Development of Adaptive Fuzzy-Neuro Generalized Learning-Vector Quantization Using PI Membership Function (AFNGLVQ-PI)

WISNU JATMIKO\(^{1}\), (Senior Member, IEEE), ANDRY SUNANDAR\(^{1}\), M. SAKTI ALVISSALIM\(^{1}\), M. IQBAL TAWAKAL\(^{1}\), M. FEBRIAN RACHMADI\(^{1}\), M. ANWAR MA'SUM\(^{1}\), (Member, IEEE), HANIF A. WISESA\(^{1}\), AND TOSHIO FUKUDA\(^{2}\), (Life Fellow, IEEE)

\(^{1}\)Faculty of Computer Science, Universitas Indonesia, Depok 16424, Indonesia
\(^{2}\)Meijo University, Aichi 468-8502, Japan

Corresponding authors: Wisnu Jatmiko (wisnuj@cs.ui.ac.id), Andry Sunandar (andry.sunandar@ui.ac.id), M. Sakti Alvissalim (m.sakti@ui.ac.id), M. Anwar Ma’sum (muhhammad_anwar@cs.ui.ac.id), and Hanif A. Wisesa (hanif.wisesa@ui.ac.id)

This work was supported by the World Class Professor (WCP) under Grant 123.7/D2.3/KP/2018 through the Ministry of Research, Technology and Higher Education of the Republic of Indonesia, 2018.

Abstract

In a real-world environment, there are several difficult obstacles to overcome in classification. Those obstacles are data overlapping and skewness of data distribution. Overlapping data occur when many data from different classes overlap with each other; this condition often occurs when there are many classes in a data set. On the other hand, skewness of data distribution occurs when the data distribution is not a Gaussian (normal) distribution. To overcome these two problems, a new method called Adaptive Fuzzy-Neuro Generalized Learning Vector Quantization using PI membership function (AFNGLVQ-PI) is proposed in this study. AFNGLVQ-PI is derived from Fuzzy-Neuro Generalized Learning Vector Quantization using the PI membership function (FNGLVQ-PI). In FNGLVQ-PI, the updated values for minimum and maximum variables in the fuzzy membership function are set based on the mean of the updated values. Whereas, in the newly proposed AFNGLVQ-PI, updated values for minimum, maximum, and mean variables are derived based on the differential equations to approximate the data distribution better. In this study, the newly proposed AFNGLVQ-PI algorithm was tested and verified on twelve different data sets. Two of the data sets are synthetic data sets where we could compare the performance of the data sets in different overlapping conditions and levels of skewness. The rest of the data sets were chosen and used as a benchmark to compare the performance of the proposed algorithm. In the experiment, AFNGLVQ-PI took first place in 18 out of 29 experiments. Furthermore, AFNGLVQ-PI also achieved positive improvements for all data sets used in the experiments, which could not be achieved by the Learning Vector Quantization (LVQ), Generalized Learning Vector Quantization (GLVQ), and other commonly used algorithms, such as SVM, kNN, and MLP.

Index Terms

AFNGLVQ-PI, classification, FNGLVQ, GLVQ, Fuzzy.

I. INTRODUCTION

In a real-world problem where many classes are available in a data set, it often occurs that several classes are overlapping with each other. Even when many features are available as discriminators, a data set that contains many classes is hard to deal with, especially for a classification task. Many approaches have been developed to solve this problem, and one of them is data fuzzification. Data that have gone through fuzzification tend to be easier to analyze because they have approximate class boundaries [1]. Moreover, the fuzzification procedure helps the classifier to resolve any uncertainty in the data set [2].

Many studies have used fuzzy methods widely in classification problems, especially with overlapping data. In [3], a new method for multi-criteria optimization classification using fuzzification, kernel, and penalty factors to predict protein-interaction hot spots was introduced. It was observed that fuzzification can help the new method to eliminate noise, outliers, anomalies, and uncertainty from data. In another
study, [4] successfully developed a self-learning fuzzy spiking neural network based on fuzzy data-clustering that is capable to tackle overlapping clusters of irregular data. Similarly, [7] has also tried to modify the neuro-fuzzy model to incorporate an evolution capability. Many previous studies, such as [5] and [6], have also used the same approach to solve overlapping data problems. For further information, a review article [8] has explained thoroughly many kinds of applications where neuro-fuzzy systems have been implemented for real-life problems.

Despite its usefulness, data fuzzification is sometimes not enough to improve a classifier’s performance because its performance depends on the shape of the fuzzification. The shape of fuzzification is based on a specific membership function, and many approaches have been used to transform data distributions into fuzzy membership functions [9], [10]–[11]. These studies proposed their unique membership functions based on the minimum, mean, and maximum values of the clustered data. In this study, an adaptive fuzzification classifier based on PI membership is developed to simplify the fuzzification process while having the ability to adapt to different data distributions.

Several previous studies have proposed different adaptive approaches to improve their fuzzy classifiers. In one study, a combination of Adaptive Neural Network and Nonlinear Model Predictive Control was proposed for multi-rate networked industrial process control where it was tested in a continuous stirred tank reactor system for its effectiveness [12]. Another study investigated the effectiveness of the Adaptive Neural Network for remotely operated underwater (ROV) vehicle’s object tracking [13]. Another study has also shown the effectiveness of the adaptive method when used together with particle swarm optimization (PSO) to optimize Self-organizing Radial Basis Function [14]. Another study developed an event-triggered fuzzy bipartite tracking control for network systems based on distributed reduced-order observers [15]. Several studies have also developed various fuzzy systems for control both in continuous-time and discrete-time systems [16], [17]. Furthermore, there are also recent studies that investigated a fuzzy system in chaotic nonlinear system [18] and improved the performance of fuzzy inertial neural networks with Markov jumping parameters [19].

The method developed in this study is built upon a method named learning vector quantization (LVQ), which is a family of single-layer neural networks [20]. LVQ itself is a supervised version of the self-organizing map (SOM) [21], which provides a simple and intuitive way to look at data distributions. LVQ uses the paradigm of competition-based learning in which every neuron, codebook, or reference vector competes to attain the closest match to the input vector, and the winner gets updated. Despite its simplicity, LVQ has been successfully used in various fields and yielded good results in many applications such as speech recognition [22], character and handwriting recognition [23], speaker identification [24], vehicle detection [25], and breast cancer detection [26]. In one of our previous studies, we have successfully implemented a tree-like neural-network to solve several problems in big data [27].

Many attempts have been made to improve the capabilities of the original LVQ. In the original study, LVQ itself has three different versions (i.e., LVQ1, LVQ2, and LVQ3) which are based on different update rules [20]. An improvement of the LVQ2 algorithm, i.e., Generalized Learning Vector Quantization (GLVQ), was proposed to ensure that the algorithm would reach convergence by minimizing the cost function [28]. Another study proposed a new method called Generalized Relevance Learning Vector Quantization (GRLVQ), a method that gives a weight to each feature based on its relevance, which can act as a feature-reduction method as well [29].

Another different approach to improve LVQ’s performance is to incorporate fuzzy theory into the algorithm to determine the winner. Previous studies have shown that fuzzy theory can be used for enhancing various algorithms. One previous study successfully developed Fuzzy SOM to mine DNA motifs [30]. Another study developed Recurrent Self-Evolving Fuzzy Neural Networks for predicting driving fatigue by using brain EEG as predictors [31]. The similarity of these studies is, instead of using Euclidean distance to measure the distance between the input vector and each reference vector, they use fuzzy similarity. This idea was first proposed by Jatmiko et al. who reported that Fuzzy-Neuro LVQ (FNLVQ) could detect outliers in the testing phase [32]. FNLVQ was also applied for heartbeat classification to determine the presence of heart diseases [33]. In another our previous study, FNLVQ was enhanced by PSO to gain better performance in discriminating odors [34], which is an extension of a previous work [35]. Similar works in optimizing fuzzy classifier were done using bee colony and genetic algorithm [36], [37]. Some other studies modified LVQ by using different approaches. For example, LVQ for online semi-supervised learning [38], LVQ using DropConnect for classification stability [39], and modification of LVQ with feed-forwards neural networks [40].

The main contribution of this paper is to present a fast and effective learning architecture based on fuzzification by approximating the distribution measurements of each feature using a membership function for each class. The proposed method is named Adaptive Fuzzy-Neuro Generalized Learning Vector Quantization using PI membership function (AFNGLVQ-PI). The main contribution of this paper is derived into two important points. First, we developed the adaptive version of FNGLVQ namely AFNGLVQ where the method adaptively updates the three values of the codebook representation (min, mean, max) instead of updates only the mean value. Second, we developed AFNGLVQ with PI membership function along with the AFNGLVQ with fuzzy triangle membership function as stated in the first point. This study shows that the proposed measures used in AFNGLVQ-PI solve a difficult case where there are highly overlapping feature values between classes. The same architecture could also
be used to learn the characteristic of a certain class in terms of its feature values. The proposed method also improved the previous algorithm called Fuzzy-Neuro Generalized Learning Vector Quantization (FNGLVQ) by taking advantage of the adaptive nature of the PI membership function. The PI membership function itself is useful to accurately solve a difficult case where the feature value has a degree of skewness. Note that we have previously proposed and conducted an early experiment on multi-codebook AFNGLVQ [41]. However, its performance gain was not substantial, and therefore, we chose the original version of AFNGLVQ (i.e., the non-multi-codebook) to be further developed in this study.

This paper discusses several comparisons between the proposed algorithm and other commonly used algorithms, which are: Naive Bayes [42], kNN [43], Multilayer Perceptron [44], Support Vector Machine [45], and J48 (i.e., a Java implementation of C4.5) [46]. The algorithms were implemented in Java Programming using WEKA Data Mining Tool [47].

The organization of the rest of this paper is as follows. Section II describes previous iterations of LVQ algorithms used to derive the newly proposed algorithm AFNGLVQ-PI, including the update rules and membership functions. Section III explains the newly proposed algorithm AFNGLVQ-PI. Section IV discusses data sets that were used to test the new algorithm and results of experiments that have been done. The section also provides more extensive discussions of the results of the experiment. Finally, the conclusions of this study are explained in Section VI.

II. PREVIOUS ITERATIONS OF LEARNING VECTOR QUANTIZATION ALGORITHM

A. LEARNING VECTOR QUANTIZATION

The unsupervised method of SOM proposed by Kohonen provides an intuitive way to gain insight into the intrinsic behavior of data [21]. By assigning a class to each codebook and adjusting the learning and update procedures, this algorithm can be turned into a supervised method and used for classification. This supervised method is called LVQ. The main idea is to pull the winning codebook closer to the input vector if they share the same class and push the winning codebook away from the input vector if they do not belong to the same class.

For every LVQ iteration using an input vector \( x \in X \), where \( X \) is the set of all input vectors, and \( w_i \in W \), where \( W \) is the set of codebooks with a previously assigned class denoted by \( i \), the closest codebook vector \( w_c \) can be defined by (1):

\[
w_c = \min(\|x - w_i\|).
\]

The winning codebook is updated according to the following rules:

- IF \( x \) AND the closest codebook vector belongs to the same class THEN

  \[
  w_c(t + 1) = w_c(t) + \alpha(t)[x(t) - w_c(t)].
  \]

- IF \( x \) AND the closest codebook vector belongs to a different class

  \[
  w_c(t + 1) = w_c(t) - \alpha(t)[x(t) - w_c(t)].
  \]

where \( 0 < \alpha(t) < 1 \) and \( \alpha \) decrease monotonically with time.

Figure 1 shows the LVQ architecture.

![FIGURE 1. Illustration of LVQ's architecture.](image-url)

B. GENERALIZED LEARNING VECTOR QUANTIZATION

GLVQ uses a different learning method from the LVQ algorithm to ensure that the reference vectors approximate the class distribution. This learning method is based on minimization of a cost function. Let \( x \) be an input vector belonging to class \( C \), \( w_1 \) be the nearest reference vector belonging to the same class as \( x \) (i.e., \( C_1 = C \)), and \( w_2 \) be the nearest reference vector that belongs to a different class from \( x \) (i.e., \( C_2 \neq C \)). A relative distance difference of \( \mu(x) \) is defined by (4):

\[
\mu(x) = \frac{d_1 - d_2}{d_1 + d_2}
\]

where \( d_1 \) and \( d_2 \) are the distances of input vector \( x \) from reference vector \( w_1 \) and reference vector \( w_2 \) respectively. The relative distance difference of \( \mu(x) \) ranges between \(-1\) and \(+1\). The value of \( \mu(x) \) will be negative if input vector \( x \) is classified correctly, whereas the value of \( \mu(x) \) will be positive if input vector \( x \) is classified incorrectly. To reduce error rates, \( \mu(x) \) should decrease for all input vectors. Therefore, a learning criterion can be formulated by minimizing the cost function \( S \), defined as in (5):

\[
S = \sum_{i=1}^{N} f(\mu(x_i))
\]

where \( N \) is the number of training input vectors and \( f(\mu) \) is a monotonically increasing function. To minimize the cost function \( S \), the values of \( w_1 \) and \( w_2 \) should be updated based on the steepest-descent method with a small positive constant \( \alpha \), as in (6):

\[
w_i \leftarrow w_i - \alpha \frac{\partial S}{\partial w_i}, i = 1, 2.
\]
and (8).
\[
\frac{\partial S}{\partial w_1} = \frac{\partial S}{\partial \mu} \frac{\partial \mu}{\partial d_1} \frac{\partial d_1}{\partial w_1} = - \frac{\partial f}{\partial \mu} \frac{4d_2}{\mu (d_1 + d_2)^2} (x - w_1) \\
\frac{\partial S}{\partial w_2} = \frac{\partial S}{\partial \mu} \frac{\partial \mu}{\partial d_2} \frac{\partial d_2}{\partial w_2} = - \frac{\partial f}{\partial \mu} \frac{4d_1}{\mu (d_1 + d_2)^2} (x - w_2)
\]

Therefore, the GLVQ learning rules can be described as in (9) and (10).
\[
w_1 \leftarrow w_1 + \alpha \frac{\partial f}{\partial \mu} \frac{d_2}{\mu (d_1 + d_2)^2} (x - w_1) \\
w_2 \leftarrow w_2 + \alpha \frac{\partial f}{\partial \mu} \frac{d_1}{\mu (d_1 + d_2)^2} (x - w_2)
\]

C. FUZZY-NEURO LEARNING VECTOR QUANTIZATION

One way to improve the original LVQ algorithm is to change the representation of the reference vectors. One such algorithm is Fuzzy Algorithms for Learning Vector Quantization (FALVQ), which was proposed by Karayiannis et al. [48]. FALVQ is derived by minimizing the weighted sum of squared Euclidean distances between an input vector representing a feature vector and the codebook weight vectors from the LVQ network. Another example is FNLVQ, proposed by Jatmiko et al. [32]. FNLVQ is an LVQ algorithm that combines fuzzy theory and LVQ. Fuzzy functions are used to represent both the codebooks and the input vector.

