Research Article

Indoor Positioning System in Learning Approach Experiments

Dodo Zaenal Abidin ¹, **Siti Nurmaini** ¹, **Erwin** ², **Errissya Rasywir** ³,
and Yovi Pratama ³

¹Department of Engineering, Sriwijaya University, Palembang, Indonesia

²Department of Informatics Engineering, Sriwijaya University, Palembang, Indonesia

³Department of Informatics Engineering, Dinamika Bangsa University, Jambi, Indonesia

Correspondence should be addressed to Dodo Zaenal Abidin; dodozaenalabidin@gmail.com

Received 8 May 2021; Revised 25 June 2021; Accepted 13 July 2021; Published 3 August 2021

Academic Editor: Yang Li

Copyright © 2021 Dodo Zaenal Abidin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The positioning system research strongly supports the development of location-based services used by related business organizations. However, location-based services with user experience still have many obstacles to overcome, including how to maintain a high level of position accuracy. From the literature studies reviewed, it is necessary to develop an indoor positioning system using fingerprinting based on Received Signal Strength (RSS). So far, the testing of the indoor positioning system has been carried out with an algorithm. But, in this research, with the proposed parameters, we will conduct experiments with a learning approach. The data tested is the signal service data on the device in the Dinamika Bangsa University building. The test was conducted with a deep learning approach using a deep neural network (DNN) algorithm. The DNN method can estimate the actual space and get better position results, whereas machine learning methods such as the DNN algorithm can handle more effectively large data and produce more accurate data. From the results of comparative testing with the learning approach between DNN, KNN, and SVM, it can be concluded that the evaluation with KNN is slightly better than the use of DNN in a single case. However, the results of KNN have low consistency; this is seen from the fluctuations in the movements of the R2 score and MSE values produced. Meanwhile, DNN gives a consistent value even though it has varied hidden layers. The Support Vector Machine (SVM) gives the worst value of these experiments, although, in the past, SVM was known as one of the favorite methods.

1. Introduction

The need for location-based service applications continues to grow, demanding trials and application of positioning systems to continue. The application of location-based services is usually used by business organizations that need new knowledge about user experience and device superiority through the use of location-based services for applications. However, location-based services with user experience still have many obstacles to overcome, including how to maintain a high level of position accuracy [1].

There are two positioning systems, namely, global positioning system (GPS), which is more widely used outdoors, and indoor positioning system (IPS) which is mainly used indoors [2]. Indoor positioning systems generally use sensors and communication technology to find objects in the indoor environment. IPS attract scientific and corporate

interest because there are huge market opportunities to implement this technology. There have been many previous studies on indoor positioning systems. However, they do not have a solid classification scheme which is structural or have a limited perspective. Finally, research is fast becoming obsolete due to the dynamic development of this technology [3].

There are many positioning systems with different architectures, configurations, accuracy, and reliability in terms of Received Signal Strength (RSS) technology such as Wi-Fi Positioning System (WPS) [4–6], RFID [7–9], radio [10, 11], Bluetooth [12–14], cellular (GSM) [10, 15, 16], and television signals [3].

Positioning with location fingerprinting is more likely to be applied because it does not require directions, stations, special hardware, towers, or anything else. However, high calibration time optimization is quite challenging to study.

The fingerprinting used in this research is based on Received Signal Strength (RSS).

Fingerprinting is used to calculate the approximate location. The concept of a fingerprint is training and positioning. In the training phase, a radio map of the observed signal strength values from different locations is recorded. Then, in the positioning phase, the signal strength observed on the user's device is compared to the radio map value using proximity matching algorithms such as K-Nearest Neighbor [17–19], particle swarm optimization [3, 20], Support Vector Machine (SVM) [17], naïve Bayes [21–23], regression [24–26], Deep Neural Network (DNN) [27, 28], and deep learning [29].

Fingerprinting is considered as an estimator that uses measurements of the Received Signal Strength to calculate the position of use that is most likely to solve this problem.

Based on this, this study tries to keep in perspective the technology of the RSS system by testing using a learning approach. The learning approach experiment is proposed using the deep learning and machine learning methods. As is well known, deep learning has achieved extraordinary results in research in the recent years [30]. However, it still needs to be tested with conventional learning methods or machine learning based on Received Signal Strength (RSS).

This research was conducted because there is a significant gap between the results of research related to RSS using one method and another. Research gaps and contributions of the proposed work are discussed by comparison with current methods and recent studies. In this research, the novelty is to estimate the RSS using 3-dimensional coordinates (x , y , and z). Previously, research on position estimation using RSS only used 2-dimensional coordinates (only x and y).

2. Methodology

2.1. Indoor Positioning System Experiment Architecture. The stages of the process are carried out in research so that research can run well, and the goals set can be achieved. In this study, the authors conducted problems formulated include how to apply the experimental learning approaches algorithm for the indoor positioning system, how to test the indoor positioning system with experimental learning approaches, and how to implement an indoor positioning system with experimental learning approaches.

The study about the theories that serve as guidelines and references for solving the problems discussed which is relevant to the problem includes Deep Neural Network, machine learning (K-Nearest Neighbor), Support Vector Machine (SVM), indoor positioning system (IPS), and position evaluation.

Sources of compound image data used are from the service signal data on devices in the Dinamika Bangsa University building. This data will be tested with an indoor positioning system with experimental learning approaches.

We also make an architectural design plan for the indoor positioning system with experimental learning approaches. At this stage, the reference point (RP) position will be determined in the Dinamika Bangsa University building, from

the first floor to the second floor. The number and position of the access points (AP) that will be used follow from the AP that have been installed; there are 9 access points with a basic service set identifier (BSSID) and a service set identifier (SSID) like Table 1.

