply with our deep neural network model. Given the processing time and the accuracy, our model receives 14 data for the input with 2 hidden layers; each of them contains 15 neurons. That means we will predict the next data by processing historical data for the prior 2 weeks or 14 days. Thus, the first 14 raw data will be set as training data and the 15th data is set as target data. Then, the next 14 raw data are taken from the first data until the 15th data, and the 16th data is set as the target data. And so on. Finally, the training data and the target data are normalized so it is worth 0 to 1.

3) Model creation

In order to build a deep neural network model, we utilize *Keras* library. We first create a new Sequential model object, then adding 2 hidden layers and 1 output layer with rectified linear unit (*ReLU*) activation function.

4) Model training

After the model creation is ready, we adjust it with the training data and the target data. Considering our previous experiments, we set the epoch parameter as many as 200 because the model has been stable with the epoch.

5) Result prediction

Training the model will produce the best neurons weight that is automatically saved in internal variables of the model. After that, we can predict how many medication is need for tomorrow by entering the last 14 training data. The result generated is then denormalized to make it true value. Lastly, the prediction data is saved into the database.

V. Result and Discussion

The system that we have proposed above needs to be tested in order to determine its performance.

A. Testing

We checked the performance of the proposed system that has been implemented by running it on a localhost server. Integrating two systems that have different platforms on actual system may result a divergence from the development phase, therefore, we need to check the validity of the integration [22]. We tested it by running several activities as shown below in Table I. The activities were selected because they are the main parts of the integration schema between PHP-based HIS and Python-based prediction system. The results is very well because all function can be operated properly.

TABLE I. Integration Testing Result

	Activity	Result
1	Set up prediction parameter from HIS interface	Run
2	The prediction module run automatically on specific time with specific parameter	Run
3	Get prediction result from HIS interface	Run

Then, we tested the prediction schema by checking the accuracy and the processing time of our model on certain medication. The historical medicines data trained were getting from a small clinic near research location. For example, we entered the daily need of vitamin C for the past 1 month in the prediction system. The device that we used for running it is *MacBook Pro* 2015 with 2,9 GHz *Intel Core i5* Processor and *8GB DDR3* Memory. The device utilized *macOS High Sierra* and *PyCharm Community Edition* 2018 as the system development environment.

While the system was running the training data, we watched resulted error values and processing times. Those values were then collected and shown in the form of line charts in Fig. 5 and Fig. 6 below. After completing the training process, we then executed the prediction process with the same data as the input. The predicted values that were produced by the system were shown in Fig. 7 below and compared with the actual values that we got from the training data. Overall, the results, as shown in the pictures below, are great because the system can produce small error within fast processing time.