Conceptually, the reason why fuzzy theory is added to LVQ in FNLVQ was to deal with error or uncertainty in the odor data set. By using fuzzy theory, errors within the data can be defined as fuzziness. The fuzziness of the data is then represented as a fuzzy membership function with values ranging from 0 to 1. The membership function in FNLVQ can be defined with various kinds of shapes, but a triangle function is the simplest form of the membership function. A triangle membership function can be defined as (11):
\[
w = (\text{min}, \text{mean}, \text{max})
\]
where \text{min} is the minimum value, \text{mean} is the average value, and \text{max} is the maximum value of the membership function yielded by a data cluster. This membership function is defined for all features in all codebooks. Because the data representation has been changed, the distance measurement in LVQ is also replaced by a similarity measurement between the fuzzified input vectors’ triangle membership function and the codebooks’ triangle membership function. The similarity measurement is obtained by calculating the intersection between the two triangles. Figure 2 illustrates this similarity measurement. The blue triangle depicts the codebook’s membership functions, The orange triangle depicts input membership functions, and \(\mu\) is the similarity value. Figure 3 depicts the overall architecture of the FNLVQ algorithm.

FNLVQ includes an update process that can be divided into three cases. In the first case, the codebook with maximum similarities belongs to the same class as the input vector. This codebook will then be translated to become closer with the input vector, hence produce higher similarity. An illustration of this translation process is shown in Figure 4. The translation is performed based on (12)-(15).
\[
w_i(t + 1) = w_i(t) + \alpha(t)((1 - \mu)(x(t) - w_i(t)))
\]
\[
w_{\text{mean}}(t + 1) = w_{\text{mean}}(t) - (1 + \beta)(w_{\text{mean}}(t) - w_{\text{min}}(t))
\]
\[
w_{\text{min}}(t + 1) = w_{\text{min}}(t) + (1 + \beta)(w_{\text{max}}(t) - w_{\text{mean}}(t))
\]
\[
w_{\text{max}}(t + 1) = w_{\text{max}}(t) - (1 + \beta)(w_{\text{max}}(t) - w_{\text{mean}}(t))
\]

In the second case, The codebook with the highest similarity does not belong to the same class as the input vector. Therefore, the triangle membership function is translated away from the input vector. An illustration of this translation process is depicted in Figure 5. The translation is performed based on (16)-(19).
\[
w_i(t + 1) = w_i(t) - \alpha(t)((1 - \mu)(x(t) - w_i(t)))
\]
\[
w_{\text{mean}}(t + 1) = w_{\text{mean}}(t) + 1 + \beta( w_{\text{mean}}(t) - w_{\text{min}}(t))
\]
\[
w_{\text{min}}(t + 1) = w_{\text{min}}(t) + (1 + \beta)(w_{\text{max}}(t) - w_{\text{min}}(t))
\]
\[
w_{\text{max}}(t + 1) = w_{\text{max}}(t) - (1 + \beta)(w_{\text{max}}(t) - w_{\text{mean}}(t))
\]
In the third case, there is no winner codebook because all the similarity values are zero. In this case, all the codebooks’ triangle membership functions are widened by a constant value. Equation 20 defines this process where $\gamma$ stands as the constant value. This process is illustrated in Figure 6.

$$w_i(t + 1) = w_i(t) \times \gamma$$

(D. FUZZY-NEURO GENERALIZED LEARNING VECTOR QUANTIZATION)

Fuzzy-Neuro Generalized Learning Vector Quantization (FNGLVQ) is a modification of the GLVQ algorithm which is combined with the fuzzification mechanism of fuzzy-neuro LVQ as proposed by Jatmiko et al. [32]. The original reason for this development was to increase the accuracy of arrhythmic heartbeat classification. In this study, the fuzzification in FNGLVQ has been claimed to perform better for overlapping class data. FNGLVQ has also overcome the weakness of FLVQ that it depends on the original vector initialization.

The FNGLVQ learning process begins by calculating similarity values ($\mu_{ij}$), where $j$ is a class label and $i$ is index of feature. These values are calculated by defining the value of the membership degree of each feature ($x_i$) for each membership function ($h_j(x)$), as shown in Equation (21):

$$\mu_{ij} = h_j(x_i)$$

Afterwards, all the similarity values are summed for each feature, as shown in (22):

$$\mu_j = \frac{1}{k} \sum_{i=1}^{k} \mu_{ij}$$

where $k$ is the number of feature. The winner will be determined by finding the maximum similarity value using (23):

$$w_p = \max(\mu_j)$$

To update the value of the winning reference vector, the misclassification error (MCE) must be calculated. In FNGLVQ, which uses a fuzzy similarity approach, the MCE can be calculated by transforming the similarity values into distances $d = 1 - \mu$. Then these distances are substituted into Equation (24):

$$\phi(x) = \frac{d_1 - d_2}{d_1 + d_2} = \frac{(1 - \mu_1) - (1 - \mu_2)}{(1 - \mu_1) + (1 - \mu_2)} = \frac{\mu_2 - \mu_1}{2 - \mu_1 - \mu_2}$$

where variable $\mu_1$ is the similarity value between input vector $x$ and the nearest reference vector that belongs to the same class as $x$ ($C_x = C$). Variable $\mu_2$ is the similarity value between input vector $x$ and the nearest reference vector that belongs to a different class from $x$ ($C_x \neq C$).

The original formula in GLVQ must be derived to minimize the cost function $S$, but this derivation is now concerning to the mean value of the reference vector, as in (25). The derivations are shown in (26) and (27).

$$\frac{\partial S}{\partial \mu_1} = \frac{\partial S}{\partial \phi} \frac{\partial \phi}{\partial \mu_1}; \quad i = 1, 2$$

$$\frac{\partial \phi}{\partial \mu_1} = \frac{\partial \phi}{\partial \mu_2} \frac{\partial \mu_1}{\partial \mu_2}; \quad i = 1, 2$$

The derivation of $\frac{\partial S}{\partial \mu_i}$ depends on the membership function used in the reference vector. In FNGLVQ, the triangle shape is used as its membership function due to its simplicity.
Only three parameters, \( \min \), \( \text{mean} \), and \( \max \) values, are needed. Therefore, the reference vector can be represented as in (28), and the triangle membership function is defined in (29). An illustration of the triangle membership function can be seen in Figure 7.

\[
w_{ij} = (w_{\min,ij}, w_{\text{mean},ij}, w_{\max,ij})
\]

\[
\mu = h(x) = \begin{cases} 
\frac{x - w_{\min}}{w_{\text{mean}} - w_{\min}}, & w_{\min} < x \leq w_{\text{mean}} \\
\frac{w_{\text{mean}} - x}{w_{\max} - w_{\text{mean}}}, & w_{\text{mean}} < x \leq w_{\max} \\
0, & \text{others}
\end{cases}
\]

(28) (29)

![FIGURE 7. An illustration of triangle membership function.](image)

The learning rule can be described as follows.

- **IF** \( w_{\min} < x < w_{\text{mean}} \) **THEN**
  \[
w_1(t + 1) \leftarrow w_1(t) - \alpha \frac{\partial f}{\partial \mu} \frac{2(1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2} \frac{x - w_{\min}}{(w_{\text{mean}} - w_{\min})^2}
\]

  (30)

- **IF** \( w_{\text{mean}} < x < w_{\max} \) **THEN**
  \[
w_1(t + 1) \leftarrow w_1(t) + \alpha \frac{\partial f}{\partial \mu} \frac{2(1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2} \frac{x - w_{\min}}{(w_{\text{mean}} - w_{\min})^2}
\]

  (31)

- **IF** \( x < w_{\min} \) \text{AND} \( x > w_{\max} \) **THEN**
  \[
w_1(t + 1) \leftarrow w_1(t); \quad i = 1, 2
\]

(34)

where every update is performed on \( w_{\text{mean}} \). The updates for \( w_{\min} \) and \( w_{\max} \) are based on the difference between the old \( w_{\text{mean}} \) and the new \( w_{\text{mean}} \)

\[
w_{\min} \leftarrow w_{\text{mean}}(t + 1) - (w_{\text{mean}}(t) - w_{\min}(t))
\]

(35)

\[
w_{\max} \leftarrow w_{\text{mean}}(t + 1) + (w_{\max}(t) - w_{\text{mean}}(t))
\]

(36)

To accommodate a better approximation of the class distribution, FNGLVQ can use the PI membership function instead of a triangle membership function. This algorithm is called FNGLVQ-PI. Equation (37) is the definition of the PI membership function in the FNGLVQ-PI algorithm, and Equations (38) - (46) give the learning rules of FNGLVQ with the PI membership function. PI membership function itself is illustrated in Figure 8.

\[
f = \begin{cases} 
2\left(\frac{x - w_{\min}}{w_{\text{mean}} - w_{\min}}\right)^2, & w_{\min} < x \leq w_{\text{mid}L} \\
1 - 2\left(\frac{w_{\text{mean}} - x}{w_{\text{mean}} - w_{\min}}\right)^2, & w_{\text{mid}L} < x \leq w_{\text{mean}} \\
1 - 2\left(\frac{w_{\max} - x}{w_{\max} - w_{\text{mean}}\right)^2, & w_{\text{mean}} < x \leq w_{\text{mid}R} \\
2\left(\frac{w_{\max} - x}{w_{\max} - w_{\text{mean}}\right)^2, & w_{\text{mid}R} < x \leq w_{\max} \\
0, & \text{others}
\end{cases}
\]

(37)

![FIGURE 8. An illustration of PI membership function which is composed from two sigmoid functions.](image)

Learning rules of FNGLVQ-PI:

- **IF** \( w_{\min} < x \leq w_{\text{mid}L} \) **THEN**
  \[
w_1(t + 1) \leftarrow w_1(t) - \alpha \frac{\partial f}{\partial \mu} \frac{2(1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2} (w_{\text{mean}} - w_{\min})^3
\]

  (38)

- **IF** \( w_{\text{mid}L} < x \leq w_{\text{mean}} \) **THEN**
  \[
w_1(t + 1) \leftarrow w_1(t) + \alpha \frac{\partial f}{\partial \mu} \frac{2(1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2} (w_{\text{mean}} - w_{\min})^3
\]

  (39)

- **IF** \( w_{\text{mid}L} < x \leq w_{\text{mean}} \) **THEN**
  \[
w_1(t + 1) \leftarrow w_1(t) + \alpha \frac{\partial f}{\partial \mu} \frac{2(1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2} (w_{\text{mean}} - w_{\min})^3
\]

  (40)

- **IF** \( w_{\text{mid}L} < x \leq w_{\text{mean}} \) **THEN**
  \[
w_1(t + 1) \leftarrow w_1(t) + \alpha \frac{\partial f}{\partial \mu} \frac{2(1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2} (w_{\text{mean}} - w_{\min})^3
\]

  (41)
A. TRIANGLE MEMBERSHIP FUNCTION

Figures 10 and 11 illustrate the proposed adaptive process using the triangle membership function. The corresponding update rules for triangle membership of AFNGLVQ are described in (47) - (58).