Determination of the position of the reference point is done from the first floor to the second floor. The following is an architectural image of the position of the object being trained and the position of the access point used on the first floor.

Figure 1 shows a map of the location of the room on the 1st floor. There are 7 indoor spaces used for training data positions as well as an “L” shaped hallway and a lobby positioned by the training object. In this study, we made a reference point position design that is connected to the AP to be tested. In the picture above, there are two different color dots with different functions. The blue dot is the reference point and the red dot is the access point.

In this AP position and reference point architecture, RSS data is taken as a reference point using a laptop. The circle symbol is the reference point position data (as many as 54 pieces) connected to 3 AP to be trained. The following is a design drawing of the reference point and access point positions used on the second floor.

Figure 2 shows a picture of the location plan of the room on the 2nd floor. There are 7 indoor spaces that are used to position the training data and an “L” shaped hallway positioned by the training object. In this study, with the AP position architecture connected to the reference point above, RSS data is taken as a reference point using a laptop. The circle symbol is taken as reference point position data (48 in the indoor and the hall of the building) connected to 6 APs for training and 4 reference points on the stairs that connect the 1st and 2nd floors. In Figure 2, there are two different color points with different functions. The blue dot is the reference point on the 2nd floor of the building used and the red dot is the access point.

Furthermore, we develop an *indoor positioning system*. At this stage, the indoor positioning system was developed using experimental learning approaches. This process involves coding and testing. The system flow is illustrated in Figure 3. In Figure 3, the architectural experiment in this study is carried out by testing 3 algorithms.

The experiment began with collecting the RSS raw data from the Dinamika Bangsa University building. Then, the raw data generated from RSS were to be stored as a receiver module whose results are stored in a dataset. Furthermore, the dataset was tested with the KNN, DNN, and SVM algorithms. Furthermore, the test results are the result of position estimation and are evaluated by several evaluation methods.

The learning approach experiment carried out included three learning approaches, namely, the approach using Deep Neural Network, the approach using K-Nearest Neighbor, and the approach using Support Vector Machine. Testing uses RSS fingerprint technology as an indoor positioning system task. The research system was built to carry out automated testing.

This approach is based on benchmarks for comparison of familiar methods used in RSS research. The use of the

TABLE 1: Access point profile.

Mac address	AP	SSID	Merc	Transmit radio
54:78:1A:89:D6:F0	F11	Hotspot-STIKOM	CISCO AIR-AP 1042N	20 dbm (100 MW)
54:78:1A:89:17:72	F12	Hotspot@stikom-db.ac.id	CISCO AIR-AP 1042N	20 dbm (100 MW)
C2:9F:DB:9D:8C:52	F13	Hotspot@stikom-db.ac.id	CISCO AIR-AP 1042N	20 dbm (100 MW)
54:78:1A:89:CD:30	F21	STIKOM-office	CISCO AIR-AP 1042N	20 dbm (100 MW)
A4:4C:11:EE:5C:8C	F22	STIKOM-office	CISCO AIR-AP 1042N	20 dbm (100 MW)
54:78:1A:88:D2:50	F23	STIKOM-office	CISCO AIR-AP 1042N	20 dbm (100 MW)
54:78:1A:89:06:90	F24	STIKOM-office	CISCO AIR-AP 1042N	20 dbm (100 MW)
A4:4C:11:EE:60:60	F25	STIKOM-office	CISCO AIR-AP 1042N	20 dbm (100 MW)
54:78:1A:89:1A:B0	F26	Hotspot@stikom-db.ac.id	CISCO AIR-AP 1042N	20 dbm (100 MW)

FIGURE 1: The AP position and reference point design on floor 1.

K-Nearest Neighbor method which is compared with the Deep Neural Network and Support Vector Machine is based on the results of previous studies which state that these three methods are the best methods for estimating positions using RSS signals. The selection of the K-Nearest Neighbor algorithm was chosen to represent machine learning based on lazy Learning, which classifies objects based on the learning data that is closest to the object.

The selection of the Deep Neural Network algorithm is chosen to represent network-based machine learning with deep learning.

The selection of the Support Vector Machine algorithm was chosen to represent machine learning based on the Structural Risk Minimization (SRM) principle with the aim of finding the best hyperplane that separates two classes in the input space.

FIGURE 2: The AP position and reference point design on floor 2.

For learning in the introduction and estimation of positions in the indoor positioning system work, we carried out three guided lessons with three machine learning algorithms: K-Nearest Neighbor, Deep Neural Network, and Support Vector Machine. Here we show the source code for this research experiment using the KNN algorithm source code to generate sample data in Python:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import neighbors
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

```
from sklearn.metrics import mean_absolute_error
```

Furthermore, the data that has degenerated is normalized, because the data obtained is raw data that has not yet neat structure and content:

```
def normalisasi (input_data):
    for i in range (0, len(input_data)):
        for j in range (0, len(input_data[i])):
            input_data [i, j] = (input_data [i, j] + 100)/100
def denormalisasi (x):
    for i in range (0, len (x)):
        for j in range (0, len (x [i])):
            x [i, j] = x [i, j] * 100-100
```

FIGURE 3: The architecture of indoor positioning system experiment.

