Received: October 30, 2019. Revised: January 6, 2020.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Cosine K-Nearest Neighbor in Milkfish Eye Classification

Eko Prasetyo[1]* Rani Purbaningtyas[1] Raden Dimas Adityo[1]

[1]*Department of Informatics, Engineering Faculty,*
University of Bhayangkara Surabaya, Surabaya, Indonesia
* Corresponding author's Email: eko@ubhara.ac.id

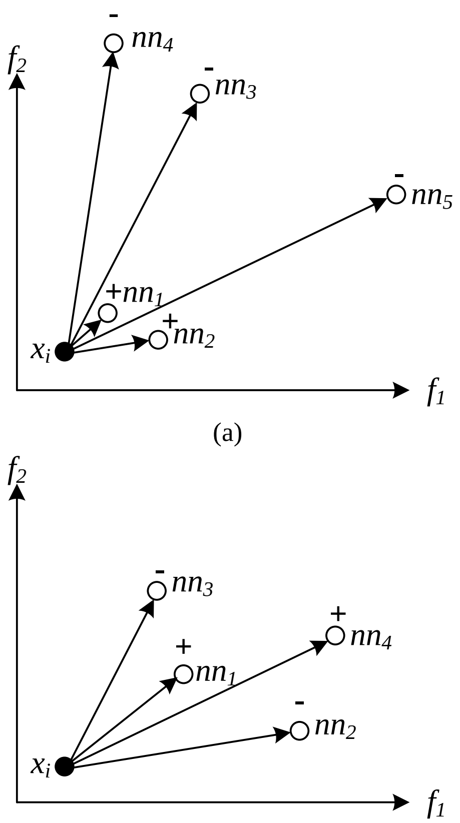
Abstract: K-Nearest Neighbors (K-NN) classification method gains refined version proposed by the researcher. The refinement aims to solve noise sensitive when using small K, and irrelevant class as classification result when using large K. The problem in the previous version of method was that the weights were calculated individually, so the result was not optimal. We propose recent weighting scheme where the weights were no longer gained from the nearest neighbor individually, but by involving all pair of the nearest neighbor, called Cosine K-NN (CosKNN). We also introduce a trigonometric map to describe the Cosine weight. CosKNN is soft value to represent ownership of each class to the testing data. Empirically, CosKNN is tested and compared with other K-NN refinement using milkfish eye, UCI, and KEEL dataset. The result shows that CosKNN hold superior performance compared to the other methods although K number is higher of which accuracy is 96.79%.

Keywords: K-nearest neighbor, Weight, Cosine, Refinement, Milkfish eye.

1. Introduction

Since the K-Nearest Neighbor (KNN) classification method was introduced in research[1], this method obtains a lot of research attention including the recent research. As in [2], conducting selection of nearest neighbors on both testing sample and training sample, called general nearest neighbor (GNN) rule that also uses the overlapping neighborhood to determine result in general nearest neighbor. The research by [3] determine the label of testing set with combination of similarity and projection angle between testing set and selected nearest neighbor. The scheme is called dependent nearest neighbor (dNN). The other research by [4] attempt to redefine the Minkowski distance metric in order to consider the relevant features only by the assumption that all features have the same importance level in classification stage. Such a scheme is designed to solve problem in high-dimensionality dataset. Furthermore, recent research by[5] conducts enhancement of KNN by calculating generalized mean distance-based k-nearest neighbor classifier (GMDKNN) using k local mean vector per class. The result of the classification is obtained by the minimum nested generalized mean distance among all the classes. Generally, the main objective in many proposed KNN refinement is due to the fact that KNN is noise sensitive when using small K in nearest neighbors selection [6]. Noise is data which contain different behavior from rest of general data. The presence of noise in the training set may cause the result of the classification to fall on the noise data, one nearest neighbor of the test data. Another weakness of KNN is when using larger K; it can cause the result of the classifications to fall on other classes that are actually irrelevant because the distance is too far from the testing data [7], as presented in Fig. 1 (a). Hence, the selection of K constitutes a highly critical problem. Generally, empirical testing is conducted to obtain the appropriate K and the best performance, as conducted by [5, 8, 9].

The initial problem of noise-sensitive occurs when determining the prediction result in KNN using class majority voting from selected K nearest neighbor, as presented in Fig. 1 (a). When using 1-NN, the testing data (black dot) would be classified to one nearest neighbor class A (+). When using 5-NN, the testing data would be classified to the most



(a)

(b)

Figure. 1 Problem result in classic K-NN: (a) 5-NN with same majority voting and (b) 5-NN with irrelevant class as classification result

dominant class in the nearest neighbor data; for example, 2 nearest neighbors are from class A(+) and 3 nearest neighbors are from class B(-) because class B has more data than class A, so the testing data are classified to class B. This scheme would inflict a misclassification when the 3 nearest neighbors are too far from the testing data. This is due to the 3 nearest neighbors are farther than the 2 neighbors in which the 3 neighbors are actually irrelevant. The other problems occur when using an even number of nearest neighbor and majority voting fall to two same majority classes, as presented in Fig. 1 (b); for example, using 4-NN, 2 nearest neighbors are from class A and B, respectively; one of which can be chosen, A or B arbitrarily [7]. The other approach to solve the problem is to use a weighting scheme [9-11]. The weighting approach is conducted by calculating the weight for each nearest neighbor based on distance between testing data to each nearest neighbor. In essence, the closer the nearest neighbor, the greater the weight gained. In the research by [12], the weight calculation is conducted using the inverse square of its distance from testing data, while the research by [11] use dual weighted voting k-nearest neighbor rule (DWKNN). The DWKNN depends on the distance between the testing data and nearest neighbors, and also the rank of the nearest neighbors, in which the closer neighbor would gain greater weight. These two weighting schemes are combined in the research by [9] where the weight calculation is conducted using the weight average of each class in order to enhance the weighted K-NN [11,12]. Research by [2] develop K-NN using two-sided environmental information of testing data and K nearest neighbor training data, called General Nearest Neighbor (GNN). The mutual neighbor generated between the two data becomes the chosen neighbor which will be chosen in the most voted class. Unfortunately, this method can trap the system in the same number of votes, as in the previous second problem. The recent KNN research indicates that there is a modification of k-nearest neighbor (MKNN) proposed by [8] which applies the smallest modified KNN (SMKNN) and the largest modified KNN (LMKNN). The strategy aims to calculate the centroid of each class then calculate the distance and weight of each data in the middle of the class. Moreover, the distance and weight of the testing data are calculated for each class center and all training data. SMKNN will take all neighbors in which the distance to neighbors is smaller than the distance to the nearest centroid. The final weight is obtained by multiplying the weight of the testing data to the selected neighbor and the initial weight of the selected neighbor. The accumulation of selected neighbor weight to the respective class will produce the final weight of the class. The class of the highest weight is the prediction class. LMKNN uses the same strategy but the distance to neighbor is greater than the distance to the nearest centroid. This method provides the same performance effect on all K choices used. However, the problem in many KNN refinements using weighting scheme merely involves the testing data and the nearest neighbor. So, the weight gained is actually similar to the inverse of the distance. In this study, we propose a weighting scheme of which weight is not gained from the nearest neighbor individually, but by calculating the weight with the involvement of a pair of nearest neighbors. The weight calculation of a pair of nearest neighbors would produce two weights, each of which corresponds to the neighbor in pair. The weight of each data in pair represents ownership of each class to the testing data according to the class of data. These weights are ultimately accumulated to obtain the final weight of each class. Furthermore, the prediction results are obtained from the largest weight accumulation.

We conduct performance comparison between CosKNN and other KNN refinements including recent K-NN as follows: classic KNN [1], Weight K-NN (WKNN) [12], dual weighted voting method for KNN rule (DWKNN) [11], combination of local mean based and distance weight k-nearest neighbor (LDWKNN) [9], General Nearest Neighbor (GNN) [2], and also smallest modified KNN (SMKNN) and largest modified KNN (LMKNN) [8], which are applied to milkfish eye dataset, UCI Machine Learning datasets, and KEEL-dataset Repository. Empirical testing is conducted using a variety of K number starting from 3 to 35 nearest neighbors. We also make comparison with other classification methods, namely Support Vector Machine [13] and Decision Tree [7].