- IF $w_{\text{mean}} < x \leq w_{\text{midR}}$ THEN
  \[ w_1(t+1) \leftarrow w_1(t) + \alpha \frac{\partial f}{\partial \phi} \frac{2(1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2} \frac{(w_{\text{mean}} - x)(x - w_{\text{max}})}{(w_{\text{max}} - w_{\text{mean}})^3} \]  
  \[ w_2(t+1) \leftarrow w_2(t) - \alpha \frac{\partial f}{\partial \phi} \frac{2(1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2} \frac{(w_{\text{mean}} - x)(x - w_{\text{max}})}{(w_{\text{max}} - w_{\text{mean}})^3} \]  
- IF $\frac{w_{\text{max}}}{2} < x \leq w_{\text{max}}$ THEN
  \[ w_1(t+1) \leftarrow w_1(t) + \alpha \frac{\partial f}{\partial \phi} \frac{2(1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2} \frac{(w_{\text{mean}} - x)^2}{(w_{\text{max}} - w_{\text{mean}})^3} \]  
  \[ w_2(t+1) \leftarrow w_2(t) - \alpha \frac{\partial f}{\partial \phi} \frac{2(1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2} \frac{(w_{\text{mean}} - x)^2}{(w_{\text{max}} - w_{\text{mean}})^3} \]  
- IF $x \leq w_{\text{min}}$ OR $x \geq w_{\text{max}}$ THEN
  \[ w_i(t+1) \leftarrow w_i(t), \quad i = 1, 2 \]  

III. ADAPTIVE FUZZY-NEURO GENERALIZED LEARNING VECTOR QUANTIZATION

Adaptive FNGLVQ (AFNGLVQ) derives the updated values based not only on the mean value of the fuzzy membership function but also on its min and max parameters. This intuition comes from an examination of the distribution of certain data set features. In these cases, the data distribution is not symmetrical but skewed to the left and right. In the original FNGLVQ, the min and max values are updated by using the same value (i.e., the mean value), so the shape of the membership function stays the same throughout the learning process. In other words, the learning process fails to capture the real data distribution. To remedy this, an adaptive measure where the min, mean, and max values are updated based on the input value is proposed. In this study, two update rules for both triangle and PI membership functions are proposed. This development is closely related to the FNGLVQ algorithm where it can use both membership functions as well. Figure 9 shows an example of feature distribution using both membership functions.

FIGURE 9. An example of feature distribution of one data set overlaid by the corresponding triangle (top) and PI (bottom) membership functions.
the mean values will be updated as in (55) and (56), and

\[
\begin{align*}
\text{w}_{1\text{mean}}(t+1) & \leftarrow \text{w}_{1\text{mean}}(t) \\
-\alpha f & \frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2} (w_{\text{max}} - x) \\
\text{w}_{2\text{mean}}(t+1) & \leftarrow \text{w}_{2\text{mean}}(t) \\
+\alpha f & \frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2} (w_{\text{max}} - x) \\
\end{align*}
\]

the max values will be updated as in (57) and (58),

\[
\begin{align*}
\text{w}_{1\text{max}}(t+1) & \leftarrow \text{w}_{1\text{max}}(t) \\
-\alpha f & \frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2} (x - w_{\text{mean}}) \\
\text{w}_{2\text{max}}(t+1) & \leftarrow \text{w}_{2\text{max}}(t) \\
+\alpha f & \frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2} (x - w_{\text{mean}}) \\
\end{align*}
\]

Despite the simplicity of using triangle shape as a membership function for the FNGLVQ algorithm and its subsequent variant, there is a severe limitation of what data can be fit reasonably well. The standard triangle shape determines that the membership function value of a data is divided linearly between the mean and min or max value of the function. However, certain data might not fit this case. For example, if the number of data with a value near the mean is significantly higher than the data near min, then it is natural to assign the data near the mean with a notably higher membership function value. Therefore, another membership function is needed to accommodate this kind of data. Figure 12 shows examples of data that can be fitted and cannot be fitted well by using the triangle membership function.

**FIGURE 12.** An illustration of the triangle membership function's drawback.

**B. PI MEMBERSHIP FUNCTION**

As previously mentioned in Section II-D, the PI membership function is used to accommodate better approximation of data. The PI membership function can be modified to behave in an adaptive manner for AFNGLVQ. Figures 13 and 14 illustrate this concept, and the corresponding update rules are described in (59) - (82).

- **IF** \( w_{\text{min}} < x \leq w_{\text{mid}} \), **THEN**
  
  The min values will be updated as in (59) and (60),

\[
\begin{align*}
\text{w}_{1\text{min}}(t+1) & \leftarrow \text{w}_{1\text{min}}(t) \\
-\alpha f & \frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2} (w_{\text{max}} - x) \\
\text{w}_{2\text{min}}(t+1) & \leftarrow \text{w}_{2\text{min}}(t) \\
+\alpha f & \frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2} (w_{\text{max}} - x) \\
\end{align*}
\]

**FIGURE 13.** An illustration of AFNGLVQ’s PI membership function and its update rules when the \( x \) value fall between \( w_{\text{min}} \) and \( w_{\text{mean}} \).

**FIGURE 14.** An illustration of AFNGLVQ’s PI membership function and its update rules when the \( x \) value fall between \( w_{\text{mean}} \) and \( w_{\text{max}} \).

\[
\begin{align*}
\text{w}_{1\text{mean}}(t+1) & \leftarrow \text{w}_{1\text{mean}}(t) \\
-\alpha f & \frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2} (x - w_{\text{mean}}) \\
\text{w}_{2\text{mean}}(t+1) & \leftarrow \text{w}_{2\text{mean}}(t) \\
+\alpha f & \frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2} (x - w_{\text{mean}}) \\
\end{align*}
\]

the mean values will be updated as in (61) and (62), and

\[
\begin{align*}
\text{w}_{1\text{max}}(t+1) & \leftarrow \text{w}_{1\text{max}}(t) \\
-\alpha f & \frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2} (x - w_{\text{mean}}) \\
\text{w}_{2\text{max}}(t+1) & \leftarrow \text{w}_{2\text{max}}(t) \\
+\alpha f & \frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2} (x - w_{\text{mean}}) \\
\end{align*}
\]

the max values will be updated as in (63) and (64).

\[
\begin{align*}
\text{w}_{1\text{max}}(t+1) & \leftarrow \text{w}_{1\text{max}}(t) \\
\text{w}_{2\text{max}}(t+1) & \leftarrow \text{w}_{2\text{max}}(t) \\
\end{align*}
\]
IF $w_{midL} < x \leq w_{mean}$ THEN

The min values will be updated as in (65) and (66),

$$w_{1\min}(t + 1) \leftarrow w_{1\min}(t)$$

$$\frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2}$$

$$4 \times (w_{\text{mean}} - x)^2$$

$$w_{2\min}(t + 1) \leftarrow w_{2\min}(t)$$

$$\frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2}$$

$$4 \times (w_{\text{mean}} - x)^2$$

(65)

(66)

the mean values will be updated as in (67) and (68), and

$$w_{1\text{mean}}(t + 1) \leftarrow w_{1\text{mean}}(t)$$

$$\frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2}$$

$$4 \times (w_{\text{mean}} - x)(x - w_{\text{min}})$$

$$w_{2\text{mean}}(t + 1) \leftarrow w_{2\text{mean}}(t)$$

$$\frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2}$$

$$4 \times (w_{\text{mean}} - x)(x - w_{\text{min}})$$

(67)

(68)

the max values will be updated as in (69) and (70).

$$w_{1\text{max}}(t + 1) \leftarrow w_{1\text{max}}(t)$$

$$w_{2\text{max}}(t + 1) \leftarrow w_{2\text{max}}(t)$$

(69)

(70)

IF $w_{\text{mean}} < x \leq w_{midR}$ THEN

The min values will be updated as in (71) and (72),

$$w_{1\min}(t + 1) \leftarrow w_{1\min}(t)$$

$$w_{2\min}(t + 1) \leftarrow w_{2\min}(t)$$

(71)

(72)

the mean values will be updated as in (73) and (74), and

$$w_{1\text{mean}}(t + 1) \leftarrow w_{1\text{mean}}(t)$$

$$\frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2}$$

$$4 \times (w_{\text{mean}} - x)(x - w_{\text{max}})$$

$$w_{2\text{mean}}(t + 1) \leftarrow w_{2\text{mean}}(t)$$

$$\frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2}$$

$$4 \times (w_{\text{mean}} - x)(x - w_{\text{max}})$$

(73)

(74)

the max values will be updated as in (75) and (76).

$$w_{1\text{max}}(t + 1) \leftarrow w_{1\text{max}}(t)$$

$$\frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2}$$

$$4 \times (w_{\text{mean}} - x)^2$$

$$w_{2\text{max}}(t + 1) \leftarrow w_{2\text{max}}(t)$$

$$\frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2}$$

$$4 \times (w_{\text{mean}} - x)^2$$

(75)

(76)

IF $w_{midR} < x \leq w_{\text{max}}$ THEN

The min values will be updated as in (77) and (78),

$$w_{1\min}(t + 1) \leftarrow w_{1\min}(t)$$

$$w_{2\min}(t + 1) \leftarrow w_{2\min}(t)$$

(77)

(78)

the mean values will be updated as in (79) and (80), and

$$w_{1\text{mean}}(t + 1) \leftarrow w_{1\text{mean}}(t)$$

$$\frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2}$$

$$4 \times (w_{\text{max}} - x)^2$$

$$w_{2\text{mean}}(t + 1) \leftarrow w_{2\text{mean}}(t)$$

$$\frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2}$$

$$4 \times (w_{\text{max}} - x)^2$$

(79)

(80)

the max values will be updated as in (81) and (82).

$$w_{1\text{max}}(t + 1) \leftarrow w_{1\text{max}}(t)$$

$$\frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_2)}{(2 - \mu_1 - \mu_2)^2}$$

$$4 \times (w_{\text{max}} - x)(x - w_{\text{mean}})$$

$$w_{2\text{max}}(t + 1) \leftarrow w_{2\text{max}}(t)$$

$$\frac{\partial f}{\partial \phi} \frac{2 \times (1 - \mu_1)}{(2 - \mu_1 - \mu_2)^2}$$

$$4 \times (w_{\text{max}} - x)(x - w_{\text{mean}})$$

(81)

(82)

IV. EXPERIMENT RESULTS AND ANALYSIS

The proposed AFNGLVQ-PI method was tested against two different types of data set: synthetic data set and data set that are available online and commonly used by other previous studies. For the synthetic data set, two synthetic data sets were used: a synthetic data set that overlaps greatly and a data set that overlaps greatly with a degree of skewness. The synthetic data sets are specifically created to observe the improvements between the proposed algorithms under several different overlap and skewness scenario.

On the other hand, the proposed method was also tested against eleven other data sets that are available online and commonly used in other studies: an arrhythmia data set, sleep data set, glass data set, Haberman data set, banknote data set, Pima Indians Diabetes data set, liver data set, wine data set, odor data set, a pinwheel data set and a yeast data set. Several of those data sets are retrieved from UCI Machine Learning Repository [58]. The arrhythmia and sleep data sets are specific biomedical data sets that are commonly used by the authors in their laboratory. Detail explanation of these data sets can be seen in appendix A - L.
A. RESULTS FROM SYNTHETIC DATA: NON-SKEWED

Using the non-skewed synthetic data set, we performed 9 different experiments in which 3 experiments for each 2-class data set, 3-class data set, and 4-class data set were conducted respectively. For the 2-class data set, the experiments were performed with 7, 4, and 2 features. For the 3-class data set, the experiments were performed with 7, 4, and 3 features. Whereas, for the 4-class data set, the experiments were performed with 8, 6, and 4 features. The main objective of this experiment is to compare the performance of the proposed FNGLVQ-PI algorithm with the GLVQ algorithm.

Figure 15 shows the results of the nine different experiments. The results show that FNGLVQ-PI produced better performance (i.e., accuracy) compared to the other 3 algorithms. Also, note that accuracy improved as the number of classes increase. The initial experiment using 2-class produced an average improvement of 1.99% for the FNGLVQ-PI when compared to the GLVQ algorithm. From the experiment using the 3-class data set, FNGLVQ-PI produced 3.46% increase in accuracy compared to the GLVQ algorithm. Finally, using the 4-class data set, the FNGLVQ-PI algorithm achieved 5.32% improvement in accuracy from the GLVQ algorithm. Overall, FNGLVQ-PI has an average of 3.27% accuracy higher than GLVQ.

B. RESULTS FROM SYNTHETIC DATA: SKEWED

The skewed synthetic data set is used to determine whether the adaptive version of the FNGLVQ-PI algorithm is more accurate in a highly overlapping skewed data set or not. Similar to the non-skewed synthetic data, we conducted 9 experiments in which 3 experiments for each 2-class data set, 3-class data set, and 4-class data set were conducted respectively. For the 2-class data set, the experiments were performed with 7, 4, and 2 features. For the 3-class data set, the experiments were performed with 6, 4, and 3 features. Whereas, for the 4-class data set, the experiments were performed with 8, 6, and 4 features.

The experiment results show that the AFNGLVQ-PI algorithm performed better than the FNGLVQ-PI algorithm. Furthermore, the FNGLVQ-PI also performed better than the GLVQ algorithm when performing classification using the overlapping skewed synthetic data set. In the experiment using 2 classes, the AFNGLVQ performed 0.36% better than the original FNGLVQ and the AFNGLVQ-PI performed 0.10% better than the AFNGLVQ algorithm. The improvement of the adaptive method is more substantial in the experiment using 3 classes where the AFNGLVQ performed 1.21% better than the FNGLVQ and the AFNGLVQ-PI performed 0.56% better than FNGLVQ-PI. Finally, using the 4-class data set, the AFNGLVQ algorithm achieved better performance than the FNGLVQ algorithm with a 4.62% increase of accuracy. The AFNGLVQ-PI also performed 1.74% better than the non-adaptive FNGLVQ-PI algorithm.

One interesting fact observed from this experiment is that the addition of the adaptive method to the FNGLVQ increased the classification accuracy substantially. However, adding the PI membership function to the AFNGLVQ did not make much difference in accuracy compared to the FNGLQV-PI. The experiment results show that the FNGLQV-PI is a better classification method than the AFNGLVQ algorithm. Therefore, this indicates that the adaptive method is an effective measure in dealing with the overlapping skewed data in a multi-classes dataset. Please see Figure 16 for the visualization of the results.
In this scenario, we compared the performance of the proposed classifier to the performance of other commonly used classifiers on synthetic non-skewed data set and skewed synthetic data set. The other classifiers are Support Vector Machine (SVM), Naive Bayes, Multilayer Perceptron (MLP), k-Nearest Neighbour (kNN), and J48. J48 is the implementation of C45 algorithm in the java programming language. The main objective of this scenario is to compare the performance of AFNGLVQ to commonly used classifiers.