After the data has been normalized, it is transformed into a two-dimensional matrix using the denormalization method (x). The data that will be generated from the process or output data are also subject to normalization with the following source code:

```
def normalisasi_output (output_data):
    #0 for x
    for j in range (0, len (output_data)):
        output_data [j, 0] = output_data [j, 0]/100
```

The above function is a method for storing the width of the building or the max x position where the data is retrieved.

```
#1 for y
for j in range (0, len (output_data)):
    output_data [j, 1] = output_data [j, 1]/100
```

The above function is a method for storing building length or max y position.

```
#2 for z
for j in range (0, len (output_data)):
    output_data [j, 2] = output_data [j, 2]/100
```

The above function is a method for storing the height of the device when retrieving RSSI. Then, the output data from the process results again undergo denormalization as shown in the following source code:

```
def denormalisasi_output (y):
    #0 for x
    for j in range (0, len (y)):
        y [j, 0] = y [j, 0] * 100
    #1 for y
    for j in range (0, len (y)):
```

```
y [j, 1] = y [j, 1] * 100
    #2 for z
    for j in range (0, len (y)):
        y [j, 2] = y [j, 2] * 100
```

The next part is the process of demonstrating the resolution of a regression problem using K-Nearest Neighbor and the interpolation of the target using both barycenter and constant weights:

```
data = "./drive/My Drive/Dataset Pak Dodo/dataset.csv"
dataframe = pd.read_csv (data, sep = ';', header = None, decimal = ",")
dataset = dataframe.values
input_data = dataset[:, 0:9]
normalisasi (input_data)
output_data = dataset[:, 9:12]
normalisasi_output (output_data)
np.random.seed (0)
X = input_data
T = output_data
y = output_data
for i, weights in enumerate (['uniform', 'distance']):
    for n_neighbors in range (1, 101):
        knn = neighbors.KNeighborsRegressor (n_neighbors, weights = weights)
        print (knn.get_params ())
        model = knn.fit (X, y)
        y_ = model.predict (T)
        score = r2_score(y, y_)
        vmse = mean_squared_error (y, y_)
        print (n_neighbors, weights, score, vmse)
```

Next is the source code for recognition and position estimation (indoor positioning system) using a Deep Neural Network. The following is the program code in this research with the Python programming language:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras import backend as act
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Next is data normalization to normalize the format and content of data that is still experiencing various noises by using the method from google.colab import files:

```
def normalisasi (input_data):
    for i in range (0, len (input_data)):
        for j in range (0, len(input_data[i])):
            input_data [i, j] = (input_data [i, j] + 100)/100
```

Next, denormalize the x data or fill in the fields from the data table with the denormalization method (x) such as the following Python program code:

```
def denormalisasi (x):
    for i in range (0, len (x)):
        for j in range (0, len (x [i])):
            x [i, j] = x [i, j] * 100-100
def normalisasi_output (output_data):
    #0 for x
    for j in range (0, len (output_data)):
        output_data [j, 0] = output_data [j, 0]/100
```

The function above is a method to store the building width or the max x position used for IPS with DNN.

```
#1 for y
for j in range (0, len (output_data)):
    output_data [j, 1] = output_data [j, 1]/100
```

The function above is a method for storing the building length or the max y position used for IPS with DNN.

```
#2 for z
for j in range (0, len (output_data)):
    output_data [j, 2] = output_data [j, 2]/100
```

The above function is a method for storing the height of the device when retrieving RSSI which is used for IPS with DNN.

```
def denormalisasi_output (y):
    #0 for x
    for j in range (0, len (y)):
        y [j, 0] = y [j, 0] * 100
```

```
#1 for y
    for j in range (0, len (y)):
        y [j, 1] = y [j, 1] * 100
#2 for z
    for j in range (0, len (y)):
        y [j, 2] = y [j, 2] * 100
```

The following function is a method for inputting layer 18 neurons:

```
model = Sequential()
model.add (Dense (18, input_dim = 18, activation = 'relu'))
```

The following function is a method for hidden layer 500 neurons:

```
for i in range (5):
    model.add (Dense (500, activation = 'relu'))
```

The following function is a method for creating a Deep Neural Network with output layer 3 neurons (x , y , and z):

```
model.add (Dense (3, activation = 'softmax'))
model.summary ()
model.compile (loss = 'mse', optimizer = 'adamax',
metrics = ['mae'])
```

The following function is a method for training and fitting the hard model on the dataset:

```
model.fit (input_data, output_data, epochs = 5000,
batch_size = 50, verbose = 2)
```

The following functions are a method for evaluating MSE (Mean Square Error) and MAE (Mean Absolute Error):

```
mse_value, mae_value = model.evaluate (input_data,
output_data, verbose = 0)
print (mse_value, mae_value)
```

Next is the source code to introduce and estimate position (indoor positioning system) using the Support Vector Machine. The following is the program code in this research with the Python programming language:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

The following function is a method for normalizing data with the following methods:

```
def normalisasi (input_data):
    for i in range (0, len (input_data)):
        for j in range (0, len (input_data [i])):
```



```

        input_data [i, j] = (input_data [i, j] + 100)/
100
def denormalisasi (x):
    for i in range (0, len (x)):
        for j in range (0, len (x [i])):
            x [i, j] = x [i, j] * 100-100
def normalisasi_output (output_data):
    #0 for x
    for j in range (0, len (output_data)):
        output_data [j, 0] = output_data [j, 0]/100

```

The function above is the method for building width or max x position used for IPS with SVM:

```

#1 for y
    for j in range (0, len (output_data)):
        output_data [j, 1] = output_data [j, 1]/100

```

The function above is the method for the building length or the max y position used for IPS with SVM.

```

#2 for z
    for j in range (0, len (output_data)):
        output_data [j, 2] = output_data [j, 2]/100