Using weighting scheme involving a pair of nearest neighbors between the testing data and two neighbors. It is possible that the distance of one neighbor to the testing data is closer than the other neighbor to the testing data, or vice versa. The weighting concept we propose is when calculating the weight between a neighbor to the testing data does not only generate the information between the neighbor with the testing data, but also the information of the distance of other neighbors to the testing data. The use of the cosine concept in weight calculation presents a new perspective that weights are calculated using a scheme involving two simultaneous parties using distance as the length of a right triangle. Thus, the weighting system contribute to the performance accuracy being more optimal in solving noise sensitive, same majority votes class problem and irrelevant class as the prediction result. We provide in-depth description in the following section. From the performance comparison, it shows that CosKNN provides a promising performance in classification problem which gain the highest accuracy performance.

2. Literature review

2.1 K-nearest neighbor

K-Nearest Neighbor (KNN) is a classification method based on nearest neighbor with a simple concept, robust on non-linear data, and can be used in multi-class cases. The concept is that if an animal walks like a duck, sounds like a duck, and behaves like a duck then the animal may be a duck [7]. It was firstly introduced in the research by [1] using the nearest neighbor concept based on distance. The nearer the neighbor to the testing data, the neighbor will be more similar. From all the training data, the K nearest neighbor will be chosen to do majority class voting. The most voted class will be used as the prediction class.

Given $(x_i,c_i),i=1,...,N$ is a pair of training data and class in which x is the training datum from a set of training data $X=\{x_i/x_i \in S^M\}$, in which N is the total training data, c is one class from a set of class $C=\{c_1,c_2,...,c_P,\}$. S is the dataset of which each data has M feature. The distance between testing data x' with training data is calculated using Eq. (1)

$$d_i=\|x_i,x'\|=\sqrt{\sum x_i-x'} \tag{1}$$

Eq. (1) uses Euclidean as the basis for distance assumption. Calculating the prediction of the most voted class uses Eq. (2)

$$c' = \underset{c \in C}{\arg\max} \sum_{i=1}^{K} \delta(c,c_i) \tag{2}$$

In which $\delta(c,c_i)$ is1 if $c=c_i$ and is 0 if $c \neq c_i$.

2.2 Weight KNN (WKNN)

One of the KNN refinements using the weighting technique is Weight KNN (WKNN) of which weight is calculated using inverse distance [12]. Furthermore, the prediction result generated comprises the accumulation of the neighbor weight according to the neighbor class. The weighting using inverse distance is shown in Eq. (3).

$$w_i=1-d_i \tag{3}$$

Inverse distance will reverse the equation meaning that the training data of short distance will gain great weight, and data of long distance will gain small weight. The accumulation of neighbor weights according to the class will be used to determine the class prediction, the class of the highest weight becomes the class prediction. Weighting using this method is insignificant, since it only involves the testing data with one of the nearest neighbors selected. We propose weighting which involves a pair of distance between the testing data with the two selected neighbors. Involving more than two distances of training data simultaneously means that the generated weight is more accurate and significant.

2.3 Dual weighted voting method for KNN rule (DWKNN)

Dual weighted voting method for KNN rule (DWKNN) sets weighting rule based on weight accumulation and weight function starting from the greatest weighting order[11]. Besides, weight calculation also uses the farthest distance between the training data to testing data. The farther the training data to the testing data, the smaller the weight of the training data. Weight calculation for each training data uses Eq. (4)

$$w_i = \begin{cases} \frac{d_K^{nn}-d_i^{nn}}{d_K^{nn}-d_1^{nn}} x \frac{1}{i} & , d_K^{nn} \neq d_1^{nn} \\ 1 & , d_K^{nn} = d_1^{nn} \end{cases} \quad (4)$$

2.4 Combination of local mean based and distance weight k-nearest neighbor (LDWKNN)

Combination of local mean based and distance weight k-nearest neighbor (LDWKNN) [9] combines the WKNN weighting method and DWKNN using the average weight from each class. The class of the largest average weight will be chosen as a prediction result. This method highly depends on the inverse calculation of the distance between the testing data and the training data, and is claimed to provide better result than the previous method.

Even though it involves distance of other neighbors, the weighting method, such as LDWKNN or DWKNN has the disadvantage that the ranking can potentially degrade distances on numerous data of very small differences to very large, or two distances with large differences in sequential ranks can result in smaller weight degradation. We propose a pair of two distances from the chosen K nearest neighbor. Given 5 nearest neighbors, the generated pair $\frac{K!}{(K-2)!2!}$ will be 20 pairs. Hence, we disregard the ranking but at the same time involve the distance of two neighbors.

2.5 General nearest neighbor (GNN)

Research by [2] develops K-NN using two-hand side neighborhood information of testing data and K nearest neighbor training data, called General Nearest Neighbor (GNN). The mutual neighbor generated between the two data becomes the chosen neighbor which will be chosen in the most voted class. Mutual neighbor will be selected on the data which is the K-nearest neighbor, both in the testing data and in each data of K-nearest neighbor. This algorithm starts by combining testing data into training set followed by looking for the K-nearest neighbor of each data. For each data in K-nearest neighbor of the testing data, it is necessary to check whether any of the data is a mutual neighbor; if so, it will be classified as general neighbor. From the list of selected general neighbor, class majority will be voted. The most voted class will be the prediction class.

This method basically provides a good idea by choosing mutual neighbor, but there is not any weighting on the selected neighbor as mutual neighbor. Likewise, determining the prediction also uses voting technique, such as the classic KNN. Class voting with conventional technique can still cause the same majority class voting which should be avoided.

2.6 Modified KNN (MKNN)

Modified KNN (MKNN) also provides a new method of classifying based on nearest neighbors, in which this method also involves the centroid role of each class [8]. There are two weights calculated; the weight calculated from the distance of the training data to the centroid of the class and the weight calculated from the distance between the selected training data and the testing data. The MKNN algorithm is as follows.

1. Calculate centroid from each class of data using Eq. (5)

$$Cen_r = \frac{1}{t}\sum_{i=1}^{N} x_i \mid c_i = c_r \quad (5)$$

   In which *t* is the number of training data in class c_r;
2. Calculate distance between training data with class centroid using Eq. (6);

$$d(x_i, Cen_r) = \sqrt{\sum x_i\text{-}Cen_r} \quad (6)$$

3. Calculate weight of training data using inverse distance as in Eq. (7);

$$w(x_i, c_r) = \frac{1}{d(x_i,c_r)} \quad (7)$$

4. Calculate distance between testing data and respective centroid;
5. Select the shortest centroid if using smallest modified KNN (SMKNN) as D_x or the farthest distance if using largest modified KNN (LMKNN) as D_x;
6. Select K neighbor in radius D_x
7. Calculate weight of each neighbor generated in step 6 using Eq. (8)

$$IS(x', x_r) = \frac{1}{d(x_i,x_{rx})} \times w(x_i, c_r) \quad (8)$$

In which $d(x_i, x_{rx})$ is the distance between testing data and neighbors generated in radius D_x.

8. Accumulate weight $IS(x', x_r)$ based on respective class then select class with highest weight accumulation as prediction class.

The method proposed in MKNN has a non-parametric effect in which the use of different K does not affect the result of the selected neighbors in radius D_x. As a result, if this method does not achieve optimal result it means that there is not any other choice that can be made. In contrast to the parametric method, there are many K options that can be observed to determine the best K to use.

3. Proposed method

This paper has been partially published in conference paper in [14], but we attempt to provide more explanation about the fundamental concept and comprehensive comparison with a number of image features and recent KNN refinement, as presented in the sub sections below.