Figure 17 shows the comparison of all tested algorithms’ performances (i.e., accuracy) on the non-skewed synthetic data set. On the experiment using the 2-class data set, 3-class data set, and 4-class data set, the AFNGLVQ-PI performed worse than the Naive Bayes where the difference in accuracy is between 0.15% and 0.26%. Naive Bayes performed better due to the characteristic of the data sets. Note that all of the data sets used in this scenario have a normal (Gaussian) distribution. Therefore, Naive Bayes could classify them accurately. On the experiment using the 2-class data set, the AFNGLVQ-PI performed better than SVM, MLP, kNN, and J48 classifiers. The AFNGLVQ-PI performed slightly better than the SVM, which is 0.01%, and performed better than MLP, kNN, and J48 with 0.33%, 0.13%, and 0.95% differences respectively. On the experiment using the 3-class data set, the AFNGLVQ-PI also performed better than SVM, MLP, kNN, and J48 with 0.03%, 1.66%, 0.17%, and 1.14 % differences respectively. Whereas, on the experiment using the 4-class data set, the AFNGLVQ-PI performed better than SVM, MLP, kNN, and J48 with 0.91%, 2.02%, 0.96%, and 1.34% differences respectively.

Figure 18 shows the comparison of all tested algorithms’ performances (i.e., accuracy) on the skewed synthetic data set. In this experiment, the AFNGLVQ-PI performed better than all five commonly used classifiers in all scenarios.
(i.e., 2-class data set, 3-class data set, and 4-class data set). The average accuracy difference is between 1.42% and 4.53%. On the experiment using the 2-class data set, the AFNGLVQ-PI performed worse than SVM (i.e., 0.06%) while performed better than Naive Bayes, MLP, kNN, and J48 with 1.12%, 0.47%, 0.40%, and 0.44% differences respectively. On the experiment using 3-class data set, the AFNGLVQ-PI performed better than all five classifiers (i.e., SVM, Naive Bayes, MLP, kNN, and J48) with 12.69%, 3.19%, 1.73%, 6.64%, and 3.50% differences respectively. Note the significant accuracy difference. On the experiment using the 4-class data set, the AFNGLVQ-PI also performed better than SVM, Naive Bayes, MLP, kNN, and J48 with 0.95%, 2.64%, 4.56%, 1.98%, and 0.32% differences respectively. In summary, based on those three variations of classes of the skewed dataset, the AFNGLVQ-PI has the highest accuracy among all of the tested algorithms in the skewed synthetic data set.

D. RESULTS FROM ARRHYTHMIA DATA SET

For the arrhythmia data set, the data has 157 features produced by wavelet decomposition from its original dataset. Figure 19 shows the experiment results from this scenario. Clearly, the adaptive version of FNGLVQ with the triangle and PI membership functions achieved higher accuracy than FNGLVQ, GLVQ, and LVQ-21. The AFNGLVQ achieved accuracy 3% higher than FNGLVQ. Compared to GLVQ, AFNGLVQ achieves accuracy 5% higher. From this experiment, it can be concluded that arrhythmic heartbeats can be successfully classified into their respective classes with a high degree of accuracy. In this experiment, AFNGLVQ-PI has a lower accuracy than SVM and MLP. However, the difference in accuracy is less than 0.04%. So, It can be said that in this case, AFNGLVQ-PI is still comparable to SVM and MLP. Furthermore, the accuracy of those three classifiers is above 99%. Compared to Naive Bayes, kNN, and J48, the proposed method has higher accuracy. The difference of accuracy is 6.41%, 3.75%, and 1.29% for Naive Bayes, kNN, and J48 respectively.

E. RESULTS FROM YEAST DATA SET

In this experiment, only four classes were used among the ten available. The four classes used were MIT, NUC, CYT, and ME3, which contain 244, 429, 463, and 163 instances respectively. Therefore, 1299 data instances were used in this experiment. Figure 20 shows the results from this experiment. From these results, it is clear that the yeast data set presents a fairly difficult problem. The GLVQ algorithm, which the authors believe is the most advanced algorithm derived from LVQ only gave 61.83% accuracy, which was higher than with the LVQ algorithm itself, which gave 53.12% accuracy. In this experiment, FNGLVQ has a lower accuracy than GLVQ which is 56.27%. However, the proposed AFNGLVQ-PI method surpassed LVQ, GLVQ, and FNGLVQ with 62.36% accuracy. This problem was also tested using a support vector machine (SVM), Naive Bayes, MLP, kNN, and J48 (Tree). The accuracy of these classifiers are 49.96%, 60.82%, 61.82%, 61.82%, and 59.20% for SVM, Naive Bayes, MLP, kNN, and J48. From these results, it can be concluded that AFNGLVQ-PI could classify this data set sufficiently well.

F. RESULTS FROM PINWHEEL DATA SET

For this data set, all algorithms were tested using all available instances. 5000 instances of balanced pinwheel data were used, and therefore each class of pinwheel data contained 1000 instances. Ten-fold cross-validation was used to validate the results. Figure 21 shows the experimental results. It is clear that initially, LVQ was the best algorithm because it won against GLVQ. Adding triangle membership fuzzification to GLVQ resulting in lower accuracy. It is shown in the chart the FNGLVQ has 85.66% only. However, after using the PI membership function, FNGLVQ-PI won over the GLVQ with 91.80% accuracy. PI membership function enhanced the FNGLVQ classifier by more than 6%. Then, after adding adaptive fuzzification, AFNGLVQ and AFNGLVQ-PI won over LVQ, GLVQ, and FNGLVQ. It can be said that adding adaptive fuzzification enhanced the FNGLVQ classifier 7% to 13%. The reasons why LVQ could initially win against GLVQ are very interesting. It seems that LVQ is quite good when the data are not high-dimensional or when they have a particular pattern, as in this synthetic pinwheel data set. In this experiment, the proposed method was also compared to commonly used classifiers. In this experiment, SVM and J48 have higher than AFNGLVQ-PI. However, the difference in accuracy is less than 1.2%. SVM achieved 99.76% accuracy whereas J48 achieved 99.36%. The other
three classifiers have lower accuracy than AFNGLVQ-PI. Naive Bayes achieved 88.78% accuracy, 9.98% lower than the proposed method. MLP achieved 90.32%, 8.35% lower than AFNGLVQ-PI. KNN achieved 97.94%, which is only slightly smaller than the proposed method.

G. RESULTS FROM ODOR DATA SET

In this experiment, all algorithms were tested using the odor dataset. The dataset has 16 features and 18 classes. Odor dataset has 1800 instances or 100 instances per class. Ten-fold cross-validation was used to validate the results. Figure 22 shows the experimental results. It is clear that initially, LVQ cannot classify odor data properly. It is shown by the chart that LVQ21 only achieved 30.67%. GLVQ escalates rapidly from LVQ21 with 98.61%. In this case, adding the fuzzification to GLVQ worsened the performance of the classifier. It is shown by the data that FNGLVQ and FNGLVQ-PI achieved 79.44% and 79.78% respectively. Besides, FNGLVQ and FNGLVQ-PI have no significant difference. It is due to the distribution of the odor data set. Most of the features have non-skewed distribution. Therefore applying the PI membership function has no significant effect compared to the triangle membership function. Adding adaptive fuzzification increases the accuracy of the classifiers up to 8% to 9%. It is shown by the figure that ANFGLVQ and AFNGLVQ-PI achieved 97.67% and 98.55% respectively. In this experiment, MLP achieved higher accuracy than AFNGLVQ-PI which is 99.72%. However, the difference is 0.83%. The other classifiers achieved lower accuracy than AFNGLVQ-PI. J48 achieve 96.72% accuracy, 1.83% less than AFNGLVQ-PI. Naive Bayes achieved 96.27% accuracy, 2.28% lower than the proposed method. SVM achieved 94.94% accuracy, 3.61% below the proposed method. KNN achieved only 65.94% accuracy, 32.61% below the proposed method.

H. RESULTS FROM SLEEP DATA SET

This experiment using the sleep dataset was conducted to show the improvement of the proposed algorithm compared to the AFNGLVQ and FNGLVQ previously proposed on [41]. Besides, the rest of the commonly used algorithm was also tested in the sleep dataset. The dataset has 16 features and 4 classes, and 4752 instances. Ten-fold cross-validation was used to validate the results. Figure 23 summarize the experimental results. From the experiment, LVQ and Naive Bayes are not good at classifying the sleep dataset. This is shown in the experiment results where the Naive Bayes and the LVQ algorithm could only reach an accuracy of 35.39% and 33.52% respectively. The GLVQ, FNGLVQ, AFNGLVQ, and FNGLVQ-PI did slightly better, ranging in the high 50% and bottom 60% in terms of accuracy. The SVM, MLP, and AFNGLVQ-PI did a fairly good job at classifying the sleep dataset with accuracy ranging in the mid 60%. The J48 and kNN algorithm did the best job overall with 74.2% and 74.83% respectively. The FNGLVQ and AFNGLVQ method with the membership function did better than the one without, with an increase of 2.1885% for the FNGLVQ-PI and 9.0698% for the AFNGLVQ-PI.

I. RESULTS FROM GLASS DATA SET

In this experiment, all algorithms were tested using the glass dataset taken from the UCI machine learning dataset. The dataset has 9 features and 6 classes. The glass dataset has only 212 instances. Ten-fold cross-validation was used to validate the results. Figure 24 shows the experimental results. The figure shows that initially, LVQ21 achieved 60.75% accuracy. Adding the stepest descent in LVQ21, GLVQ surpassed LVQ21 with 66.82% accuracy. Adding triangle membership function to GLVQ, FNGLVQ has a slightly lower accuracy than GLVQ which is 65.89%. However, by adding the fuzzy membership function to GLVQ, FNGLVQ-PI has higher accuracy than GLVQ even though only 0.47%. Using adaptive fuzzification worsened the performance of FNGLVQ as FNGLVQ-PI achieved only 50.93%. However, adaptive fuzzification enhanced FNGLVQ-PI as shown that.
AFNGLVQ-PI achieved 71.49%. This phenomenon is caused by the small number of dataset instances. Besides, the feature data is also spread non-uniformly. Therefore it is difficult to build triangle distribution. In this experiment, AFNGLVQ was also compared to SVM, Naive Bayes, MLP, kNN, and J48. Among those five classifiers, SVM has the highest accuracy which is 69.15%. The lowest accuracy is 49.53% achieved by Naive Bayes. Compared to those five classifiers, AFNGLVQ has better accuracy with a 2.34% to 21.96% difference.

J. RESULTS FROM WINE DATA SET
In this experiment, all algorithms were tested using the wine dataset taken from the UCI machine learning dataset. The dataset has 13 features and 3 classes. The wine dataset has 175 instances, there are 58 instances of class 1, 70 instances of class 2, and 47 instances of class 3. Result of ten folds cross-validation is shown in Figure 25. GLVQ has good accuracy with 97.14%. It was a rapid growth from LVQ21 which is 64% only. FNGLVQ and FNGLVQ-PI achieved 92.57% and 94.86% respectively. It shows that in this case fuzzification does not enhance the GLVQ, but weakened it. Besides, The PI membership function is better than the triangle membership function due to the distribution of the dataset. However, adding adaptive fuzzification to FNGLVQ-PI increases the performance of the classifier. As shown in the data that AFNGLVQ-PI achieved 97.71% that surpassed GLVA and FNGLVQ. In this experiment, the wine dataset is also tested to SVM, Naive Bayes, MLP, kNN, and J48. The result shows that SVM achieved the lowest accuracy which is only 45.71%. It is 52% below AFNGLVQ-PI and 18.29% below LVQ21. The other four classifiers’ accuracy is also lower than AFNGLVQ-PI. However, the accuracy is above 96%, slightly below AFNGLVQ.

K. RESULTS FROM HABERMAN DATA SET
In this experiment, all algorithms were tested using the Haberman dataset taken from the UCI machine learning dataset. The dataset has 2 features, 2 classes, and 305 instances. There are 224 instances of class 1, and 81 instances of class 2. Several features in this data set have a normal distribution, but the other features have skewed distribution. The result of ten-fold cross-validation experiment is shown in Figure 26. The result of the experiment has a similar trend to the previous experiment. GLVQ escalates rapidly from LVQ21 which is about 8.5% improvement. In this case, adding fuzzification does not increase accuracy. It is shown in the figure that FNGLVQ and feature is not. Experiment result for ten cross-validations is shown in Figure 26. As the initial classifier, LVQ21 achieved 69.83%. Applying optimization to LVQ21, GLVQ achieved 71.48% accuracy. After adding fuzzification to GLVQ, FNGLVQ and FNGLVQ-PI achieved 1% and 2.5% higher than GLVQ. Applying adaptive fuzzification, AFNGLVQ-PI achieved 1% higher than FNGLVQ-PI. However, adaptive fuzzification does not enhance FNGLVQ with triangle membership function as AFNGLQV has a lower accuracy than FNGLVQ itself. It tells us that to classify Haberman data more suitable approximated using PI membership function than triangle membership function. The proposed method has the same accuracy as MLP. Compared to SVM, Naive Bayes, and kNN, the proposed method still has better accuracy. The difference in accuracy is less than 1.7%. Compared to J48, in this experiment, the proposed method has 4.6% higher accuracy.
FNGLVQ-PI has a lower accuracy than GLVQ. However, the difference in accuracy is less than 1%. Applying adaptive fuzzification, AFNGLVQ-PI achieved better accuracy than FNGLVQ-PI. Besides, AFNGLVQ-PI also surpassed GLVQ. Adaptive fuzzification does not increase FNGLVQ with triangle membership function as shown in the figure that AFNGLVQ achieved 5% accuracy below FNGLVQ. The proposed method has better accuracy than SVM, Naive Bayes, MLP, kNN, and J48. Naive Bayes and MLP have slightly smaller accuracy than AFNGLQ-PI which the difference is less than 1%. AFNGLVQ-PI surpassed kNN and J48 by 1.95% and 1.31% accuracy respectively. As for SVM, this experiment achieved small accuracy which is only 65.19%. It is 11.34% below AFNGLVQ, and it is about 2.5% lower than LVQ21.