```

The above function is the method for device height when retrieving RSSI which is used for IPS with SVM. Furthermore, after the data are transformed into a two-dimensional matrix with the denormalization method (x), the data that will be generated from the process or output data are also subject to normalization with the following source code:

```

def denormalisasi_output (y):
    for j in range (0, len (y)):
        y [j, 0] = y [j, 0] * 100
        for j in range(0,len (y)):
            y [j, 1] = y [j, 1] * 100
        for j in range (0, len (y)):
            y [j, 2] = y [j, 2] * 100
data = "../drive/My Drive/Dataset Pak Dodo/dataset.csv"
dataframe = pd.read_csv (data,sep = ';',header = None,
decimal = ",")
dataset = dataframe.values
input_data = dataset[:, 0:9]
output_data = dataset[:, 9:12]
X = input_data
T = input_data
y = output_data
clf = svm.SVR (verbose = True)
regr = MultiOutputRegressor (clf)
regr.fit (X, y)
y_ = regr.predict (T)
score1 = regr.score (T, y)
score = r2_score (y, y_)

```

```

vmse = mean_squared_error (y, y_)
print (score1, score, vmse)

```

Furthermore, after the training stage is carried out, it is followed with the testing phase. The following is the testing phase that will be carried out with the three algorithms used, namely, K-Nearest Neighbor, Deep Neural Network, and Support Vector Machine.

At the testing stage, the preliminary test results are carried out from the results of the indoor positioning system research with experimental learning approaches conducted after the learning approach experiment research activities. In the early stages of this study, an experimental scheme of indoor positioning system research was determined with an experimental learning approach. Reporting is carried out to account for the course of research and its development. At this stage, experimental schemes are needed so that the final results of the study can be analyzed. Making reports from the results of an indoor positioning system research with experimental learning approaches carried out after the research activities produces a system.

Regarding the computational costs of the proposed research methodology, it is quite economical. RSS retrieval at 1 point of collection takes 5 minutes. Meanwhile, there are 54 reference points on the 1st floor of the building and 48 reference points on the 2nd floor. Thus, the overall duration of data collection is approximately $(54 \times 5) + (48 \times 5) = 270 + 240 = 510$ minutes or approximately 8.5 hours.

For the runtime cost of the dataset execution results of more than 59 thousand lines of RSS data with the parameters specified at the top, it takes about 10 to 12 seconds to execute in each iteration. Experiments were carried out to find the best network from 5 types of parameters on deep neural networks, including experiments to find the best optimizer, activation function, number of hidden layers, network layers, and epochs. As an illustration, one iteration does the calculation from the input until it gets the target value.

3. Results and Analysis

3.1. Deep Neural Network Testing. Before analyzing the results of the tests conducted in this study, the RSS data plot was taken from the Dinamika Bangsa University building. This is done to see an overview of the distribution of data used and the distribution of data frequencies at a certain scale to see the relationship between the RSS data.

The RSS data was taken from the Dinamika Bangsa University building's 1st and 2nd floors, with 106 points. For each point, the RSS data were collected within 5 minutes. After being collected, the data taken is only data from 9 access points that have been set. We conducted the preparation of the data into a dataset, such as merging some data field, attribute selection, and normalization. The normalization process is carried out to normalize the data distribution.

From Figure 4, it can be seen that the distribution of RSS data taken from the Dinamika Bangsa University building

FIGURE 4: RSS data plot chart.

on the 1st and 2nd floors contained homogeneous colors piled up for each reference point, indicating that, for each reference point, the RSS value obtained was relatively consistent, so it can be said that the data used in this study can be processed using multivariate regression.

Figure 4 shows a visualization of the RSS plotting used in this study. Each color consists of F11, F12, F13, F21, F22, F23, F24, F25, and F26; the code is a reference point code on the 1st and 2nd floors of the Dinamika Bangsa University building, where reference data is taken. There are 9 access points: 3 AP points on the 1st floor and 6 AP points on the 2nd floor.

In the scenario in Table 2, the DNN architecture is shown. We used the function of ReLU activation on the input layer and softmax on the output layer; we also conducted the test to find the optimum optimizer as the hidden layer. The optimizer comparison is given in Table 2.

Table 2 gives the optimizer comparison experiment. We test ReLU activation function on some optimizer methods such as RMSprop, Sgd, Adagrad, Adadelata, Adam, Adamax, and Nadam. The MSE value of those tests that can be used for next testing is Adamax. The optimizer comparison test results about the optimizer comparison chart can be seen in Figure 5.

In Figure 5, the DNN architecture that is used only consists of the input layer and output layer, with the function of ReLU activation on the input layer and softmax on the output layer; the hidden layer has not been involved, because the focus in this scenario is only on finding the optimum optimizer. The epoch is used with as many as 20 iterations; the loss function used is MSE (Mean Square Error) which is always used for multivariate regression. In Table 2, it can be seen that the smallest MSE value is that of the Adamax optimizer, 0, 042224. It can be concluded that the most optimum optimizer is Adamax.

Table 3 gives the comparison about the activation functions. We conducted the test on some activation methods. Here are the values of MSE from the test conducted.

In Table 3, it can be seen that the smallest MSE value is when the hidden layer uses the ReLU activation function,

TABLE 2: The optimizer comparison.

No.	Activation function	Optimizer	MSE
1	ReLU	RMSprop	0, 042394
2	ReLU	Sgd	0, 043077
3	ReLU	Adagrad	0, 04252
4	ReLU	Adadelata	0, 04226
5	ReLU	Adam	0, 042256
6	ReLU	Adamax	0, 042224
7	ReLU	Nadam	0, 042273

and the results are relatively far when seen in Figure 6. For the comparison of activation function, we use ReLU, linear, sigmoid, softmax, tanh, elu, selu, softplus, softsign, and hard sigmoid, while the output layer only uses softmax.