Our proposed method is a recent weighting system involving two simultaneous parties. Both parties are testing data and 2 nearest neighbors selected. Simultaneous weight calculation is conducted using a trigonometric map. Conceptually, we will use the distance of two neighbors d_1 and d_2 as the adjacent and the opposite side of right triangle. For hypotenuse side, it is calculated using d_1 and d_2 in perpendicular position. From the concept, cosine is calculated between two sides of perpendicular.

The illustration of weight calculation is presented in Fig. 2. Fig. 2 is the fundamental concept in our study. We use a trigonometric function [15] with a right triangle as the basic map. The trigonometric map is a right triangle constructed from a pair distance of nearest neighbor as the adjacent and the opposite side. In other words, we also use a hypotenuse as the third side. It is called trigonometric map. It is exemplified in Fig. 2 (a); we obtain two nearest neighbors nn_1 and nn_2 with distance of d_1 and d_2, respectively. The nn_1 is from class 1 (+) and nn_2 is from class 2 (-). For the pair, we obtain two maps according to Figs. 2 (b) and (c) for nn_1 and nn_2, respectively. For each map, we calculate a soft value which represents ownership of each class to the testing data. The soft value is called Cosine KNN (CosKNN). The CosKNN represents ownership of each class to the testing data which range from 0 to 1.

According to Fig. 2 (b), we can calculate the Cosine value of α_1 for the nn_1 when paired with nn_2 using Eq. (9);

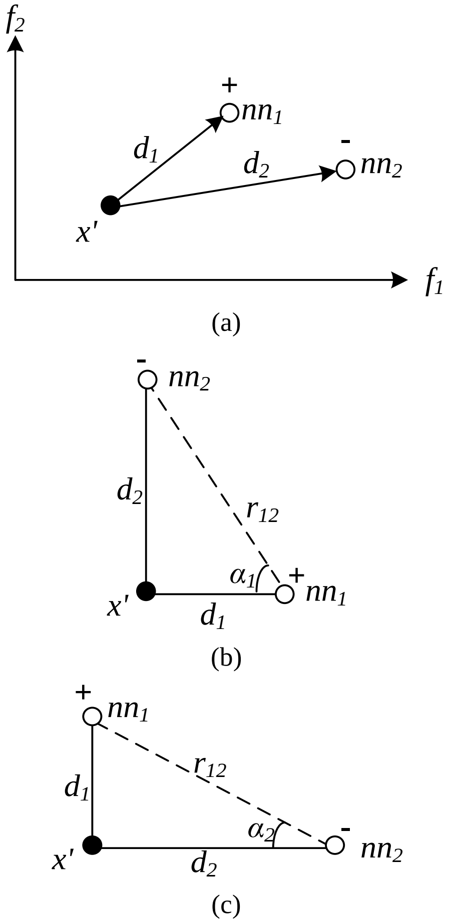


Figure. 2 Trinometric map: (a) distance map, (b) trigonometric map of nn_1, and (c) trigonometric map of nn_2

$$Cos(\alpha_1) = \frac{d_1}{r_{12}} \tag{9}$$

The $Cos(\alpha_1)$ has a range of value between 0 and 1. When distance d_1 is small then $Cos(\alpha_1)$ is close to 0, but represents a significantly high ownership of class 1(+) from nn_1 when paired with nn_2. It is required that the nearest neighbor with lower distance obtain greater weight, so Eq. (9) is adjusted in order to obtain the $Cos(x_1, x_2)$ weight as follows.

$$Cos(x_1, x_2) = 1 - Cos(\alpha_1) = 1 - \frac{d_1}{r_{12}} \tag{10}$$

$Cos(x_1, x_2)$ is cosine weight for nn_1 (x_1) when paired with nn_2 (x_2). We conduct complementary operation to the cosine value from the map, so the lower distance would obtain greater weight which represents higher ownership of the related class, and vice versa.

Given the data set class is $C = c_1, c_2, \ldots, c_n$, where n is the number of classes. While $X = x_1, x_2, x_i, \ldots, x_K$ is the nearest neighbor selected from the training data, K is the number of nearest neighbors. For a pair of x_i and x_j as the part of selected nearest neighbor in CosKNN, the general equation is formulated as follows.

$$Cos(x_i, x_j) = 1 - \frac{d_i}{r_{ij}} = 1 - \frac{d_i}{\sqrt{d_i + d_j}} \tag{11}$$

$$Cos(x_j, x_i) = 1 - \frac{d_j}{r_{ij}} = 1 - \frac{d_j}{\sqrt{d_i + d_j}} \tag{12}$$

Where d is the distance of training data to the testing data. $Cos(x_i, x_j)$ and $Cos(x_j, x_i)$ are cosine weights obtained from the pair of nearest neighbors nn_1 and nn_2, respectively. $Cos(x_i, x_j)$ represents the ownership degree of related class from x_i when paired with x_j, while $Cos(x_j, x_i)$ represents the ownership degree of related class from x_j when paired with x_i.

This cosine weight concept of nearest neighbor pair is applied to all selected K-nearest neighbors. For K-nearest neighbors, $C(K, 2)$ nearest neighbor pair. $C(K, 2)$ is the combination of K number with 2 elements, for example the 3-NN is neighbor data with the following data and class; (x_1,c_1), (x_2,c_2), and (x_3,c_1), then the pairs obtained are: x_1x_2, x_1x_3, x_2x_3. Furthermore, each pair will obtain 2 cosine weights according to (11) and (12). Each cosine weight would be classified to the related class of the data. For example, 3-NN would obtain the cosine weight as follows.

$Cos(x_1, x_2)$ of nn_1 would be owned by class c_1
$Cos(x_2, x_1)$ of nn_2 would be owned by class c_2
$Cos(x_1, x_3)$ of nn_1 would be owned by class c_1
$Cos(x_3, x_1)$ of nn_3 would be owned by class c_1
$Cos(x_2, x_3)$ of nn_2 would be owned by class c_2
$Cos(x_3, x_2)$ of nn_3 would be owned by class c_1

It is known that Cosine weighting technique with K-nearest neighbors can generate as many as $\frac{K!}{(K-2)!2!}$ pairs. Given that there are 5 nearest neighbors, it means that the result will be $\frac{5!}{(5-2)!2!}$ pairs or 20 pairs. To achieve the classification result, it is required to sum up all cosine weight to the related class of each data. For example, to sum up all cosine weight of the R testing data (R, c_R) to the class c_r , the equation used is Eq. (13)

$$SumofCos(R, c_r) = \sum_{i=1}^{K} Cos(x_i, x_j)\,|c_i = c_r \tag{13}$$

The $SumofCos(R, c_r)$ is the sum up weight of the R data to the c_r class. It is a soft value which ranges in [0,∞]. A value of zero (0) means that none of the nearest neighbor has a class c_r. The weight can be higher when the number of nearest neighbors increases. If a class does not contain any data in the nearest neighbor, it will obtain zero weight. The more the nearest neighbors are used, the higher the weight achieved by a class.

To obtain the class decision, the highest cosine weight accumulation among all class in the dataset is selected using Eq. (14)

$$c_R = arg\,max(SumofCos(R, c_r)),\quad r = 1, \dots, P \tag{14}$$

In which this equation will generate the highest value of $SumofCos(R, c_r)$ among class P as prediction result of c_R. The class with the highest value is given from this equation as a predicted result.

The algorithm of CosKNN is as follows.

```
Input:
    $R$                          : the testing data
    [($x_i$,$c_i$), $i$=1,…,$N$]  : the training data
    $K$                          : The number of
                                   nearest neighbors
Step 1 : Calculate the distance $d_i$ between the testing data and the training data as $D$
    e.g. using Euclidean distance;
    for i=1 to $N$
        $d_i = \|R, x_i\|_2$
    end
Step 2 : Sort all distance in $D$ then select $K$ lowest distance as $D_K$
    $D_K \subseteq D$
Step 3 : Calculate the cosine weight for all pair selected nearest neighbor in $D_K$ using (11) and (12);
Step 4 : Calculate the summarize weight of each class using (13);
Step 5 : Select the class of the highest weight obtained in Step 4 as the class label for the R testing data.
```

The objectives in using cosine weight in our study are explained as follows.