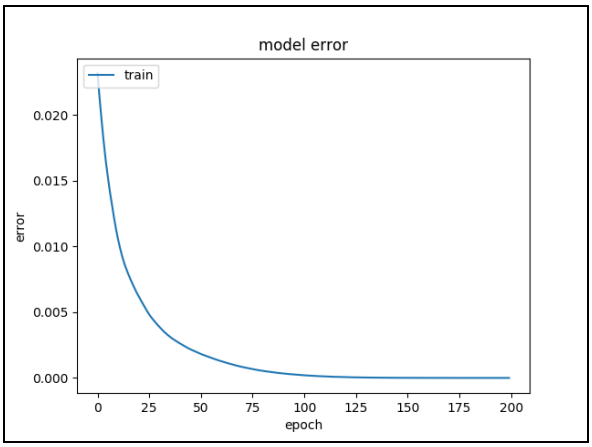

Figure 5. Error per epoch of vitamin C need

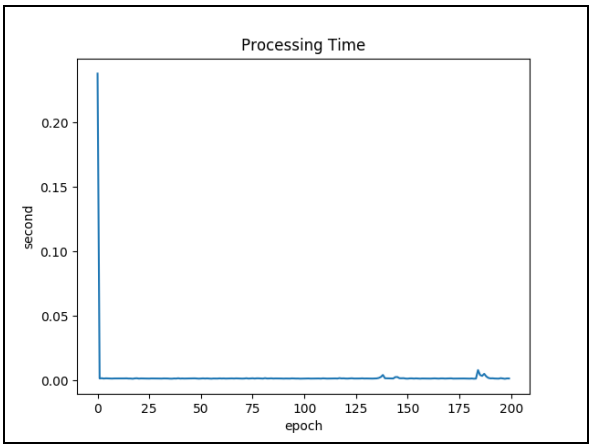

Figure 6. Processing time of training the model

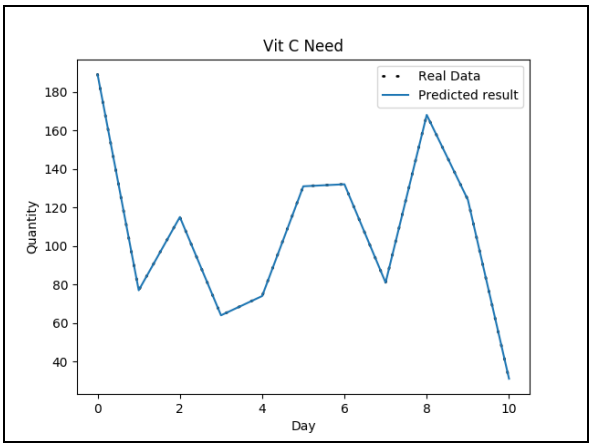

Figure 7. Comparation between predicted value and real value of vitamin C need

B. Discussion

Integration testing results above (shown at Table I) indicate that the existing HIS based on PHP platform is able to execute the prediction module based on Python libraries and get the prediction results without calling directly. This happened because of the *scheduled job module* that runs continuously to check if there is any pending job has not been performed. The pending job based on Python contains commands to execute the *prediction module* according to its parameter stored in the database (see Fig. 3). After executed, the pending job that will be deleted from memory creates a new pending job again with an execution time that has been defined from database, e.g. at 12 pm. So, when the *scheduled job module* finds the new pending job, it can be executed because it is not the time for the execution. As long as the *scheduled job module* keep running and checking the pending job in the background, this integration schema can work properly.

The output from the execution of the *prediction module* is the prediction of the number of certain medical needs that are automatically saved in the database. The *prediction result module* that is part of the existing HIS has known the table and its schema in the database so that the module only fetch the data from the database and show them to users (see Fig. 2). Therefore, the users get the need prediction from the existing HIS without knowing that the prediction job running on the other platform.

The prediction module can generate accurate results because it relies on deep learning model implementation. It has been proven that the model can solve many complex problems, especially that is related to forecasting and classification problems [23]. The presence of *Keras* and *Tensorflow* libraries based on *Python* makes the implementation process of deep learning become easier and simpler, with only a few codes. In addition to free to use, the libraries also provide a lot of complete documentation so that the utilization of deep learning model is more interesting and popular in software developers.

In this study, we set up a deep learning model with 2 hidden layers, each of them contains 15 neurons. The number of these neurons is obtained from the expansion from the input layer containing 14 neurons. The input layer represents the medical needs of previous 14 days or 2 weeks which are used as a base for determining the future medical needs. For more details, see Fig. 8 below. By using this composition, the model is able to predict accurately in a short period of time. This happens because every neuron has multiple weighted connection to link to other neurons. This weight greatly determines the accuracy of deep learning. The more the number of weighted links in deep learning, the better the level of accuracy. However, this also means the processing time is slower.

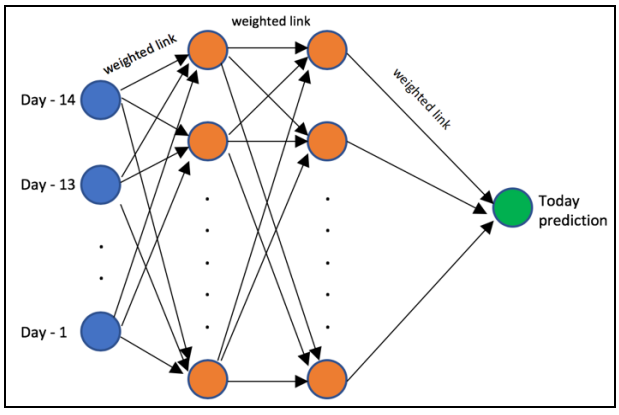

Figure 8. The deep learning model used

VI. Conclusion and Future Work

Based on the issue of scarcity of medicines and other medical materials, we successfully integrated existing HIS application with need prediction module. The utilization of the database management system and the Python-based scheduled job as the integration media makes users get useful outlook about medicines need without need to create prediction module from scratch. When tested on a real system, the integration schema run properly with high accuracy. Further, the system needs to be examined to test its reliability by executing as many predictive processes simultaneously to be able to analyze how robust the proposed system is.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

The first author is the main researcher who conducted the study and wrote the paper while the second author acts as the supervisor who reviewed the result of the research.