Figure 6 shows the visualization from Table 3, with epoch = 20 iterations; the optimizer used is Adamax, where the architecture is still the same as the previous scenario, and the activation functions that are tried on the input layer are ReLU, linear, sigmoid, softmax, tanh, elu, selu, softplus, softsign, and hard sigmoid, while the output layer only uses softmax. The number of hidden layers used is only 1, and the activation function used on the output layer is softmax. From the scenario of Table 3 and Figure 6, we get that the smallest value of MSE is 0, 042217 from ReLU activation method. It can be concluded that the optimum activation function to be used is ReLU.

Next, in this scenario in Table 4, we conducted the selection of the number of neurons using the divide-and-conquer scheme, meaning that it is tried with a small number of 5 neurons, followed by a large number of 1000 neurons; then we looked for the midpoint of each distance, where we got 500, 750, and 250. Here is the result of the MSE value with the number of neurons experiments in Table 4.

In Table 4, the optimizer used is Adamax, the activation function at the input and hidden layer is ReLU, the activation function at the output layer is softmax, and the numbers of neurons tested are 5, 1000, 500, 750, and 250, with epochs of 20 iterations in each. From those tests, we got that the smallest MSE is 0, 041755. The smallest value of MSE

FIGURE 5: The optimizer used.

TABLE 3: The activation functions comparison.

No.	Activation function	Optimizer	MSE
1	ReLU	Adamax	0, 042217
2	Linear	Adamax	0, 044014
3	Sigmoid	Adamax	0, 042552
4	Softmax	Adamax	0, 042437
5	Tanh	Adamax	0, 042417
6	Elu	Adamax	0, 042546
7	Selu	Adamax	0, 042527
8	Softplus	Adamax	0, 042666
9	Softsign	Adamax	0, 042359
10	Hard sigmoid	Adamax	0, 042888

FIGURE 6: Visualization of activation function.

TABLE 4: Neurons comparison.

Neurons	MSE
5	0, 042117
1000	0, 041832
500	0, 041755
750	0, 041765
250	0, 041793

is from 500 neurons. It means that we conclude using the 500 neurons as the optimum schema of the DNN architecture.

In Figure 7, we visualize the network of the optimizer used which is Adamax; the activation function at the input and hidden layer is ReLU, the activation function at the output layer is softmax, the number of neurons is 500 with epoch of 20 iterations. The parameters we use here are based on the smallest value from the previous test.

FIGURE 7: The DNN with the number of neurons which produces the smallest error.

In Table 5, it can be seen that the number of hidden layers and the smallest number of neurons in producing MSE are 5 and 500, respectively.

Figure 8 shows the visualization of the comparison value of the hidden layers number. We test the number of neurons such as 500, 250, and 100 with the hidden layers numbers being 5, 6, 7, and 10. Here are the results of the combination of those tests in Figure 8.

The comparison of the numbers of hidden layers is shown in Figure 8, with the activation function on the input layer and the hidden layer being ReLU, and the activation function at the output layer is softmax.

Table 6 shows that the bigger the number of iterations performed, the smaller the MSE.

Table 6 shows the result of MSE from the epochs number comparison. The numbers of epochs are 100, 500, 1000, and 5000. The numbers of neurons are 250 and 500. From those variants of epochs and neurons numbers, we have the combinations of epochs and neurons.

Figure 9 shows the chart of the value from Table 6. Here is the comparison of the numbers of epochs.

Figure 9 shows visualization of Table 6. The figure is a graph comparing the numbers of epochs. From the variation in the comparison of the numbers of epochs tested, the optimum value is obtained from the number of 5000 epoch iterations on 500 neurons.

Figure 10 shows the visualization of the network we built from the previous experiments.

In Figure 10, visualization of the network with hidden layer architecture is shown. The architecture we built was based on the schema of experiment we did in the previous step. The experiments were conducted to find the optimum optimizer, the optimum activation function, the number of neurons, the number of hidden layers, and the number of epochs. So, from the optimum result, the architecture in Figure 10 is built.

3.2. Final Network of DNN. Of the five scenarios carried out such as selecting the best optimization with the appropriate number of epochs on the deep neural network, the smallest MSE value is obtained, namely,

- (i) The optimizer used is Adamax
- (ii) The activation function used in the input layer and hidden layer is ReLU, and the activation function used in output layer is softmax
- (iii) Number of hidden layers is 5
- (iv) Number of neurons per hidden layer is 500
- (v) Number of epochs is 5000 iterations

3.3. K-Nearest Neighbor Testing. In this section, we present part of the results of technical activities for conducting an indoor positioning system (IPS) on RSSI data using the K-Nearest Neighbor algorithm. The following section is a log of running results from program code in the Python programming language. In the sections below, there are fields for "algorithm," "leaf size," "metric," "metric_params," "n_jobs," "p," and "weight." This parameter is a parameter calculated by the K-Nearest Neighbor algorithm. The following is the log data from running with KNN.

3.4. Nearest Neighbors Regression

```
{'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-
kowsk', 'metric_params': None, 'n_jobs': None,
'n_neighbors': 1, 'p': 2, 'weights': 'uniform'}
```

- (1) uniform 0.9124244832178853 0.000595053327145
6269

TABLE 5: The hidden layers comparison.

No.	Hidden Layer	Neurons	MSE
1	5	500	0, 041627
2	10	500	0, 041667
3	7	500	0, 041636
4	5	250	0, 041635
5	10	250	0, 041691
6	7	250	0, 041655
7	6	250	0, 041645
8	5	100	0, 04166
10	10	100	0, 041684
11	7	100	0, 041675
12	6	100	0, 041669

FIGURE 8: The comparison of the numbers of hidden layers.

TABLE 6: Comparison of the numbers of epochs.

Epochs	Neurons/layer	MSE
100	500	0, 041592911
500	500	0, 041517451
1000	500	0, 041510854
5000	500	0, 041505528
100	250	0, 041591525
500	250	0, 041518033
1000	250	0, 041517792

FIGURE 9: The comparison of the numbers of epochs.