1. Solving the same majority votes class problem
   Same majority votes class possibly occur when using even number in determining the K nearest neighbor, and when most neighboring classes are classified into two classes. In the CosKNN, such an issue is resolved by weighting cosine scheme for each pair of K-nearest neighbor. The weight accumulation for each class is obtained, so the class with the highest *SumofCos* as the classification result is selected. In this study,

we avoid the same majority votes class problem through good solution.

2. Robust to the problem of classification result to irrelevant class

   This problem possibly occurs especially when using high number as K-nearest neighbor, as exemplified in Fig. 1 (a), we use 5-NN, the nn_1 and nn_2 from class A (+) are significantly close to the testing data, while the nn_3, nn_4, and nn_5 from class B (-) are significantly far from the testing data. According to the K-NN classic method, we take class B as the classification result, since class B is the majority class. Such a case is solved by a weighting scheme, in which the closer the neighbors to the testing data, the greater the weight obtained.

4. Result and Discussions

4.1 Dataset

In this study, the main dataset is in the form of features generated from 71 segmented milkfish eye images used by [14]. The dimension of the dataset is 71 data by 18 features. The features consist of mean and standard deviation generated from 3 color space image, namely red, green, blue from RGB color space, hue, saturation, intensity from HSV color space, and L, a, b from Lab color space. Each color space component generates mean and standard deviation, hence the 18 features. The dataset is used to conduct performance analysis empirically with other recent K-NN refinement by other researcher followed by comprehensive analysis.

We also compare CosKNN performance with some recent K-NN refinements and other classification methods using a number of datasets from UCI Machine Learning (https://archive.ics.uci.edu) [16] and KEEL-datasets Repository (https://sci2s.ugr.es/keel/) [17], [18]. The datasets from UCI are as follows; Iris, Vertebral, Diabetic, Wine, Blood, Audit data, and Divorce. And the datasets from KEEL are as follows; Balance, Banana, Phoneme, Yeast, Ring, and Zoo. The summary of the dataset is presented in Table 1.

Table 1. The summary of datasets

Repo.	**Name**	**Inst.**	**Features**	**Num. of Classes**
Prasetyo *et al*[14]	Milkfish eye	71	18	4
UCI	Iris	150	4	3
UCI	Vertebral Column	310	6	2
UCI	Diabetic Retinopathy	1151	20	2
UCI	Wine	178	13	2
UCI	Blood	748	4	2
UCI	Audit data	777	17	2
UCI	Divorce	170	54	2
UCI	Seeds	210	7	3
KEEL	Balance	625	4	3
KEEL	Banana	5300	2	2
KEEL	Phoneme	5405	5	2
KEEL	Yeast	1484	8	10
KEEL	Ring	7400	20	2
KEEL	Zoo	101	16	7

Table 2. Confusion matrix

		Predicted class	
		Positive (+)	Negative (-)
Actual class	Positive (+)	TP	FN
	Negative (-)	FP	TN

We conduct comparison between CosKNN with other K-NN refinement including recent K-NN, namely classic KNN [1], Weight K-NN (WKNN) [12], dual weighted voting method for KNN rule (DWKNN) [11], combination of local mean based and distance weight k-nearest neighbor (LDWKNN) [9], General Nearest Neighbor (GNN) [2], and also smallest modified KNN (SMKNN) and largest modified KNN (LMKNN) [8]. Performance analysis is conducted empirically with some selected K, namely 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, and 35. We use K starting from low K until high K to solve the misclassification into other class that is actually irrelevant because the distance is significantly far from the testing data.

The testing method used is K-fold cross validation, where in this test we use K = 2, it means 50% instances as training set and 50% instances as testing set. For performance comparison, we use accuracy, where accuracy is the proportion of test data that is correctly predicted divided by all test data. The accuracy calculation is obtained from the confusion matrix as presented in Table 2.

Where TP is the proportion of positive data that is predicted to be positive, FP is the proportion of negative data that is wrongly predicted positive, FT is the proportion of positive data that is wrongly predicted negative, and TN is negative data that is predicted to be negative. This case is for two classes classification, so accuracy is obtained as in Eq. (15).

$$Accuracy = \frac{TP+TN}{TP+FN+FP+TN} \quad (15)$$

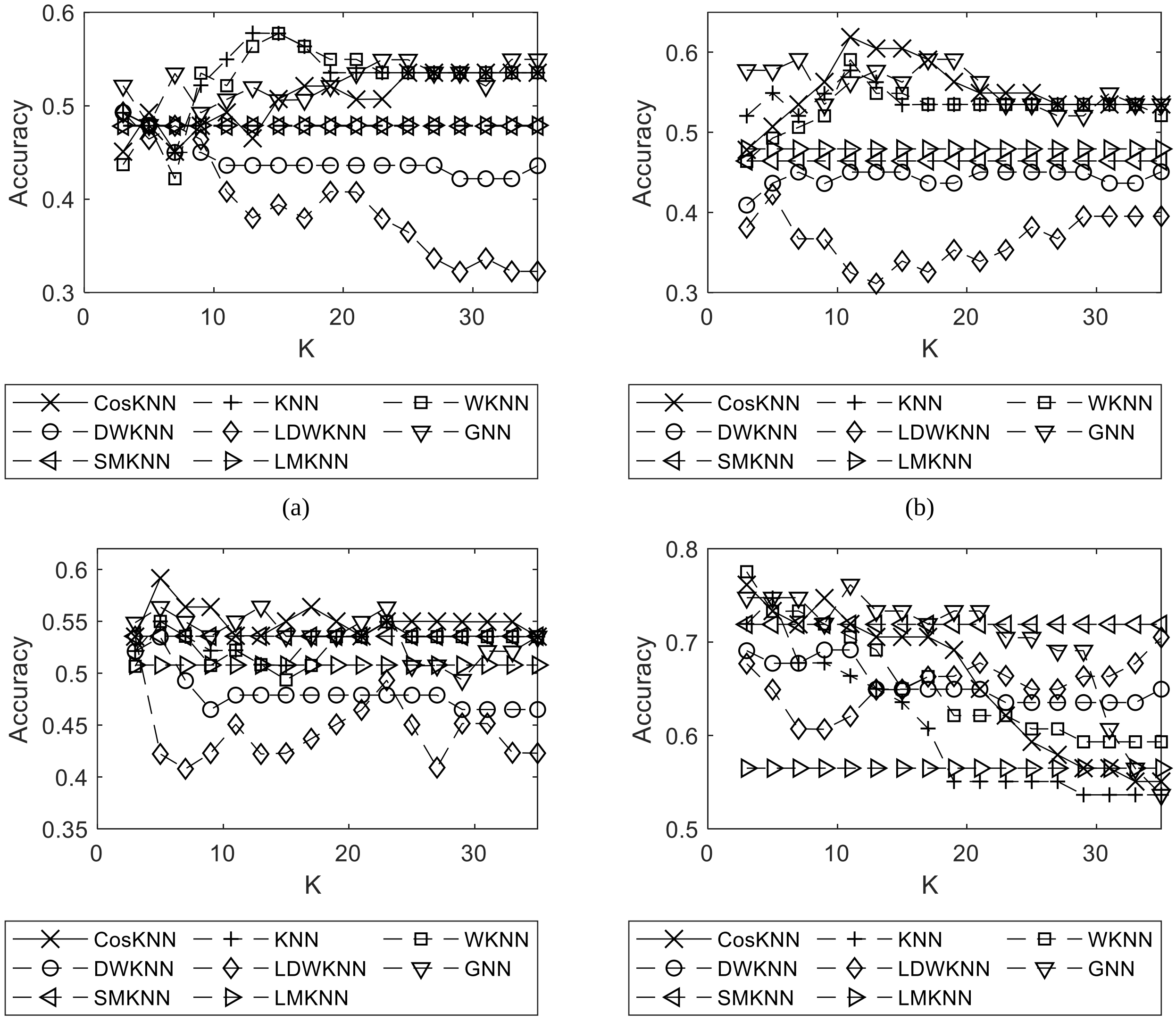
We can also represent accuracy in percentage by multiplying by 100%.