Acknowledgment

This research publication was funded by the Faculty of Engineering, Udayana University.

References

[1] N. Izzatty, N. Hazana, and A. Shamsuddin, "Adoption of Hospital Information System (HIS) in Malaysian public hospitals," *Procedia - Soc. Behav. Sci.*, vol. 172, pp. 336-343, 2015.

[2] P. Soontornpipit, C. Taratep, and W. Teerawat, "The study of hospital information systems in the 8 th health region," *Procedia - Procedia Comput. Sci.*, vol. 86, pp. 252-256, 2016.

[3] S. A. Malik, A. Nordin, and R. N. Al-ehaidib, "Requirements Engineering (RE) process for the adaptation of the Hospital Information System (HIS)," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 9, no. 1, pp. 8-17, 2019.

[4] J. M. Myerson, *Enterprise Systems Integration*, CRC Press, 2001.

[5] E. Gholz, A. D. James, and T. H. Speller, "The second face of systems integration: An empirical analysis of supply chains to complex product systems," *Res. Policy*, vol. 47, no. 08, pp. 1478-1494, 2018.

[6] N. Q. Vinh, "INS/GPS integration system using street return algorithm and compass sensor," *Procedia - Procedia Comput. Sci.*, vol. 103, pp. 475-482, 2017.

[7] C. Y. Lo, C. T. Lin, and C. L. Tsai, "Mobile restaurant information system integrating reservation navigating and parking management," *Int. J. Eng. Technol.*, vol. 3, no. 2, pp. 173-181, 2011.

[8] S. Haykin, *Neural Networks and Learning Machines*, third edition, New Jersey: Pearson Education, 2009, vol. 3.

[9] I. Aizenberg, *Complex-Valued Neural Networks with Multi-Valued Neurons*, Springer Science & Business Media, 2011.

[10] S. K. Rogers and M. Kabrisky, *An Introduction to Biological and Artificial Neural Networks for Pattern Recognition*, SPIE Press, 1991.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Adv. Neural Inf. Process. Syst.*, pp. 1097-1105, 2012.

[12] M. A. Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015.

[13] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*, Springer, 2018.

[14] M. Summerfield, *Programming in Python 3: A Complete Introduction to the Python Language*, Addison-Wesley Professional, 2009.

[15] F. Pedregosa, R. Weiss, and M. Brucher, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825-2830, 2011.

[16] D. Arden. (2017). Applied deep learning - Part 1: Artificial neural networks. *Towards Data Science*. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

[17] Tensorflow. Why TensorFlow. [Online]. Available: https://www.tensorflow.org/about

[18] Keras-team. Keras: The Python deep learning library. [Online]. Available: https://keras.io

[19] V. Turchenko, E. Chalmers, and A. Luczak, "A deep convolutional auto-encoder with pooling - Unpooling layers in caffe," *Int. J. Comput.*, vol. 18, no. 1, pp. 8-31, 2019.

[20] Acunetix. (2019). Part 2: PHP security mini guide - Directory traversal & code injection. [Online]. Available: https://www.acunetix.com/websitesecurity/php-security-2/

[21] D. Bader. (2016). Python schedule library. [Online]. Available: https://schedule.readthedocs.io/en/stable/.

[22] B. Schatz and C. Pfaller, "Integrating component tests to system tests," *Electron. Notes Theor. Comput. Sci.*, vol. 260, pp. 225-241, 2010.

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

I Putu Arya Dharmaadi is a lecturer at Udayana University. He graduated from Informatics Magister Program in Bandung Institute of Technology (ITB) in 2015. And he is interested in web programming, android development, computer vision, and machine learning.

I Made Sukarsa is a lecturer at Udayana University. He obtained his Master Degree in Department of Electrical Engineering at Gadjah Mada University in 2003. And he is interested in data warehouse, middleware, and IT governance.