FIGURE 10: The architecture of network with hidden layer.

- (1) `{'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 2, 'p': 2, 'weights': 'uniform'}`
- (2) `uniform 0.9137361277678296 0.0005867071968303365 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 3, 'p': 2, 'weights': 'uniform'}`
- (3) `uniform 0.9150285411581965 0.0005722586575219916 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 4, 'p': 2, 'weights': 'uniform'}`
- (4) `uniform 0.915026812410376 0.0005607869423574921 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 5, 'p': 2, 'weights': 'uniform'}`
- (5) `uniform 0.916984859192242 0.0005527381406783125 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 6, 'p': 2, 'weights': 'uniform'}`
- (6) `uniform 0.9184047118954836 0.0005417671451376087 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 7, 'p': 2, 'weights': 'uniform'}`
- (7) `uniform 0.9194581283766722 0.0005352283295517903 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 8, 'p': 2, 'weights': 'uniform'}`
- (8) `uniform 0.9204137213779005 0.0005321783178257543 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 9, 'p': 2, 'weights': 'uniform'}`
- (9) `uniform 0.9214873701209102 0.0005284177663245653 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 10, 'p': 2, 'weights': 'uniform'}`
- (10) `uniform 0.9220479904435784 0.0005282394804836506 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 11, 'p': 2, 'weights': 'uniform'}`
- (11) `uniform 0.9223855431020228 0.0005288642992810072 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 12, 'p': 2, 'weights': 'uniform'}`
- (12) `uniform 0.9225686183135231 0.0005293692650207558 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 13, 'p': 2, 'weights': 'uniform'}`
- (13) `uniform 0.9231641904953097 0.0005284410250192611 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 14, 'p': 2, 'weights': 'uniform'}`
- (14) `uniform 0.9235985915511365 0.000528734017506188 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 15, 'p': 2, 'weights': 'uniform'}`
- (15) `uniform 0.9237418371504814 0.0005302653518815723 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 16, 'p': 2, 'weights': 'uniform'}`
- (16) `uniform 0.9237729777689404 0.0005329023092141768 {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'min-kowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 17, 'p': 2, 'weights': 'uniform'}`

The next part shows the figures about visualization graphs of R2 and MSE values for 100 nearest neighbors using distance as the measurement. Figure 11 shows the line chart of R2 score of 100 nearest neighbors using distance.

From Figure 11, the difference in the stability of the evaluation value along with the increase in the value of K used in KNN (distance) can be seen.

FIGURE 11: The visualization of R2 score on KNN (distance) with $K = 100$.

Figure 12 shows the MSE value of the previous test of KNN using distance with K value being 100.

Figure 12 shows visualization graphs of MSE values for 100 nearest neighbors (distance). The difference of the stability of the evaluation value along with the increase in the value of K used in KNN can be seen; the curve is the opposite of that of the R2 score.

Figure 13 shows the result of the calculation of the KNN algorithm for RSS using uniform measurement. It shows K results with uniform calculations using $K = 100$.

Figure 13 shows the results of R2 score of the K-Nearest Neighbor algorithm using uniform measurement. R2 score is the proportion of variance in the dependent variable which can be predicted from the independent variable.

Figure 14 shows the result of MSE from the experiment of KNN using uniform measurement with K value being 100.

In Figure 14, we got a curve that is opposite to that in Figure 13. The value of MSE is another method for evaluating RSS position estimation methods. The method produces medium errors which are probably better for small mistakes but sometimes make a big difference. We got the curve where the value of MSE is decreasing, while the K value is increasing.

3.5. Support Vector Machine Testing. In this section, the RSS test results are displayed using the Support Vector Machine (SVM) algorithm.

Table 7 shows the result of testing the SVM algorithm using RSS data. The results obtained show the lowest results compared to the KNN and DNN algorithms.

Table 8 shows the results of SVM testing using normalization values. It is expected that the results obtained are better. However, it turns out that the results obtained are much smaller than those of the testing in Table 7.

3.6. Analysis of Testing Results of KNN, DNN, and SVM on RSS Data. In this section, the MSE of learning approaches, the K-Nearest Neighbor (KNN), Deep Neural Network (DNN),

and Support Vector Machine (SVM) algorithms, is displayed.

Table 9 shows the details of range values in learning approach experiments of KNN (using uniform measurement and distance), DNN, and SVM. We also present some parameters needed in each algorithm such as K value of KNN and the number of neurons with hidden layers used in DNN.

From Table 9, regarding the results of MSE from testing the dataset with KNN that uses calculations using uniform measurement or weight, the highest MSE value is 6.056 with $K = 100$, the lowest MSE value is 5.285 with $K = 10$, and the average value of MSE amounted to 5.725 with $K = 3$ and $K = 44$. Meanwhile, regarding the results of MSE from testing the dataset with KNN that uses calculations using distance, the highest MSE value is 5.950 with $K = 1$, the lowest MSE value is 3.485 with $K = 2$, and the average value of MSE is 4.163 with $K = 49$ and $K = 50$.

Then, the MSE results from the dataset test produced the highest MSE value of 4.169, with the number of hidden layers being 10 and the number of neurons being 250; the lowest MSE value is 4.163 with the number of hidden layers being 7 and the number of neurons being 500; and the average value of MSE is 4.166 with the number of hidden layers being 5 and the number of neurons being 500 for DNN algorithm.

From all three algorithms used for experiment and testing on these RSS data, SVM produced the highest MSE value, 11.06. Figure 15 shows the comparison of average values of the KNN, DNN, and SVM algorithms we used.