M. RESULTS FROM BANKNOTE DATA SET
In this experiment, all algorithms were tested using the banknote authentication dataset taken from the UCI machine learning repository. The dataset has 4 features, 2 classes, and 1371 instances. There are 761 instances of class 1, and 610 instances of class 2. Most of the features in this data set have skewed distribution. Figure 28 shows result of ten-fold cross-validation using this dataset. In this dataset, LVQ21 already has good accuracy which is 95.47%. The enhancement made by GLVQ increases the accuracy of 99.05%. The fuzzification does not increase the accuracy as shown in the figure that FNGLVQ and FNGLVQ-PI have 94.75% and 97.30% respectively. However, adaptive fuzzification increases the accuracy significantly as shown that AFNGLVQ and AFNGLVQ-PI achieved 99.34% and 100% respectively. The enhancement of accuracy is about 4.6% and 2.7% for triangle and PI membership respectively. Furthermore, AFNGLVQ and AFNGLVQ-PI surpassed GLVQ. This dataset is a satisfying formulation of adaptive fuzzification proposed in this paper that is used for a skewed overlap dataset. In this experiment, SVM has the same accuracy as the proposed method which is 100%. The proposed method is also better than MLP even though with a slight difference. Compared to kNN and J48, the proposed method also gain better accuracy. The difference in accuracy is more than 1% compared to those two classifiers. In this experiment, Naive Bayes achieved the lowest accuracy which is just 83.95%. It is due to the skewed distribution of the dataset that difficult to be handled by Naive Bayes.

N. RESULTS FROM LIVER DATA SET
In this experiment, all algorithms were tested using the Indian liver dataset taken from the UCI machine learning repository. The dataset has 10 features, 2 classes, and 583 instances. Class 1 has 416 instances, whereas class 2 has 167 instances. Most of the features in this dataset have skewed distribution. Besides, there is a feature that has 2 peaks separately. In other words, with this data, there are two data groups. The experiment result using ten folds cross-validation is shown in Figure 29. In this experiment GLVQ, FNGLVQ, FNGLVQ-PI, and AFNGLVQ-PI have the same accuracy which is 71.36%. It is 8.75% higher than the accuracy of LVQ21 as the initial classifier. AFNGLVQ has a slightly smaller accuracy than those four classifiers which is 71.18%. This event shows that fuzzification, adaptive fuzzification does not enhance the performance of the classifier. PI membership function also does not affect compared to the triangle membership function. It is caused by the fact that the dataset has two peaks. It is difficult to approximate the data just using one reference vector (codebook). To classify it properly, the classifier needs two or more reference vectors, one for each peak. However, in this case, the proposed method still has the same performance as GLVQ and FNGLVQ. In this case, SVM has the highest accuracy among them which is 72.38%, about 1% higher than the proposed method. Naive Bayes has the lowest accuracy which is only 55.74%, about 25.6% lower than the proposed method. Once again, it is because the dataset has two peaks and skewed distribution. Compared to kNN and J48, the proposed method still has higher accuracy about 1.5%. Whereas compared to MLP, the proposed method has 2.7% higher accuracy.

O. SCORING
The proposed AFNGLVQ-PI algorithm has been developed and verified against eleven different data sets. However, it is still difficult to determine which algorithm is better. Therefore, a point-based vote was used to identify the winner. An algorithm received 11 points if it emerged as the winner in any experiment. The other algorithms received 10, 9, 8, 7, to 1 if they finished in second, third, fourth, fifth, to last.
TABLE 1. Accuracy improvements of newly proposed algorithm and other algorithms compared to the LVQ algorithm.

| Algorithm      | Accuracy Improvement |
|----------------|----------------------|
| LVQ            | GLVQ                 |
| n2c-1          | 1.13%                |
| n2c-2          | 0.745%               |
| n2c-3          | 1.885%               |
| n3c-1          | 0.964%               |
| n3c-2          | -0.014%              |
| n3c-3          | 1.04%                |
| n4c-1          | 2.303%               |
| n4c-2          | 10.486%              |
| n4c-3          | 9.808%               |
| s-n2c-1        | 0.845%               |
| s-n2c-2        | 0.35%                |
| s-n2c-3        | 2.45%                |
| s-n3c-1        | 1.28%                |
| s-n3c-2        | 2.53%                |
| s-n3c-3        | 2.69%                |
| s-n4c-1        | -0.5225%             |
| s-n4c-2        | 7.0825%              |
| s-n4c-3        | 4.9075%              |
| eeg-157        | 10.67%               |
| yeast          | 8.776%               |
| pinwheel       | -5.6211%             |
| odor           | 67.94%               |
| glass          | 6.07%                |
| wine           | 33.14%               |
| Haberman       | 1.65%                |
| Pima           | 8.48%                |
| banknote       | 3.58%                |
| Liver          | 8.75%                |
| Average        | 6.944264286%         |

FIGURE 29. Experiment results from liver data set for all algorithms.

FIGURE 30. Results from point based vote for all algorithms in all data set scenario.

respectively. Using this method, it is hoped that a winner could be identified among the algorithms.

The results of this point-based vote are shown in Table 4 and Figure 30. Table 4 shows the point-based votes for the non-skewed dataset (first 9 data), synthetic skewed data set (next 9 data), the arrhythmia data sets with 157 features, the yeast data, the pinwheel data set, the odor data set, the glass data set, the wine data set, the Haberman data set, the Pima Indians Diabetes data set, the banknote data set, and the liver data set.

From Figure 30, it is clear that AFNGLVQ-PI came out ahead of the other algorithms. Furthermore, from Table 4, it is
TABLE 2. Accuracy improvements of newly proposed algorithm compared to the other algorithms.

| Data Set       | LVQ       | GLVQ       | FNGLVQ      | FNGLVQ-PI   | AFNGLVQ | AFNGLVQ-PI |
|----------------|-----------|------------|-------------|-------------|---------|------------|
| n2c-1          | 2.28 %    | 1.15 %     | 0.225 %     | -0.145 %    | 0.29 %  | -          |
| n2c-2          | 2.655 %   | 1.91 %     | 0.35 %      | -0.04 %     | 0.58 %  | -          |
| n2c-3          | 3.635 %   | 1.75 %     | 1 %         | 0.14 %      | 0.35 %  | -          |
| n3c-1          | 5.397 %   | 4.433 %    | 1.204 %     | 0.344 %     | 7.29 %  | -          |
| n3c-2          | 3.963 %   | 3.977 %    | 3.31 %      | 0.57 %      | 1.604 % | -          |
| n3c-3          | 4.697 %   | 3.657 %    | 2.32 %      | 0.76 %      | 1.37 %  | -          |
| n4c-1          | 5.34 %    | 3.037 %    | 1.355 %     | 0.552 %     | 2.676 % | -          |
| n4c-2          | 15.745 %  | 5.259 %    | 2.617 %     | 0.367 %     | 1.587 % | -          |
| n4c-3          | 17.42 %   | 7.612 %    | 6.102 %     | 0.807 %     | 1.757 % | -          |
| s-n2c-1        | 2.625 %   | 1.78 %     | 0.325 %     | -0.04 %     | 0.19 %  | -          |
| s-n2c-2        | 2.585 %   | 2.235 %    | 0.61 %      | 0.025 %     | 0.43 %  | -          |
| s-n2c-3        | 4.71 %    | 2.26 %     | 1.205 %     | 0.325 %     | 0.445 % | -          |
| s-n3c-1        | 0.9567 %  | -0.3233 %  | 1.03 %      | -0.2533 %   | -2.233 %| -          |
| s-n3c-2        | 5.92 %    | 3.39 %     | 3.4067 %    | 0.7767 %    | 2.9767 %| -          |
| s-n3c-3        | 8.6533 %  | 5.9633 %   | 2.7133 %    | 1.1666 %    | 0.76 %  | -          |
| s-n4c-1        | 8.7975 %  | 8.275 %    | 6.2375 %    | 2.2275 %    | 1.975 % | -          |
| s-n4c-2        | 17.65 %   | 10.5675 %  | 8.4225 %    | 1.58 %      | 2.3225 %| -          |
| s-n4c-3        | 18.22 %   | 13.3125 %  | 13.5475 %   | 1.415 %     | 10.82 % | -          |
| eeg-157        | 16.4361 % | 5.7661 %   | 3.0986 %    | 0.4131 %    | 0.3503 %| -          |
| sleep-4c       | 31.44 %   | 5.07 %     | 6.84 %      | 4.65 %      | 9.07 %  | -          |
| yeast          | 9.2379 %  | 0.4619 %   | 6.2356 %    | 1.5397 %    | 11.0855 %| -          |
| pinwheel(0.1)  | 5.6714 %  | 11.2925 %  | 13.0129 %   | 6.8716 %    | 0.1103 %| -          |
| odor-16f-3c(0.05) | 67.88 %  | -0.06 %    | 19.11 %     | 18.77 %     | 0.88 %  | -          |
| glass(0.05)    | 10.74 %   | 4.67 %     | 5.6 %       | 4.2 %       | 20.56 % | -          |
| wine(0.05)     | 33.71 %   | 0.57 %     | 5.14 %      | 2.85 %      | 15.42 % | -          |
| Haberman       | 5.25 %    | 3.6 %      | 2.62 %      | 0.98 %      | 19.67 % | -          |
| Pima(0.5)      | 8.87 %    | 0.39 %     | 1.05 %      | 0.78 %      | 7.3 %   | -          |
| banknote(0.5)  | 4.53 %    | 0.95 %     | 5.25 %      | 2.7 %       | 0.66 %  | -          |
| Liver          | 8.75 %    | 0 %        | 0 %         | 0 %         | 0.18 %  | -          |
| Average        | 11.51 %   | 3.90 %     | 4.27 %      | 1.87 %      | 4.11 %  | -          |

evident that among all 28 data set scenarios, AFNGLVQ-PI won for 10 data set scenarios, with an average score of 9.86 out of 11.00 points. FNGLVQ-PI came in second place, with an average value of 7.72. SVM, Naive Bayes, and MLP came in third, fourth, and fifth places with average values of 7.31, 6.82, and 6.41 points respectively. LVQ placed last with an average value of 1.82 points.

This result shows that the combination of the adaptive fuzzification method and the PI membership function can make LVQ and GLVQ better. The disadvantage of this additional method is that improvements were not substantial in certain scenarios, especially for the banknote, several skewed data sets, and several non-skewed data sets. However, the improvements for complex data sets like the arrhythmia data set, ECG-157, if AFNGLVQ-PI was compared with the LVQ and GLVQ algorithms. This shows that the use of the adaptive fuzzification method and the PI membership function is a good decision for classifying complex data. A complete presentation of these improvements can be seen in Tables 1. Table 1 shows the improvements attained by the proposed algorithm and other algorithms compared to the LVQ algorithm.