4.2 Performance analysis of CosKNN in milkfish eye dataset

Especially for milkfish eye dataset, we conduct empirical test with a number of color image components, such as red, green, blue from RGB color space, hue, saturation, intensity from HSV color space, and L, a*, b* from Lab color space. The researchers also use 2-fold cross validation of which proportional is 50:50 for training set and testing set, respectively.

The comparison result is presented in Fig. 3. It can be seen in Fig. 3 that CosKNN consistently has high accuracy compared to other methods. Though in certain K, CosKNN indicates accuracy lower than GNN, as in Figs. 3 (a), (b), (c), (e), (f), (g), and (i); overall, the performance of CosKNN always indicates high accuracy result; expect for Fig. 3 (d) in which the performance of CosKNN remains decreasing as K increases starting from 11.

In a case where all features are used, CosKNN tends to show low performance alongside high number of K. It is considered to be reasonable because eye milkfish dataset has low number; by using 2-fold cross validation, we gain 35 instances as the training set followed by selecting all the training set as the nearest neighbor when using 35-NN. We can avoid such a problem by using larger number of data, thus we use several datasets from UCI Machine Learning as comparison cases. Description of the comparison is presented in the next section.

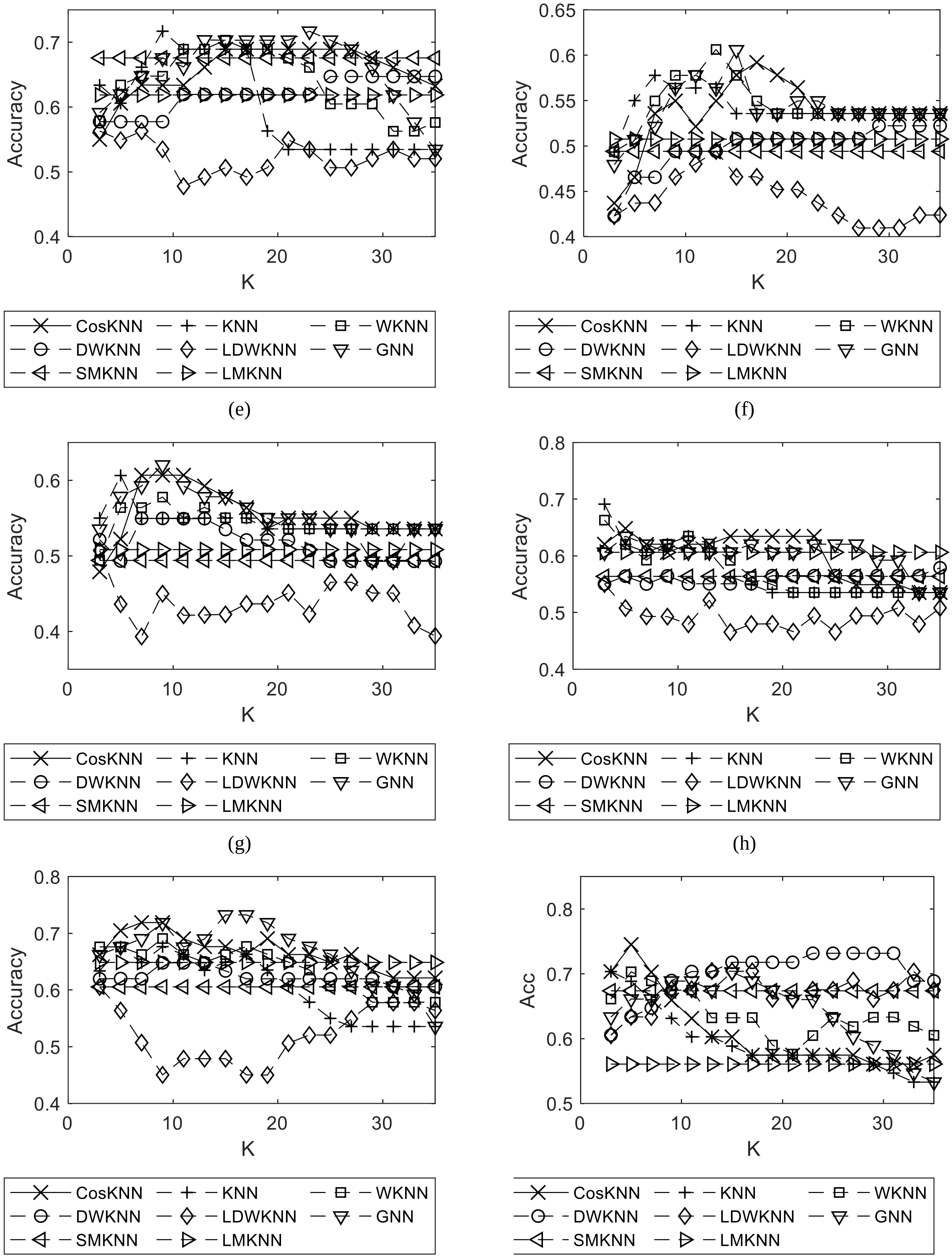


(a)

(b)

(c)

(d)



(e)

(f)

(g)

(h)

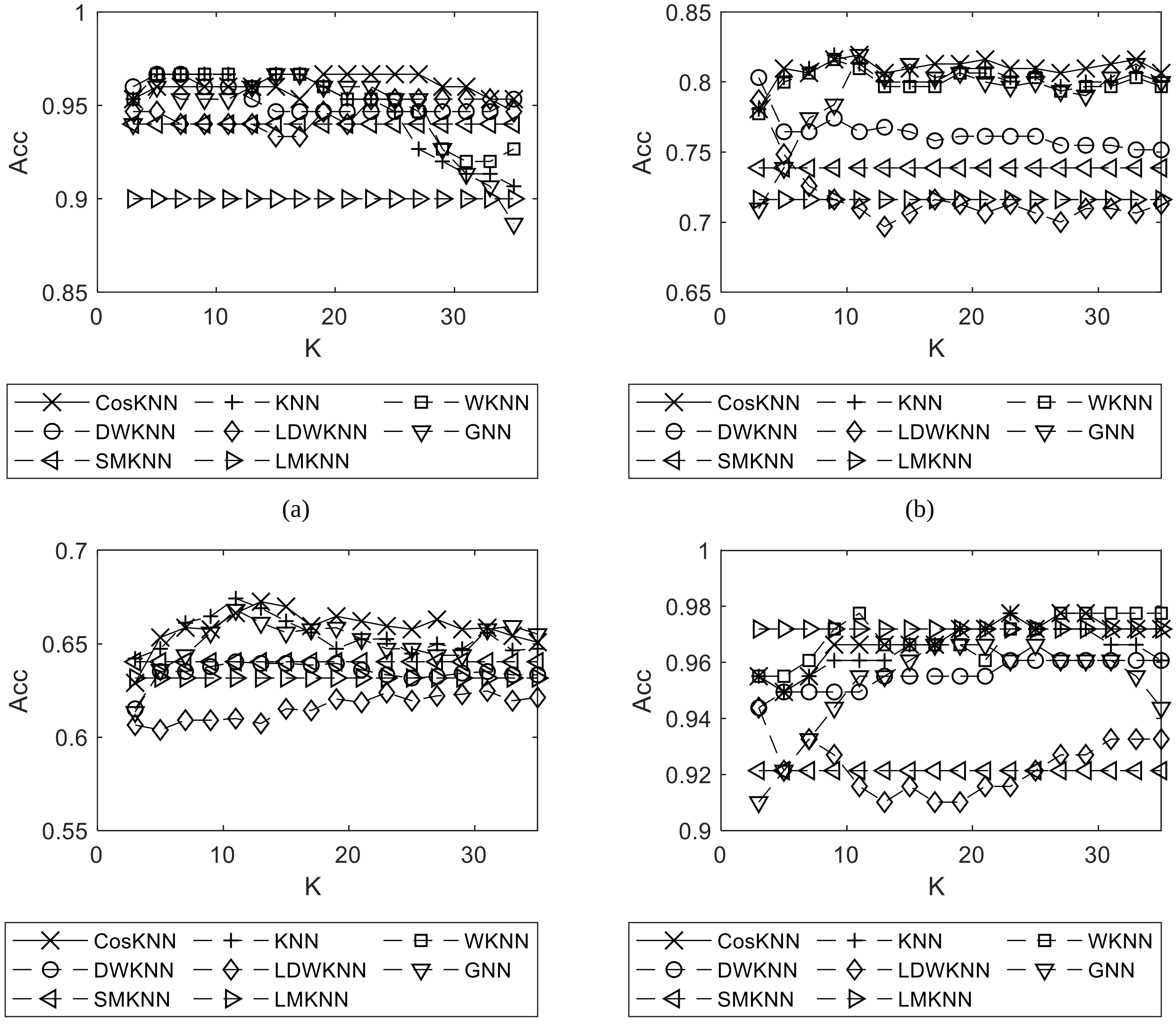
(i)