In Figure 15, regarding the results of MSE from testing the dataset with KNN that uses calculations using uniform measurements, the average value of MSE is 5.725. Meanwhile, regarding the results of MSE from testing the dataset with KNN which uses calculations using distance, the average value of MSE is 4.163. Then, the average MSE value is 4.166 using the DNN algorithm. The value of MSE produced by SVM is 11.06, which is significantly high.

From these comparisons, we got the optimum MSE value from DNN algorithm. So, we can conclude that the

FIGURE 12: The visualization of MSE on KNN (distance) with $K = 100$. FIGURE 13: The chart of R2 score on KNN (uniform) with $K = 100$. FIGURE 14: The chart of MSE score on KNN (uniform) with $K = 100$.

TABLE 7: Testing result of SVM on RSS.

Parameter	Score
R2 score	0.875786
MSE	11.06557

TABLE 8: The normalized MSE value in iterations.

Iteration	Normalized MSE
1st iteration	0.531005
2nd iteration	0.531005
3rd iteration	0.001919

TABLE 9: The details of range values in learning approach experiments.

Range	KNN (uniform)		KNN (distance)		MSE DNN	MSE SVM
	MSE KNN	K	MSE KNN	K		
MAX	6.056	$K = 100$	5.950	$K = 1$	4.169	11.06
MIN	5.285	$K = 10$	3.485	$K = 2$	4.163	
AVG	5.725	$K = 3 \text{ \& } K = 44$	4.163	$K = 49 \text{ \& } K = 50$	4.166

FIGURE 15: The comparison of average MSE scores in learning approaches for IPS data.

neural network approach is the optimum algorithm to implement the indoor positioning system of our RSS data. The KNN algorithm is slightly higher than DNN but not significantly. But the SVM value of MSE is so much higher than the others. The result of SVM can be caused by the weakness of the SVM in handling the multivariate value in the RSS data. There is no iteration in SVM to repeat the exact value until the result becomes convergent. So, the conclusion is that DNN still is the optimum algorithm used in this multivariate regression of RSS data.

4. Conclusion

From the results of the comparison test with the learning approaches DNN and KNN, it can be concluded that the evaluation with KNN is slightly better than the use of DNN.

However, the results of KNN have low consistency; this is seen from the fluctuations in the movements of the R^2 score and MSE values produced. Meanwhile DNN gives a consistent value even though it has varied number of hidden layers. Testing the dataset with KNN calculations using uniform measurement, highest MSE value is 6.056, with $K = 100$.

Meanwhile, regarding the MSE results from the dataset testing with KNN using calculations using distance, the highest MSE value is 5.950 and the lowest MSE value is 3.485 with average MSE value of 4.163.

Then, the highest MSE value using DNN for testing the dataset is 4.169, the lowest MSE value is 4.163, and the average MSE value is 4.166, with the number of hidden layers being 5 and the number of neurons being 500. These DNN results are so much better than those of KNN algorithm. The results of SVM as a popular algorithm and known to be able to separate hyperplanes very well, in this case, do not show good results at all. The MSE value produced by SVM is so much higher compared to those produced by KNN and DNN.

In the future, the work in this research can be developed for the RSS estimation process using other types of algorithms or using larger data dimensions with different parameters than those used in this research.

Data Availability

The data tested are the signal service data on the device in the Dinamika Bangsa University building. The data can be accessed through the following link: <https://drive.google.com/file/d/1Sq4ffUh8Z9kEO3URVr2XiYgM3qMDYmv/view?usp=drivesdk>.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the research team of Dinamika Bangsa University and Sriwijaya University. The team worked tirelessly all the time and went through many obstacles and problems, but, with good cooperation, this research was completed properly. The authors acknowledge Dinamika Bangsa Foundation for fully funding this research.

References

- [1] V. Bourne, "The rise of indoor positioning," 2016, <http://www.indooratlas.com/wp-content/uploads/2016/09/A-2016-Global-Research-Report-On-The-Indoor-Positioning-Market.pdf>.
- [2] G. D. Bonde, "Finding indoor position of person using wi-fi & smartphone: a survey," *International Journal for Innovative Research in Science and Technology*, vol. 1, no. 8, pp. 202–207, 2015.
- [3] R. F. Brena, J. P. García-Vázquez, C. E. Galván-Tejada, D. Muñoz-Rodríguez, C. Vargas-Rosales, and J. Fangmeyer, "Evolution of indoor positioning technologies: a survey," *Journal of Sensors*, vol. 2017, Article ID 2630413, 21 pages, 2017.
- [4] A. H. Lashkari, B. Parhizkar, and M. N. A. Ngan, "WIFI-based indoor positioning system," in *Proceedings of the 2010 2nd International Conference on Computer and Network Technology*, pp. 76–78, Bangkok, Thailand, April 2010.
- [5] A. Mathisen, S. Krogh Sorensen, A. Stisen, H. Blunck, and K. Gronbaek, "A comparative analysis of indoor WiFi positioning at a large building complex," in *Proceedings of the 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Alcalá de Henares, Spain, October 2016.
- [6] B. J. Shin, K. W. Lee, S. H. Choi, J. Y. Kim, W. J. Lee, and H. S. Kim, "Indoor WiFi positioning system for android-based smartphone," in *Proceedings of the 2010 International Conference on Information and Communication Technology*