From Table 2, it is apparent that AFNGLVQ-PI is more accurate than the other algorithms. The average percentage shown at the bottom of the table shows the improvement of the proposed method compared to other algorithms. From the standard LVQ algorithm, the AFNGLVQ-PI has achieved the greatest improvements compared to the other algorithms, with a +5.132% average improvement. The greatest improvement made by AFNGLVQ-PI was +67.88% for the odor data set. The smallest improvement made by AFNGLVQ-PI was −0.32% for one of the synthetic skewed overlap data set. On the other hand, the smallest improvements overall were from the FNGLVQ-PI algorithm, which showed an average improvement of 1.87% only. The greatest improvement from that algorithm was +18.77% for the odor data set, and the
TABLE 3. Accuracy improvements of newly proposed algorithm compared to the other algorithms.

|         | AFNGLVQ-PI | SVM | NaiveBayes | MultiLayerPerceptron | kNN | J48 |
|---------|-------------|-----|------------|----------------------|-----|-----|
| n2c-1   | -           | 0.015 % | -0.185 % | 0.365 % | 0.09 % | 1.305 % |
| n2c-2   | -           | 0.12 % | -0.15 % | 0.3 % | 0.22 % | 0.95 % |
| n2c-3   | -           | -0.11 % | -0.12 % | 0.32 % | 0.065 % | 0.59 % |
| n3c-1   | -           | 0.067 % | -0.17 % | 1.324 % | 0.157 % | 0.83 % |
| n3c-2   | -           | 0 % | -0.14 % | 2.44 % | 0.057 % | 1.14 % |
| n3c-3   | -           | 0.01 % | -0.187 % | 1.23 % | 0.3 % | 1.44 % |
| n4c-1   | -           | -0.138 % | -0.255 % | 1.602 % | -0.013 % | 0.965 % |
| n4c-2   | -           | 0.447 % | -0.295 % | 2.244 % | 0.392 % | 1.337 % |
| n4c-3   | -           | 2.407 % | -0.235 % | 2.205 % | 1.897 % | 1.7 % |
| s-n2c-1 | -           | 0.185 % | 0.87 % | 0.635 % | 0.505 % | 0.795 % |
| s-n2c-2 | -           | 0.235 % | 0.91 % | 0.435 % | 0.455 % | 0.43 % |
| s-n2c-3 | -           | 0.11 % | 1.575 % | 0.365 % | 0.255 % | 0.115 % |
| s-n3c-1 | -           | 5.0807 % | 0.4387 % | -0.0163 % | 4.6567 % | 1.1167 % |
| s-n3c-2 | -           | 11.134 % | 3.244 % | 1.547 % | 6.657 % | 3.37 % |
| s-n3c-3 | -           | 21.8633 % | 5.8473 % | 3.6533 % | 8.6203 % | 6.0103 % |
| s-n4c-1 | -           | -0.1825 % | 4.6105 % | 5.8175 % | 1.2105 % | 0.9025 % |
| s-n4c-2 | -           | 0.7425 % | 2.4625 % | 4.4625 % | 2.0225 % | 0.3275 % |
| s-n4c-3 | -           | 2.278 % | 0.818 % | 3.388 % | 2.705 % | 0.53 % |
| eeg-157 | -           | -0.1539 % | 6.4128 % | -0.3682 % | 3.7453 % | 1.2933 % |
| sleep-4c | -           | -0.3578 % | 29.6717 % | -1.1364 % | -9.8695 % | -9.2382 % |
| yeast   | -           | 13.395 % | 1.5397 % | 0.5357 % | 0.5389 % | 3.1563 % |
| pinwheel(0.1) | -           | -1.093 % | 9.8922 % | 8.5119 % | 0.7504 % | -0.6899 % |
| odor-lle65-3c(0.05) | -  | -3.61 % | 2.28 % | -1.17 % | 32.61 % | 1.83 % |
| glass(0.05) | -           | 2.34 % | 21.96 % | 6.07 % | 8.41 % | 5.6 % |
| wine(0.05) | -           | 52 % | 0.57 % | 1.14 % | 1.14 % | 1.71 % |
| Haberman | -           | 1.31 % | 0.98 % | 0 % | 0.78 % | 1.95 % | 1.31 % |
| Pima(0.5) | -           | 11.34 % | 0.39 % | 0.78 % | 0.78 % | 1.95 % |
| banknote(0.5) | -  | 0 % | 16.05 % | 0.22 % | 1.17 % | 1.68 % |
| Liver   | -           | -1.02 % | 15.62 % | 2.75 % | 1.38 % | 2.58 % |
| Average | -           | 4.33232069 % | 4.291393103 % | 1.706413793 % | 2.561968966 % | 1.271224138 % |

The smallest improvement was −0.25% for the synthetic skewed data set. In Table 3, it is apparent that AFNGLVQ-PI also made the great improvement to the other commonly used algorithm,
with 4.33%, 4.29%, 1.70%, 2.56%, and 1.27% improvements for SVM, Naive Bayes, MLP, kNN, and J48 respectively. Compared to other commonly used algorithms, the greatest improvement made by AFNGLVQ-PI was 29.96% for the sleep data set. The least improvement made by AFNGLVQ-PI was —9.86% for the sleep data set by kNN algorithm.

From these experimental results, it is clear that AFNGLVQ-PI succeeded in maintaining its performance for all data sets used in the experiments. Positive improvements were successfully produced by AFNGLVQ-PI compared to both the LVQ and GLVQ algorithms. Although the improvements yielded by AFNGLVQ-PI were not very high compared to GLVQ, the proposed AFNGLVQ-PI algorithm did not exhibit negative improvement like GLVQ when it was compared to the LVQ algorithm.

P. COMPLEXITY
The AFNGLVQ-PI has been shown to excel compared to other methods. One may wonder whether this improvement in accuracy comes with some additional cost in complexity. However, since the number of parameters remains the same compared to AFNGLVQ, and the derivatives of the PI function takes a simple form to compute, the complexity remains the same. For all the algorithms in consideration, the operations in each iteration consist of parameter updates which required the evaluation of a similarity function. In the case of LVQ and GLVQ, the similarity measure is done by computing Euclidean distance, while in FLVQ, FNGLVQ, FNGLVQ-PI, AFNGLVQ, and AFNGLVQ-PI it is done by evaluating the difference in membership values.

Table 5 summarizes the complexity of the algorithms in one iteration for a k features. Suppose there are k features and c classes. LVQ and GLVQ adjust one parameter for each feature and each class, therefore ck parameters need to be updated at every iteration for each class. FLVQ, FNGLVQ, and FNGLVQ-PI use membership functions which are defined by three parameters (min, mean, max). However, only one of the parameters is optimized at one time, therefore they also have ck times parameter update. AFNGLVQ and AFNGLVQ-PI also use membership functions, but all three parameters are used, which requires 3 ∗ ck parameters to be updated. All the similarity measures and the update processes for each parameter and each class belong to the same complexity class O(1), so the complexity in each iteration depends and grows linearly on the number of parameters to be updated and the number of target classes.

Q. ADVANTAGES
There are several advantages of using AFNGLVQ-PI architecture for classification. One of the advantages is the speed of training as well as the speed of testing. AFNGLVQ-PI is based on a very simple type of Neural Network based on LVQ and GLVQ. This makes it possible to train data consisting of hundreds of features in a short amount of time. The complexity of training grows linearly with the number of features, the number of target classes, and the number of training iterations, while the complexity for testing amounts only to ck membership function evaluation; where k is the number of features and c is the number of target classes.

It is also important to note that a small number of iteration is required to achieve good results, all the experiments were performed with the maximum number of iteration set to 150.

Another advantage is that after the training is finished, an approximate distribution of the feature values describing a certain class can be obtained from the tuned parameters of the PI-membership function. This can provide intuitive insight into what describes a certain class in terms of certain features.

Furthermore, the more the membership function fits the distribution of the training data, the more relevant the feature is in identifying a certain class. Hence, the method can also be used to help manual feature selection.

V. LIMITATION
The AFNGLVQ-PI has been demonstrated clearly, along with its performance, complexity, and advantages. Despite its strong points, the proposed method has its limitation. First, the method is tested in 1-dimensional (1-D) data. The method has not been tested in multi-dimension data such as images and 3D point clouds. For cases in multi-dimensional data, the proposed method needs a transformation to transform n-D data into 1-D data. Second, the current version of the proposed method has one reverence vector (codebook). It is challenging to observe the performance of the AFNGLVQ-PI with multiple reference vector. Therefore the authors recommend developing the multiple reference vector version of the AFNGLVQ-PI.

VI. CONCLUSION
The development of AFNGLVQ-PI has been proposed, tested, and thoroughly discussed in this paper. Overlapping data and various types of data distribution have been the main problems encountered when implementing adaptive fuzzification and the PI membership function in the GLVQ algorithm. The synthetic non-skewed data experiment has proved that the FNGLVQ-PI algorithm has been substantially better than the GLVQ algorithm by an average of 3.44% in the 3-classes data, 4.74% in the 4-classes data, and 1.533% in the 2-class data. The skewed synthetic data have proved that the proposed AFNGLVQ-PI has been better in comparison to the FNGLVQ-PI data with an average increase of 0.80%. The 4 other data sets used in these experiments have
varied from commonly used data sets from the UCI Machine Learning repository to more complex biomedical data sets. The results using those have shown that the AFNGLVQ-PI algorithm has performed better than other versions of these algorithms, although the improvements have not been substantial in all cases. However, AFNGLVQ-PI has succeeded in yielding positive improvements for all data set scenarios, which could not be said of the GLVQ algorithm.

VII. FUTURE WORKS
From the current achievement, the authors recommended the following ideas for future study. First, investigating the performance of AFNGLVQ-PI in 2D and 3D data. The authors also recommend investigating the performance of AFNGLVQ-PI in various dataset characteristics e.g. imbalanced dataset, multi-modal dataset, higher overlapping dataset, time-series dataset, etc. Second, the authors recommend developing an enhanced version of AFNGLVQ-PI with multiple reference vectors per class. The reference vectors may be generated in the initiation process (before the training) or generated by evolving mechanisms during the training process. Third, the authors recommend investigating the use of other membership functions e.g. Gaussian or Trapezoidal membership function. Fourth, the authors recommend developing multiple hidden layer version of AFNGLVQ to enhance the classification ability of the method.

APPENDIX A
ARRHYTHMIA DATA SET
MIT-BIH provides free access to an arrhythmia database on the physionet site [49], which has been used extensively by many studies to test the ability of various algorithms to discriminate arrhythmic heartbeat [50]. This database contains 48 records from 47 patients. The subjects were 25 men aged 32 to 89 years and 22 women aged 23 to 89 years. Each record contains two channels of 30-min ECG lead signal originating from the upper MLII lead and lower leads V1/V2/V4/V5. The frequency of the ECG data is 360 Hz. Fifteen classes are available, but three classes were removed from this experiment because the number of samples was too small. The remaining twelve classes that were processed are:

- Normal (NOR)
- Left Bundle Branch Block (LBBB)
- Right Bundle Branch Block (RBBB)
- Atrial Premature (AP)
- Aberrant Atrial Premature (AAP)
- Nodal (Junctional) Premature (NP)
- Supraventricular Premature (SVP)
- Premature Ventricular Contraction (PVC)
- Fusion Ventricular and Normal (FVN)
- Atrial Escape (AE)
- Ventricular Escape (VE)
- Fusion of Paced and Normal (FPN)

Before classification, the arrhythmia data needed to be preprocessed. The procedures performed for the arrhythmia data set were baseline wander removal (BWR), beat extraction, outlier removal, and feature extraction. The mentioned processes can be seen at A. After preprocessing processes, the arrhythmia data were ready to be classified. The feature extraction process, which is the wavelet-decomposition algorithm, yielded multiple numbers of features (300, 157, 86, 50, 32, 24, and 23) that will be used in classification. To visualize the data distribution, a 2D projection of the arrhythmia data was plotted from two features obtained using principal component analysis (PCA). Figure 31 shows this 2D data projection, which reveals that the data indeed overlap.

A. BASELINE WANDER REMOVAL
Baseline wander is a kind of low-frequency noise in the ECG signal caused by various activities of human internal organs such as breathing or limb movement. To estimate and remove this noise, a cubic spline method was used. Baseline noise was reduced by simply subtracting the original signal from the estimated signal. Figure 32 illustrates this method.

B. BEAT EXTRACTION
The ECG signal data obtained from the repository were divided into single heartbeats. To do this, the annotation of the R peak in every signal was used. From the R point, 149 points were extracted to the left and 150 points to the right to estimate a single heartbeat. Thus, a region of 300 points in width was obtained to represent a single estimated beat. This technique is called the cutoff technique and is illustrated in Figure 33.

C. OUTLIERS REMOVAL
Outlier removal eliminates beats that are outside the distribution of normal beats. Such beats often disrupt the learning process. To do this, a simple procedure called the interquartile range (IQR) was used. This technique uses a percentile as a boundary to determine outliers. Figure 34 illustrates the
outlier removal process using IQR. After the IQR application, the number of beats was decreased by almost half. Table 6 shows the number of beats before and after outlier removal.

![Figure 33. Illustration of cutoff technique.](image)

![Figure 34. Illustration of outliers removal process by using IQR.](image)

**Table 6.** Samples Number per Class Before and After Outliers Removal.

| Class                    | With Outliers | No Outliers |
|--------------------------|---------------|-------------|
| Normal                   | 74,754        | 40,113      |
| Left bundle branch block | 8,071         | 4,402       |
| Right bundle branch block| 7,255         | 2,955       |
| Premature ventricular contraction | 7,123 | 3,737 |
| Paced                    | 3,619         | 1,898       |
| Atrial premature         | 2,546         | 1,236       |
| Fusion of ventricular and normal | 802  | 412        |
| Fusion of paced and normal | 260  | 105        |
| Nodal                    | 229           | 148         |
| Aberrant atrial premature | 150           | 61          |
| Ventricular escape       | 106           | 103         |
| Node (Functional) premature | 83         | 37          |

**D. FEATURE EXTRACTION**

To reduce the size of the training data set and consequently computation time and storage space, a feature reduction technique was used to decrease the number of features. Originally, there were 300 point features for each data point. After feature extraction, this number was reduced to fewer than 100 points. Based on previous studies, a discrete wavelet transformation with Daubechies order 4 was used as a basis function. The wavelet transform of a signal $f(x)$ is defined as:

$$W_s f(x) = f(x) * \psi_s(x) = \frac{1}{s} \int_{-\infty}^{+\infty} f(t) \psi_s(x - s t) dt$$

(83)

where $s$ is the scale factor and $\psi_s(x) = \frac{1}{s} \psi(x/s)$ is the dilation of a basic wavelet $\psi(x)$ by a scale factor $s$. Let $s = 2^j$ ($j \in \mathbb{Z}$, $\mathbb{Z}$ is the integral set); then WT is called a dyadic wavelet [51]. The dyadic wavelet of a digital signal $f(n)$ can be calculated using the Mallat algorithm:

$$S_{2^j} f(n) = \sum_{k \in \mathbb{Z}} h_k S_{2^{j-1}} f(n - 2^{-j-1} - k)$$

(84)

$$W_{2^j} f(n) = \sum_{k \in \mathbb{Z}} g_k S_{2^{j-1}} f(n - 2^{-j-1} - k)$$

(85)

where $S_{2^j}$ is a smoothing operator, $S_{2^j} f(n) = a_j$ is the low-frequency component (or approximation in wavelet jargon) of the original signal, and $W_{2^j} f(n) = d_j$ is the high-frequency component (or detail) of the original signal. The wavelet-decomposition algorithm was used with multi-level decomposition to obtain a signal with multiple numbers of features (300, 157, 86, 50, 32, 24, and 23).