(j)

Figure. 3 Performance comparison between CosKNN and other K-NN based methods on eye milkfish dataset with variety of color space: (a) red (RGB), (b) green (RGB), (c) blue (RGB), (d) hue (HSV), (e) saturation (HSV), (f) intensity (HSV), (g) l (Lab), (h) a* (Lab), (i) b* (Lab), and (j) all features

Result presented in the eye milkfish dataset uses all features (Fig. 3 (j)); it can be seen that the accuracy of CosKNN is the highest when K is low (given K=5, the accuracy is 0.74), but when K increases, the accuracy of CosKNN decreases and is the same as the LMKNN and KNN method. The same pattern also occurs on KNN but the accuracy is lower compared to CosKNN. In the case of eye milkfish dataset, the method capable to maintain the accuracy alongside increased K is DWKNN, in which when K increases, for example K=21 until 31 the accuracy generated is the highest among other methods. Thus, the DWKNN method relatively has better performance.

When using all K value, however, CosKNN gains a superior performance indicating higher performance than other comparable methods. Of all performance achieved by CosKNN, it is also apparent that CosKNN has good prospect for performance development, since it can generally gain high accuracy. The result can be the solution of irrelevant class problem when using high number of K.

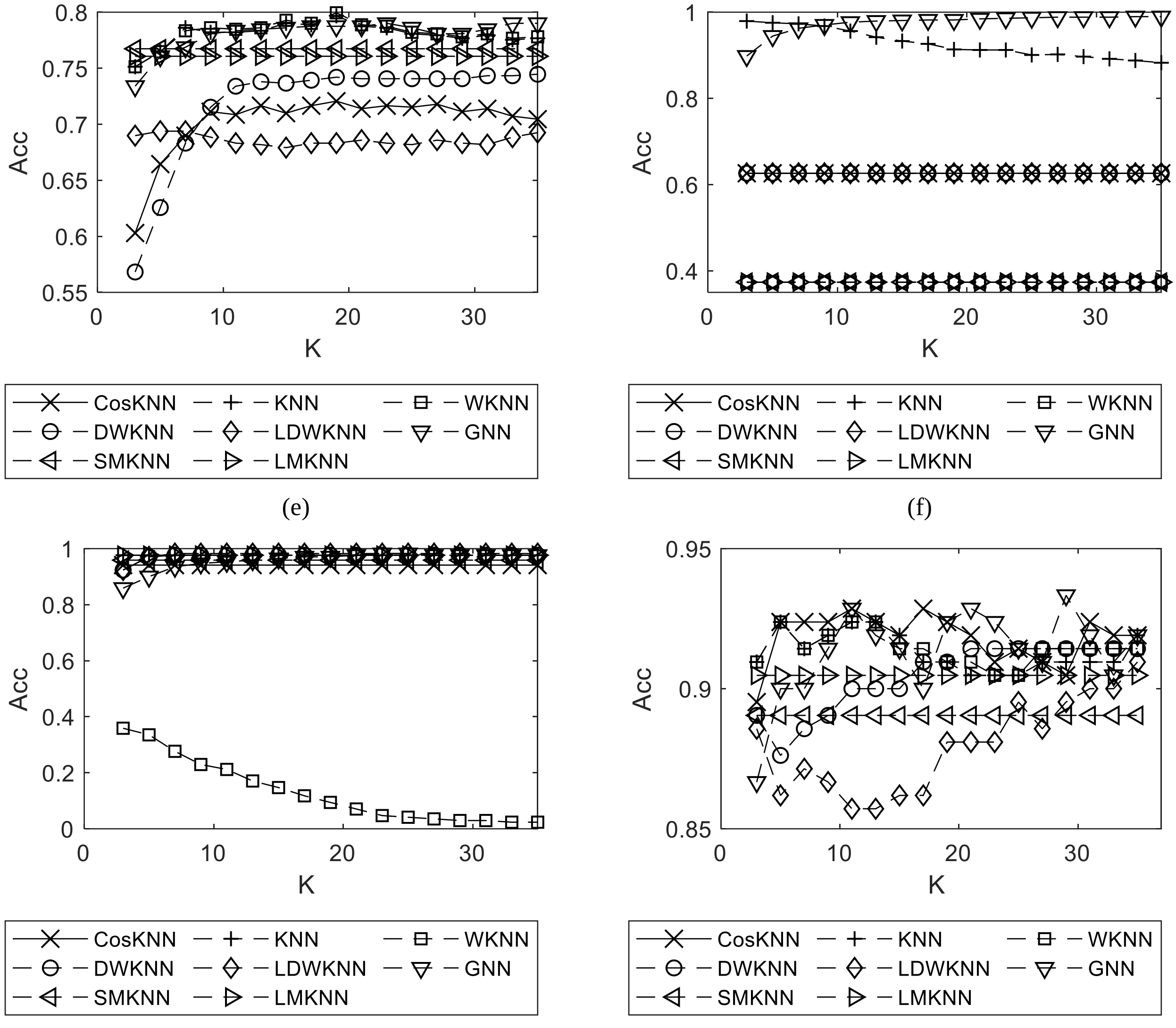


(a)

(b)

(c)

(d)



(e)

(f)

(g)

(h)

Figure. 4 Performance comparison with UCI Machine Learning datasets: (a) iris dataset, (b) vertebral column dataset, (c) diabetic retinopathy dataset, (d) wine dataset, (e) blood dataset, (f) audit data dataset, (g) divorce dataset, and (h) seeds dataset

4.3 Performance analysis with UCI machine learning datasets

UCI Machine Learning Repository is a public dataset managed by University of California, Irvine, School of Information and Computer Sciences. UCI is a collection of data widely used for various purposes from machine learning, theory testing, data generation to empirical studies of science development. Initiated by David Aha in 1987 until recently, there have been 488 datasets and widely used by educators, students, and other researchers around the world, both as primary and secondary data. There are four categories of datasets provided: classification, regression, clustering, others. We use 8 classification datasets from UCI to support our proposed method testing.

In this performance comparison, we use all features of each dataset in the classification. We also conduct performance analysis using a variety number of K starting from 3 to 35. All tests are conducted using 2-fold cross validation. We compare the CosKNN to other K-NN-based methods using some datasets as presented in Table 1, namely, iris, vertebral column, diabetic retinopathy, wine, blood, audit data, divorce, and seeds. In Fig. 4, it can be seen that the dataset in terms of iris, vertebral column, diabetic retinopathy, wine, and seeds, CosKNN has a superior performance compared to the other methods.