- Convergence (ICTC)*, pp. 319–320, Jeju, Republic of Korea, November 2010.
- [7] M. S. Miah and W. Gueaieb, "Mobile robot trajectory tracking using noisy RSS measurements: an RFID approach," *ISA Transactions*, vol. 53, no. 2, pp. 433–443, 2014.
 - [8] F. Seco, "Smartphone-based cooperative indoor localization with RFID technology," *Sensors*, vol. 18, no. 1, p. 266, 2018.
 - [9] J. Zhang, Y. Lyu, J. Patton, S. C. G. Periaswamy, and T. Roppel, "BFVP: a probabilistic UHF RFID tag localization algorithm using Bayesian filter and a variable power RFID model," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 8250–8259, 2018.
 - [10] J. A. del Peral-Rosado, R. Raulefs, J. A. Lopez-Salcedo, and G. Seco-Granados, "Survey of cellular mobile radio localization methods: from 1G to 5G," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1124–1148, 2018.
 - [11] C. Wu, Z. Yang, C. Xiao, and S. Member, "Automatic radio map adaptation for indoor localization using smartphones," *IEEE Transactions on Mobile Computing*, vol. 17, no. 3, pp. 517–528, 2018.
 - [12] B. Benaissa, F. Hendrichovsky, K. Yishida, M. Köppen, and P. Sin, "Phone application for indoor localization based on BLE signal fingerprint," in *Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Paris, France, February 2018.
 - [13] L. Chen, H. Kuusniemi, Y. Chen, L. Pei, T. Kröger, and R. Chen, "Motion restricted information filter for indoor bluetooth positioning," *International Journal of Embedded and Real-Time Communication Systems*, vol. 3, no. 3, pp. 54–66, 2012.
 - [14] Y. Zhuang, J. Yang, Y. Li, L. Qi, and N. El-Sheimy, "Smartphone-based indoor localization with bluetooth low energy beacons," *Sensors*, vol. 16, no. 5, p. 596, 2016.
 - [15] M. H. Abdel Meniem, A. M. Hamad, and E. Shaaban, "GSM-based positioning technique using relative received signal strength," *International Journal of Handheld Computing Research*, vol. 4, no. 4, pp. 38–51, 2013.
 - [16] A. Varshavsky, E. De Lara, J. Hightower, A. Lamarca, and V. Otsason, "GSM indoor localization," *Pervasive and Mobile Computing*, vol. 3, no. 6, pp. 698–720, 2007.
 - [17] A. H. Salamah, M. Tamazin, M. A. Sharkas, and M. Khedr, "An enhanced WiFi indoor localization system based on machine learning," in *Proceedings of the 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 4–7, Alcalá de Henares, Spain, October 2016.
 - [18] R. F. Malik, Mardiah, A. Farissi et al., "WLAN based position estimation system using classification fuzzy K-nearest neighbor (FK-NN)," *IOP Conference Series: Earth and Environmental Science*, vol. 248, no. 1, Article ID 012003, 2019.
 - [19] S. Nurmaini, A. Darmawahyuni, A. N. Sakti Mukti, M. N. Rachmatullah, F. Firdaus, and B. Tutuko, "Deep learning-based stacked denoising and autoencoder for ECG heartbeat classification," *Electronics*, vol. 9, no. 1, p. 135, 2020.
 - [20] C.-H. Lin, J.-L. Chen, and Z.-L. Gaing, "Combining biometric fractal pattern and particle swarm optimization-based classifier for fingerprint recognition," *Mathematical Problems in Engineering*, vol. 2010, Article ID 328676, 14 pages, 2010.
 - [21] Y. Lukito and A. R. Chrismanto, "Recurrent neural networks model for WiFi-based indoor positioning system," in *Proceedings of the 2017 International Conference on Smart Cities, Automation & Intelligent Computing Systems (ICON-SONICS)*, pp. 121–125, Yogyakarta, Indonesia, November 2017.
 - [22] C. Song, J. Wang, and G. Yuan, "Hidden naive Bayes indoor fingerprinting localization based on best-discriminating AP selection," *ISPRS International Journal of Geo-Information*, vol. 5, no. 10, p. 189, 2016.
 - [23] Z. Wu, Q. Xu, J. Li, C. Fu, Q. Xuan, and Y. Xiang, "Passive indoor localization based on CSI and naive Bayes classification," *IEEE Transactions on Systems, Man, and Cybernetics Systems*, vol. 48, no. 9, pp. 1–12, 2017.
 - [24] H. I. Alsaadi, L. K. Almajmaie, and W. A. Mahmood, "Forecasting of Twitter hashtag temporal dynamics using locally weighted projection regression," in *Proceedings of the 2017 International Conference on Engineering and Technology (ICET)*, Antalya, Turkey, August 2017.
 - [25] P. Mirowski, P. Whiting, H. Steck et al., "Probability kernel regression for WiFi localisation," *Journal of Location Based Services*, vol. 6, no. 2, pp. 1–20, 2012.
 - [26] H. Zou, M. Jin, H. Jiang, L. Xie, and C. J. Spanos, "WinIPS: WiFi-based non-intrusive indoor positioning system with online radio map construction and adaptation," *IEEE Transactions on Wireless Communications*, vol. 16, no. 12, pp. 8118–8130, 2017.
 - [27] A. Adege, H.-P. Lin, G. Tarekegn, and S.-S. Jeng, "Applying deep neural network (DNN) for robust indoor localization in multi-building environment," *Applied Sciences*, vol. 8, no. 7, p. 1062, 2018.
 - [28] W. Zhang, K. Liu, W. Zhang, Y. Zhang, and J. Gu, "Deep neural networks for wireless localization in indoor and outdoor environments," *Neurocomputing*, vol. 194, pp. 279–287, 2016.
 - [29] M. Zhang, J. Li, Y. Li, and R. Xu, "Deep learning for short-term voltage stability assessment of power systems," *IEEE Access*, vol. 9, pp. 29711–29718, 2021.
 - [30] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagc, "Deep learning techniques in big data analytics," *Big Data Technologies and Applications*, Selerity, New York, NY, USA, pp. 133–156, 2016.