**APPENDIX B**

**SYNTHETIC DATA SETS: SKEWED AND NON-SKEWED**

**A. NON-SKEWED SYNTHETIC DATA SETS**

In this study, 2 types of synthetic data are generated to justify the algorithm that has been proposed. The non-skewed synthetic data has three types of data: The 2 class data, the 3 class data, and the 4 class data. All three types of data have a Gaussian distribution. The Gaussian form of the data is used because most of the real-world data that have ever existed are Gaussian, and it is also the most commonly used to assume the distribution of data. This synthetic data have a high overlap degree. This dataset will be used to prove whether the use of the fuzzy concept will improve the accuracy of the classification. In other words, it will be used to determine whether the FNGLVQ and the FNGLVQ-PI will perform better than the GLVQ algorithm in an overlapping data condition.

For the 2 class data set, there are 3 types of overlap conditions. The first condition is the overlap happens at the end of the distribution, as show in Figure 35 (left). In the other conditions, the data set have the same mean but different standard deviation as shown in Figure 35 (mid and right). The scatter plot of each feature is shown in Figure 36.

In the synthetic dataset consisting of three classes, there are several kinds of overlapping conditions. Among them is a condition in which the distribution of the features of a class overlaps with the features of the other class. The condition is shown on the left Figure 37. Other conditions are the conditions in which most of the overlap involves only two classes,
FIGURE 35. Feature distribution of non-skewed synthetic data set with 2 class.

FIGURE 36. Scatter plot of non-skewed synthetic data set with 2 class.

FIGURE 37. Feature distribution of non-skewed synthetic data set with 3 class.

FIGURE 38. Scatter plot of non-skewed synthetic data set with 3 class.

while the other class only overlap in a small amount of data. The condition is indicated by the right figure 37. The scatter plot for the two conditions overlap is shown in Figure 38.

FIGURE 39. Feature distribution of non-skewed synthetic data set with 4 class.

FIGURE 40. Scatter plot of non-skewed synthetic data set with 4 class.

FIGURE 41. An example of and overlap condition in a non-skewed data set with 4 classes.

B. SKEWED SYNTHETIC DATA SETS

The synthetic non-skewed dataset consisting of four classes have a wide range of overlap conditions. Most of these conditions are an extension of the overlap condition on previous data (class 2 and class 3). There are two variations of new overlap conditions used in this dataset. The first condition is that there are two pairs of classes, which have high overlap, while the other four classes have a low level of overlap. The condition is shown in Figure 39 left. The figure shows that class 1 and class 3 have a high overlap. Similarly, classes 2 and 4 are overlapping. However, the level of overlap for class 1,2,3 and 4 is only a small amount. Other conditions are the conditions under which the four classes have a very high overlap level. This condition is shown in Figure 39 right. In the picture, it is shown that the class 1,2,3 and 4 overlap because the mean distance of the four classes was very close. The scatter plot for the two types of overlap condition is shown in Figure 40.

The skewed dataset is obtained from the non-skewed dataset by adding some skewness on some features. By adding skewness, the classifier will difficultly distinguish them. Besides, only several features receive skewness. This dataset is utilized to show how effective does it improve the FNGLVQ in classification accuracy with the concept of adaptivity given during training. Some examples of skewed features in this experiment can be found in Figure 42. The scatter plot of the dataset is shown in Figure 43.

FIGURE 42. Example of overlapping data in an ecg data set.

FIGURE 43. Scatter plot of overlap data set with a degree of skewness in 2, 3, and 4 class.
APPENDIX C
SLEEP DATA SET

Sleep stages and disorders are usually examined using polysomnography (PSG). The examination involves attaching several sensors to various parts of the human body, which is highly inconvenient. However, recent studies have shown that the ECG signal alone could be used as a promising feature to discriminate sleep stages. This study was pioneered by the work of Chazal et al. [52]. The ECG signal is much simpler to obtain and process, and therefore it is an ideal data source for the examination of sleep stages and disorders.

The authors collaborated with Mitra Kemayoran Hospital in Jakarta to collect sleep records from 10 subjects, of whom 7 were male, aged from 23 to 40, which were called the Mitra data set. Each record represents nine hours of sleep, and 21 physiological channels were used to monitor each subject, including ECG, EEG, and EMG. Sleep stages were annotated once every 30 seconds (one epoch) as recommended by AASM [53].

For sleep-stage problems, at least three different approaches can be used to extract features from the ECG signal. The first approach uses heart-rate variability (HRV) features calculated from the RR interval. The second method uses HRV features extracted from ECG-derived respiratory (EDR) features, and the last approach uses features calculated directly from the raw ECG signal.

To obtain HRV features from the RR interval, the length between every pair of adjacent R peaks in the ECG signal must be calculated. To do this, the positions of the R peaks in the ECG signal must be known. The MIT-BIH data already have annotations of every R peak made by doctors. However, the Mitra data set did not have these, and therefore the ecgpuwave tool of WFDB was run to annotate the R peaks automatically [49]. This program is based on the improved version of the Pan and Tompkins algorithm [54]. From the ten records of the Mitra data set, three records were excluded because the tools failed to provide correct QRS annotations. Two other records were also excluded because they were recorded from healthy patients, yet the data scarcely resembled healthy heartbeat waves. Hence, only five records were used for this experiment. Figure 44 shows an example diagram of ECG signals. It shows the important segment of the ECG, which is labeled as the QRS complex, and the distance between two QRS complexes (or their R peaks) as the RR interval.

From the list of 30-second RR intervals, the HRV features could be calculated. These could be differentiated into time-domain and frequency-domain features. The time-domain features were:

- NN/RR, the ratio between the number of normal RR intervals (NN) and the total number of RR intervals.
- AVNN, the average of all NN intervals within one epoch.
- SDNN, the standard deviation of all NN intervals within one epoch.
- RMSSD, the square root of the mean of the squared differences between adjacent NN intervals.
- PNN, the percentage of differences between adjacent NN intervals that were greater than 50 ms, and the frequency-domain features were
  - TOTPWR, the total spectral power of all NN intervals up to 0.04 Hz
  - ULF, the total spectral power of all NN intervals up to 0.003 Hz
  - VLF, the total spectral power of all NN intervals between 0.003 and 0.04 Hz
  - LF, the total spectral power of all NN intervals between 0.04 and 0.15 Hz
  - HF, the total spectral power of all NN intervals between 0.15 and 0.4 Hz
  - LF/HF, the ratio of low- to high-frequency power.

Furthermore, the results from Yilmaz et al. [55], who used additional features derived from HRV features, were also incorporated. These features were:

- Median, the median length of the RR interval.
- IQR (interquartile range), the difference between the 75th and 25th percentiles of the RR interval value distribution.
- MAD (mean absolute deviation), the mean of the absolute values obtained by subtracting the mean of the RR interval values from the total of the RR interval values in one epoch.

For the HRV features calculated from EDR, an approach proposed by Chazal et al. [52] was used. During the breathing phase, as the lung is filled with air and then emptied, the body ECG is influenced by electrode motions related to the heart and to changes in thoracic electrical impedance. The ECG signal was therefore filtered twice using median filters of 200 ms and 600 ms to produce a baseline for the ECG signal. The original ECG signal was then subtracted from the resulting baseline signal to produce a baseline-corrected ECG signal. Then the area under each QRS complex up to 100 ms before and after the R peak was calculated to generate the EDR signal. After this, the same set of HRV features could be extracted from this signal, as explained earlier.

As for the third set of features, these were calculated directly from the ECG signal as proposed in other studies [56], [57]. The list of features is as follows:

- Energy: $\sum_i x_i^2$
- 4th Power: $\sum_i x_i^2$
- Curve Length: $\sum_i ||x_i - x_{i-1}||$
- Nonlinear Energy: $\sum_i -x_i * x_{i-2} + x_{i-1}^2$
- Peak Power: $\max(\text{PSD})$
- Peak Frequency: $\text{index}(\max(\text{PSD}))$
- Mean PSD: $\mean(\text{PSD})$
- Median PSD: $\median(\text{PSD})$
- Spectral Entropy: $\sum_i \text{PSD}_j * \log(\text{PSD}_j)$
- Katz Fractal Dimension: $\log N / \left( \log N + \log(\max(\sqrt{(x_i-x_0)^2 + x_{i+1}^2})) \right) / \sum_i \sqrt{(x_i-x_{i+1})^2 + x_{i+1}^2}$
- Detrended Fluctuation Analysis: the slope of the line relating the log of root-mean-square fluctuation to log N.
- Higuchi Fractal Dimension: based on the Higuchi algorithm
- Hjorth Mobility: $\sqrt{\frac{\sum_i \frac{1-n}{n} - 1}{\sum \frac{1}{n}}}$
- Hjorth Complexity: $\sqrt{\frac{\sum_i \frac{1}{n} - 2 \frac{1}{n^2} - \frac{1}{n^2}}{\sum_i \frac{1}{n} - 1}}$
- Petrosian Fractal Dimension: $\log N / \left( \log N + \log(\frac{N}{N+0.4}) \right)$

For this study, all the features discussed above were combined and used, and therefore a total of 43 features were used in these experiments. Figure 45 shows a projection of the sleep data set in 3D space. The plot reveals certain characteristics of the sleep data set: the data are scattered, and there are strong indications of overlapping data for each class. This data set turned out to be one of the most challenging to be examined in this study.

**APPENDIX D**

**SYNTHETIC PINWHEEL DATA SET**

This synthetic data set was generated from the pinwheel model. The pinwheel data set used here consisted of balanced data in which all five classes contained the same number of instances, 500 data points for each class. Using the same data set, the performance of AFNGLVQ-PI was compared with those of the other classification algorithms. This data set was used to determine how well the proposed method could read the data’s distribution and shape. A scatter plot of the pinwheel data set can be seen in Figure 46 and the histogram can be seen in Figure 47.

![Figure 45. 3D plot of sleep stage data set.](image)

**APPENDIX E**

**YEAST DATA SETS**

The yeast data sets are all available online on the UCI Machine Learning repository Web site [?]. This repository is famous among machine-learning researchers because it offers many types of data set that can be used freely for research. Some of these data sets are very often used by researchers. In these experiments, the IRIS, yeast, and letter-image data sets were used.

The yeast data set contains eight features and is divided into ten different classes. However, six of the classes were excluded from these experiments to determine how well the proposed algorithm could classify a mid-high dimensional data set (eight features) with only a small number of classes (less than ten). Only four classes were included: MIT, NUC, CYT, and ME3, with 244, 429, 463, and 163 data points in each class. Figure 48 shows a projection of the yeast data set in 2D space and the histogram can be seen in Figure 49.

The experiments were performed against all five data sets introduced in the previous section. A ten-fold cross-validation test scheme was used. Six different algorithms were tested to solve all five classification problems: LVQ, GLVQ, FNGLVQ, FNGLVQ-PI, AFNGLVQ, and AFNGLVQ-PI. FNGLVQ and FNGLVQ-PI were developed to address biomedical classification problems such as ECG arrhythmia data set and sleep data set, whereas the proposed algorithms, AFNGLVQ and AFNGLVQ-PI, were proposed for more general classification problems. This section will present all the results from these experiments, with several charts and graphs.
and a brief discussion to provide more detail about each set of results.

APPENDIX F
ODOR DATA SETS
The odor data set a mixture of several different odors. The mixture consists of 33.3% of the first odor, 33.3% of the second odor, and 33.3% of alcohol. The alcohol concentrations vary from 0% to 70%. 3 odors were used in this data, citrus, cananga, and rose. This data set has 16 features, where each feature will represent the aroma that it represents. In this study, there are 18 classes used in the data set. Figure 50 is a plotted data set of the odor data set and the histogram can be seen in Figure 51.

APPENDIX G
PIMA INDIAN DIABETES DATA SETS
The Pima data set is a classification data set obtained from the UCI Machine Learning Repository. This data set is gathered by the National Institute of Diabetes and Digestive and Kidney Diseases of female patients more than 21 years old of Pima Indian heritage. This data set has eight attributes. The attributes are the number of times the patient is pregnant, the Plasma glucose concentration a 2 hours in an oral glucose test, diastolic blood pressure (in mm Hg), triceps skinfold thickness (mm), 2-hour serum insulin (mm U/ml), BMI (kg/m2), diabetes pedigree function, age, and a class variable. There is 768 number of instance in this dataset. Figure 52 is a plotted diagram of the wine data set and the histogram can be seen in Figure 53.

APPENDIX H
WINE DATA SETS
The wine data set is a classification data set that have 13 attributes with 178 instances. It is the analysis of wine chemicals that grow in the same Italian region but from three different cultivars. The attributes for this dataset are alcohol, malic acid, ash, the alkalinity of the ash, magnesium, total phenols, flavonoids, nonflavanoids phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines, and proline. Figure 54 is a plotted diagram of the wine data set and the histogram can be seen in Figure 55.
APPENDIX I
ILPD (Indian LIVER Patient) DATA SETS
The Indian Liver Patient Dataset (ILPD) is a classification data set with 10 attributes and \(\text{583} \) instances. This data set is obtained from a total of \(\text{583} \) patients, where \(\text{416} \) of those patients are liver patients. Of all the patients, \(\text{441} \) patients are male and \(\text{142} \) patients are female. This data set has a class label called selector, which divides the data into two groups, whether the patient is a liver patient or not. As mentioned previously, this data set has eleven attributes, which are age, gender, total bilirubin, direct bilirubin, alkaline phosphatase, alanine aminotransferase, aspartate aminotransferase, total proteins, albumin, and albumin to globulin ratio. Figure 56 is a plotted diagram of the liver data set and the histogram can be seen in Figure 57.