The result presented in the iris, vertebral column, diabetic retinopathy, and wine datasets indicate a similar pattern, there are some methods that can maintain performance when K increases from low to high including CosKNN, as well as KNN and GNN. Meanwhile, other method, such as LMKNN more often than not belongs to the method of low accuracy, it achieves low accuracy in the vertebral column, diabetic retinopathy, and wine datasets. Moreover, in the iris and wine dataset, all methods achieve significantly high performance of which accuracy is above 0.9.

In contrast to the result of the blood dataset, the performance accuracy of CosKNN along with DWKNN, LDWKNN is below other methods although reaching an average of 0.7024, 0.7186, and 0.6858, respectively; whereas, other methods tend to achieve better accuracy. The result achieved by all methods basically remains acceptable.

Different result can also be seen in the data audit dataset in which only two methods are able to present significant result, namely KNN and GNN. Meanwhile, other methods achieve low accuracy below 0.63. It includes CosKNN which achieve performance accuracy below 0.63 along with the majority of other methods. The divorce and seeds datasets also indicate different result from the previous result in which all the methods compared provide good performance. All achieved accuracy performance are above 0.8, except WKNN in divorce dataset of which accuracy is below 0.4 and continue to decrease with increased K until accuracy is close to zero.

All in all, CosKNN shows good performance on 7 out of 8 datasets tested. It means that CosKNN is feasible to use as a method to solve classification issue. Also, CosKNN can generate high performance when using higher number of K, so CosKNN can solve irrelevant class problems as a result of the prediction, but the accuracy of CosKNN is relatively similar to the other current methods.

4.4 Performance analysis with KEEL-dataset repository

KEEL-dataset repository is a portal which provides software and a collection of datasets to conduct data collection in various fields. KEEL is an open source (GPLv3) Java software for GUI-based data mining with a combination of computational intelligence. A number of data mining which can be done include classification, clustering, regression, and association. At present, 908 datasets have been collected that can be used for research by both educators, students, and other researchers.

We used six classification datasets for performance comparison between our proposed method CosKNN and other KNN-based methods. The datasets are as follows; balance, banana, phoneme, yeast, ring and zoo. The result of performance comparison on all datasets is presented in Fig. 5. From Fig. 5, it can be seen that generally there are methods which achieve high accuracy, but there are also methods which achieve low accuracy. Given the balance, banana, yeast, and zoo datasets, of the 4 datasets CosKNN is classified as high accuracy alongside GNN and KNN; on zoo dataset, however, CosKNN indicates lower accuracy than other methods. Other methods indicate high accuracy on one dataset, but low accuracy on the others. For example, SMKNN and LMKNN indicate high accuracy on balance and zoo datasets, but lower accuracy on banana and yeast dataset. Phoneme and ring datasets are different cases, which are divided into three: high, medium, and low accuracy, yet CosKNN fails to achieve high accuracy on both datasets.

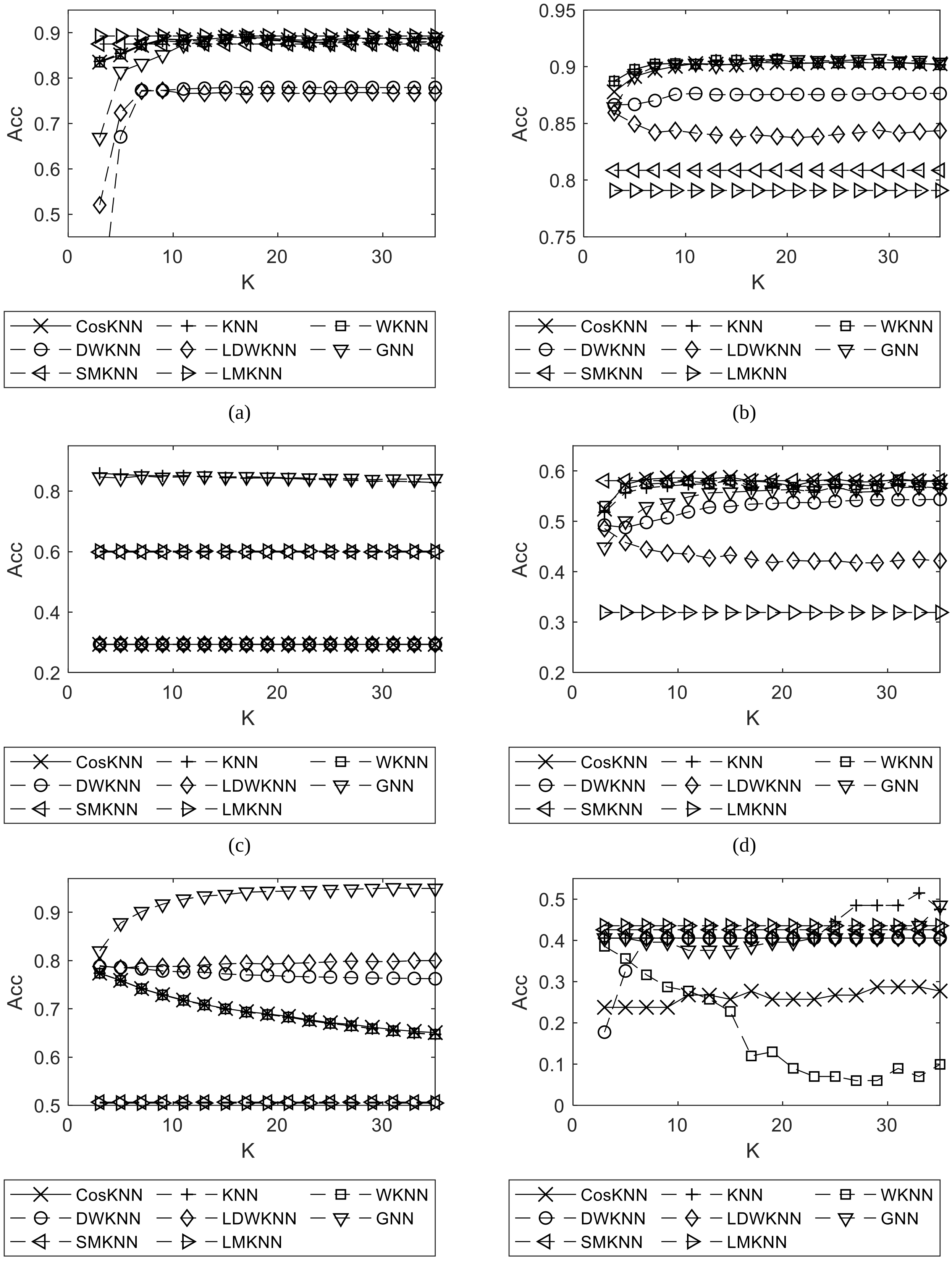
Overall, CosKNN shows good performance on the KEEL dataset, of the six datasets, there are four datasets of which accuracy of CosKNN is classified as high compared to others. It indicates that CosKNN can potentially solve irrelevant class problems because CosKNN can maintain curation performance even though the K value increases. CosKNN can be further developed using large amounts of data and classes.

4.5 Performance analysis with other classification methods

We also conduct performance analysis between CosKNN with other classification results, namely Support Vector Machine (SVM) [13] and Decision Tree (DT) [7]. SVM is a classification method that attempts to maximize the hyperplane boundary. The method is based on statistical learning theory of which results are significantly better than other methods. SVM also performs well on datasets with high dimensions. In this study, we use RBF as the kernel function. The SVM also uses multiclass SVM to solve multiclass problem for eye milkfish and iris datasets. DT is a tree used as a reasoning procedure in order to obtain answers to classification problem. Flexibility constitutes the key point of this method, especially as it provides the advantage of suggestion visualization (in the form of a decision tree) that makes the prediction procedure observable. In this study, we use C4.5 as the DT algorithm.