APPENDIX J
HABERMAN DATA SETS
The Haberman survival data set contains the study data on the survival of a patient who has done surgery on breast cancer. The study was done from 1958 to 1970 at the University of Chicago’s Billings Hospital. There are 3 attributes in this classification data set, with 306 instances. The three attributes in this dataset, are the age of the patient, operation year of the patient, and the number of positive axillary nodes detected. Figure 58 is a plotted diagram of the Haberman data set and the histogram can be seen in Figure 59.

APPENDIX K
GLASS DATA SETS
The glass data set has 10 attributes, with 214 instances. This data set is a study of glass classification that is used at the scene of a crime. In a criminal investigation, glass can be used as evidence. There are ten attributes in this data set: the id number, the refractive index, the sodium content, magnesium content, aluminum content, silicon content, potassium content, calcium content, barium content, and the iron content of the suspected glass. Figure 60 is a plotted diagram of the glass data set and the histogram can be seen in Figure 61.

APPENDIX L
BANKNOTE DATA SETS
The banknote authentication data set contains data that was extracted from banknotes to evaluate the authentication
process of the banknote. The image size is extracted from is 400 x 400 pixels with a resolution of 660 dpi. The features were extracted using wavelet. This data contains 1372 instances with 5 attributes. As mentioned before, there are five attributes in this data set: variance of the wavelet transformed image, the entropy of the image, and the class of the image. Figure 62 s a plotted diagram of the banknote data set and the histogram can be seen in Figure 63.
[24] M. Sarma and K. K. Sarma, “Speaker identification model for assamese language using a neural framework,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Aug. 2013, pp. 1–7.

[25] Z. Xinye, A. Jubai, and Y. ZhiFeng, "A method of LVQ network to detect vehicle based on morphology," in Proc. WRI Global Congr. Intel. Syst., 2009, pp. 339–343.

[26] Y. M. George, H. H. Zayed, M. I. Roushdy, and B. M. Elbagoury, “Perception rule improvement on FIMT-DD for large traffic data stream,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2016, pp. 5161–5167.

[27] A. Sato and K. Yamada, “Generalized learning vector quantization,” in Proc. Adv. Neural Inf. Process. Syst., 1996, pp. 423–429.

[28] B. Hammer and T. Villmann, “Generalized relevance learning vector quantization,” Neural Netw., vol. 15, nos. 8–9, pp. 1059–1068, Oct. 2002.

[29] T. Fukuda, F. Arai, and B. Kusumoputro, “Artificial odor classification and comparison with fuzzy algorithm LVQ in artificial odor discrimination system,” ISA Trans., vol. 41, no. 4, pp. 395–407, Oct. 2002.

[30] W. Jatmiko, T. Fukuda, F. Arai, and B. Kusumoputro, “Artificial odor discrimination system using multiple quartz resonator sensors and various neural networks for recognizing fragrance mixtures,” IEEE Sensors J., vol. 6, no. 1, pp. 223–233, Feb. 2006.

[31] S. Derhami and A. E. Smith, “HGA: Hybrid genetic algorithm in fuzzy rule-based classification systems for high-dimensional problems,” Appl. Soft Comput., vol. 41, pp. 91–93, Apr. 2016.

[32] C. Caraveo, F. Valdez, and O. Castillo, “Optimization of fuzzy controller design using a new bee colony algorithm with fuzzy dynamic parameter adaptation,” Appl. Soft Comput., vol. 43, pp. 131–142, Jun. 2016.

[33] Y.-Y. Shen, Y.-M. Zhang, X.-Y. Zhang, and C.-L. Liu, “Online semi-supervised learning with learning vector quantization,” Neurocomputing, vol. 399, pp. 467–478, Jul. 2020.

[34] J. Ravichandran, M. Kaden, S. Saralajew, and T. Villmann, “Variants of DropConnect in learning vector quantization networks for evaluation of classification stability,” Neurocomputing, vol. 403, pp. 121–132, Aug. 2020.

[35] N. Floropoulos and A. Tefas, “Complete vector quantization of feedforward neural networks,” Neurocomputing, vol. 367, pp. 55–63, Nov. 2019.

[36] I. Hermawan, M. I. Tawakal, J. M. A. Setiawan, I. Habibie, and W. Jatmiko, “Adaptive multi codebook fuzzy neuro generalized learning vector quantization for sleep stages classification,” in Proc. Int. Conf. Adv. Comput. Sci. Inf. Syst. (ICACSIS), Sep. 2013, pp. 431–436.

[37] H. George John and P. Langley, “Estimating continuous distributions in Bayesian classifiers,” in Proc. 11th Conf. Uncertainty Artif. Intell. San Francisco, CA, USA: Morgan Kaufmann, 1995, pp. 338–345.

[38] T. Cover and P. Hart, “Nearest neighbor pattern classification,” IEEE Trans. Inf. Theory, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” Inst. Cogn. Sci., California Univ., San Diego, La Jolla, CA, USA, 1985.

[40] C. Cortes and V. Vapnik, “Support-vector networks,” Mach. Learn., vol. 20, no. 3, pp. 273–297, 1995.

[41] R. Quinlan, “Bagging, boosting, and C4.5,” in Proc. AAAI/IAAI, vol. 1, Aug. 1996, pp. 725–730.

[42] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: An update,” ACM SIGKDD Explor. Newslett., vol. 11, no. 1, pp. 10–18, 2009.

[43] N. B. Karayiannis and P.-I. Pai, “Fuzzy algorithms for learning vector quantization,” IEEE Trans. Neural Netw., vol. 7, no. 5, pp. 1196–1211, Sep. 1996.

[44] A. L. Goldberger, L. A. N. Amaral, L. Glass, M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, “PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals,” Circulation, vol. 101, no. 23, pp. e215–e220, Jun. 2000, doi: 10.1161/101.CIR.101.23.e215.

[45] I. Güler and E. D. Ubeiyli, “ECG beat classifier designed by combined neural network model,” Pattern Recognit., vol. 38, no. 2, pp. 199–208, Feb. 2005.

[46] S. Mallat and S. Zhong, “Characterization of signals from multiscale edges,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 14, no. 7, pp. 710–732, Jul. 1992.

[47] Philip de Chazal, Costor Heneghan, Elaine Sheridan, Richard Reilly, Philip Nolan, and Mark O’Malley, “Automated processing of the single-lead electrocardiogram for the detection of obstructive sleep apnea,” IEEE Trans. Biomed. Eng., vol. 50, no. 6, pp. 686–696, Jun. 2003.

[48] The International Classification of Sleep Disorders, Revised: Diagnostic and Coding Manual. American Academy of Sleep Medicine, Chicago, IL, USA, 2001.

[49] J. Pan and W. J. Tompkins, “A real-time QRS detection algorithm,” IEEE Trans. Biomed. Eng., vol. BME-32, no. 3, pp. 230–236, Mar. 1985.

[50] S. Mallat and S. Zhong, “Characterization of signals from multi-resolution analysis,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 14, no. 7, pp. 710–732, Jul. 1992.

[51] B. Yilmas, H. Musa Asyali, E. Arikian, S. Yetkin, and F. Özgen, “Sleep stage and obstructive apneaic epoch classification using single-lead ECG,” Biomed. Eng. Online, vol. 9, pp. 39–53, Aug. 2010.

[52] F. S. Bao, X. Liu, and C. Zhang, “PyEEG: An open source python module for EEG/MEG feature extraction,” Comput. Intell. Neurosci., vol. 2013, Art. no. 406391.

[53] W. Jatmiko, T. Fukuda, F. Arai, and B. Kusumoputro, “Artificial odor discrimination system using multiple quartz resonator sensors and various neural networks for recognizing fragrance mixtures,” IEEE Sensors J., vol. 6, no. 1, pp. 223–233, Feb. 2006.

[54] S. Derhami and A. E. Smith, “HGA: Hybrid genetic algorithm in fuzzy rule-based classification systems for high-dimensional problems,” Appl. Soft Comput., vol. 41, pp. 91–93, Apr. 2016.

[55] J. Ravichandran, M. Kaden, S. Saralajew, and T. Villmann, “Variants of DropConnect in learning vector quantization networks for evaluation of classification stability,” Neurocomputing, vol. 403, pp. 121–132, Aug. 2020.

[56] N. Floropoulos and A. Tefas, “Complete vector quantization of feedforward neural networks,” Neurocomputing, vol. 367, pp. 55–63, Nov. 2019.

[57] I. Hermawan, M. I. Tawakal, J. M. A. Setiawan, I. Habibie, and W. Jatmiko, “Adaptive multi codebook fuzzy neuro generalized learning vector quantization for sleep stages classification,” in Proc. Int. Conf. Adv. Comput. Sci. Inf. Syst. (ICACSIS), Sep. 2013, pp. 431–436.

[58] H. George John and P. Langley, “Estimating continuous distributions in Bayesian classifiers,” in Proc. 11th Conf. Uncertainty Artif. Intell. San Francisco, CA, USA: Morgan Kaufmann, 1995, pp. 338–345.

[59] T. Cover and P. Hart, “Nearest neighbor pattern classification,” IEEE Trans. Inf. Theory, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[60] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” Inst. Cogn. Sci., California Univ., San Diego, La Jolla, CA, USA, 1985.

[61] C. Cortes and V. Vapnik, “Support-vector networks,” Mach. Learn., vol. 20, no. 3, pp. 273–297, 1995.

[62] R. Quinlan, “Bagging, boosting, and C4.5,” in Proc. AAAI/IAAI, vol. 1, Aug. 1996, pp. 725–730.

[63] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: An update,” ACM SIGKDD Explor. Newslett., vol. 11, no. 1, pp. 10–18, 2009.

[64] N. B. Karayiannis and P.-I. Pai, “Fuzzy algorithms for learning vector quantization,” IEEE Trans. Neural Netw., vol. 7, no. 5, pp. 1196–1211, Sep. 1996.
M. IQBAL TAWAKAL received the B.S. and M.Sc. degrees in computer science from Universitas Indonesia and the KTH Royal Institute of Technology, in 2011 and 2016, respectively. He currently works as a Data Scientist at Trav eloaka. His research interests include intelligent systems, neural networks, and data science.

M. FEBRIAN RACHMADI received the B.S. and M.Sc. degrees in computer science from Universitas Indonesia and The University of Edinburg, in 2014 and 2016, respectively, and the Ph.D. degree from The University of Edinburg, in 2019. He currently works as a Researcher at RIKEN Japan. His research interests include neural networks, deep learning, and brain imaging.

M. ANWAR MA’SUM (Member, IEEE) received the B.S. and M.Sc. degrees in computer science from Universitas Indonesia, in 2013 and 2014, respectively. He currently works as a Lecturer and Researcher with the Faculty of Computer Science, Universitas Indonesia. His research interests include intelligent systems and machine learning.

HANIF A. WISESA received the B.S. and M.Sc. degrees in computer science from Universitas Indonesia and The University of Melbourne, in 2014 and 2017, respectively. He currently works as a Business Analyst at Tokopedia. His research interests include intelligent systems, neural networks, and data science.

TOSHI FUKUDA (Life Fellow, IEEE) graduated from Waseda University, Tokyo, Japan, in 1971, and received the M.Eng. and D.Eng. degrees from the University of Tokyo, in 1973 and 1977, respectively. He studied at the Graduate School of Yale University from 1973 to 1975. He joined the National Mechanical Engineering Laboratory, Japan, in 1977, the Science University of Tokyo, in 1982, and then joined the Department of Mechanical Engineering, Nagoya University, Japan, in 1989. He worked at the University of Stuttgart, as an Humboldt Fellow (1979–1981). He is the Professor Emeritus of Nagoya University, and also a Professor of Meijo University, Waseda University, and Beijing Institute of Technology. He has been working as a Professor of the Shenyang University of Technology, Suzhou University, Institute of Automation, Chinese Academy of Science. Russell Springer Chaired Professor at UC Berkeley, Seoul National University, Advisory Professor of Industrial Technological Research Institute in Taiwan, and so on. He is a Foreign Member of the Chinese Academy of Sciences (2017). He is mainly engaged in the research fields of intelligent robotic systems, micro and nano robotics, bio-robotic system, and technical diagnosis and error recovery system. He was the President of the IEEE Robotics and Automation Society (1998–1999), the Director of the IEEE Division X, Systems and Control (2001–2002), the Founding President of the IEEE Nanotechnology Council (2002–2005), the Region 10 Director (2013–2014), the Director of Division X, Systems and Control (2017–2018), and the IEEE President-Elect (2019). He was the Editor-in-Chief of the IEEE/ASME TRANSACTIONS ON MECHATRONICS (2000–2002). He was the Founding General Chair of the IEEE International Conference on Intelligent Robots and Systems (IROS) held in Tokyo (1988). He was the Founding Chair of the IEEE Conference on Nanotechnology (2001), the IEEE Workshop on Robot and Human Communication (1994), the IEEE Workshop on Advanced Robotics Technology and Social Impacts (ARSO, 2005), the Founding Chair of the IEEE Workshop on System Integration International (SII, 2008), the Founding Chair of the International Symposium on Micro-Nano Mechatronics and Human Science (MHS, 1990–2012), the IEEE Conference on Cyborg and Bionic Systems (2017), and the IEEE Conference on Intelligence and Safety of Robots (2018).