The result of the comparison is presented in Table 3. From the table, CosKNN indicates better result compared to other methods on 5 datasets in which CosKNN achieves highest accuracy on the iris, vertebral column, diabetic retinopathy, seeds,

Received: October 30, 2019. Revised: January 6, 2020.



(a) (b)

(c) (d)

(e) (f)

Figure. 5 Performance comparison with KEEL-dataset repository: (a) balance dataset, (b) banana dataset, (c) phoneme dataset, (d) yeast dataset, (e) ring dataset, and (f) zoo dataset

Table 3. Performance comparison of CosKNN with other methods

Dataset	**Accuracy (%)**									
	CosKNN	**K-NN**	**WKNN**	**DWKNN**	**LDWKNN**	**GNN**	**SMKNN**	**LMKNN**	**SVM**	**DT**
Milkfish eye	60.89	59.32	63.60	**69.94**	67.14	63.96	67.38	56.07	66.03	63.41
Iris	**96.03**	94.74	95.13	95.21	94.54	94.54	94.0	90.0	65.33	92.0
Vertebral Column	**80.95**	80.34	80.02	76.31	71.67	79.11	73.87	71.61	78.71	80.90
Diabetic Retinopathy	**65.85**	65.41	35.35	63.46	61.57	65.04	64.03	63.16	53.26	60.47
Wine	96.79	96.49	96.92	95.54	92.29	95.20	92.13	**97.19**	**97.19**	91.57
Blood	70.24	78.04	**78.18**	71.86	68.58	78.01	76.74	76.07	76.47	75.13
Audit data	62.63	92.67	37.37	62.63	62.63	97.52	37.37	37.37	99.36	**1**
Divorce	94.12	97.65	13.18	97.30	**97.82**	95.78	95.88	97.65	89.41	96.47
Seeds	**91.81**	91.31	91.42	90.44	87.95	91.28	89.05	90.48	92.38	90.48
Balance	88.06	87.65	87.91	74.12	75.00	85.95	87.52	**89.28**	87.52	78.4
Banana	**90.24**	**90.24**	**90.24**	87.44	84.24	90.10	80.87	79.09	64.21	87.77
Phoneme	29.35	84.29	29.35	29.35	29.35	84.43	59.86	60.18	**87.52**	78.4
Yeast	57.75	56.39	57.13	52.71	43.14	54.83	**58.09**	31.94	19.95	51.48
Ring	69.64	69.48	69.49	77.15	79.35	92.82	50.66	50.47	**95.43**	86.66
Zoo	26.26	43.25	17.44	38.75	40.57	40.45	42.57	43.55	**48.51**	40.57

and banana datasets. As a matter of fact, the highest accuracy is achieved by LDWKNN of 97.82% on the divorce dataset, but the accuracy of other methods on that particular dataset is also above 90 including LMKNN, DWKNN, DT, SMKNN, GNN, and CosKNN, of which accuracy of those methods are highly significant. The compared classification methods of SVM and DT indicate better performance on the wine, phoneme, ring, and zoo datasets for SVM, and data audit dataset for DT. A specific case in the comparison is found in the yeast and zoo datasets in which accuracy for all methods indicates below 60% that it is reasonable if CosKNN does not show good performance; the highest accuracy is found in SMKNN of 58.09%.When sorted from the highest accuracy of all datasets, the superior method is CosKNN, SVM, WKNN, KNN, DWKNN, LDWKNN, SMKNN, DT in which each method is superior on 5 datasets, 4 datasets, 2 datasets, 1 dataset, 1 dataset, 1 dataset, 1 dataset, 1 dataset, respectively. The performance of GNN is relatively the same as the other methods, but it does not achieve high accuracy.

Based on the analysis conducted, it can be concluded that CosKNN generally presents superior performance compared to other methods, despite the differences in the compared methods. The highest accuracy on the CosKNN method is 96.79% on the wine dataset. The compared methods of which performance are relatively similar to CosKNN are DWKNN and LDWKNN, even KNN. Besides solving the irrelevant class problem, CosKNN can also solve the same majority votes class problem as such a problem does not occure during testing.

5. Conclusions

Based on the results and discussion, it can be concluded that CosKNN generates high accuracy performance in a number of datasets of which accuracy achieves up to 96.79%. CosKNN is also robust to irrelevant class problem alongside increased number of K by recent weighting system which involves two parties simultaneously. Even though high accuracy performance of CosKNN is similar to the recent refinements of KNN, CosKNN performs better in terms of empirical testing of features generated in certain color spaces. The limitation of CosKNN to be further explored in future studies is that CosKNN highly depends on training set for prediction session, thus innovation is imperative for the KNN-based method in order to store training data as memory for prediction.

Acknowledgments

We would like to express gratitude to the Directorate of Research and Community Service (DRPM) DIKTI that fund this study in the scheme of Basic Research of 2019 at University of Bhayangkara Surabaya, with contract number 008/SP2H/LT/MULTI/L7/2019 on 26 March 2019, and 170/LPPM/IV/2019/UB on 4 April 2019.

References

[1] T. Cover and P. Hart, “Nearest neighbor pattern classification”, *IEEE Transactions on*

Information Theory, Vol. 13, No. 1, pp. 21–27, 1967.

[2] Z. Pan, Y. Wang, and W. Ku, "A new general nearest neighbor classification based on the mutual neighborhood information", *Knowledge-Based Syst.*, Vol. 121, pp. 142–152, 2017.

[3] Ö. F. Ertuğrul and M. E. Tağluk, "A novel version of k nearest neighbor: Dependent nearest neighbor", *Appl. Soft Comput.*, Vol. 55, pp. 480–490, 2017.

[4] J. López and S. Maldonado, "Redefining nearest neighbor classification in high-dimensional settings", *Pattern Recognit. Lett.*, Vol. 110, pp. 36–43, 2018.

[5] J. Gou, H. Ma, W. Ou, S. Zeng, Y. Rao, and H. Yang, "A generalized mean distance-based k-nearest neighbor classifier", *Expert Syst. Appl.*, Vol. 115, pp. 356–372, 2019.

[6] X. Wu and V. Kumar, *The top ten algorithms in data mining*. CRC Press, 2009.

[7] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Pearson Addison Wesley, 2005.

[8] S. M. Ayyad, A. I. Saleh, and L. M. Labib, "Gene expression cancer classification using modified K-Nearest Neighbors technique", *Biosystems*, Vol. 176, pp. 41–51, 2019.

[9] K. U. Syaliman, E. B. Nababan, and O. S. Sitompul, "Improving the accuracy of k-nearest neighbor using local mean based and distance weight", *Journal of Physics: Conference Series*, Vol. 978, No. 1, p. 012047, 2018.

[10] A. B. Hassanat, M. A. Abbadi, G. A. Altarawneh, and A. A. Alhasanat, "Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach", *International Journal of Computer Science and Information Security*, Vol. 12, No. 8, pp. 33-39, 2014.

[11] J. Gou, T. Xiong, and Y. Kuang, "A Novel Weighted Voting for K-Nearest Neighbor Rule", *J. Comput.*, Vol. 6, No. 5, 2011.

[12] G. E. A. P. A. Batista, and D. F. Silva, "How k-nearest neighbor parameters affect its performance", *JAIIO - Simposio Argentino de Inteligencia Artificial*, 2009.

[13] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 2009.

[14] E. Prasetyo, R. D. Adityo, and R. Purbaningtyas, "Classification of Segmented Milkfish Eyes using Cosine K-Nearest Neighbor", In: *Proc. of International Conference on Applied Information Technology and Innovation,* pp. 120–125, 2019.

[15] F. Klein, *Elementary mathematics from an advanced standpoint. Geometry*. Dover Publications, 2004.

[16] D. Dua and C. Graff, "UCI Machine Learning Repository", 2019. [Online]. Available: http://archive.ics.uci.edu/ml.

[17] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, and J. M. Garrell, "KEEL: a software tool to assess evolutionary algorithms for data mining problems", *Soft Comput.*, Vol. 13, No. 3, pp. 307–318, 2009.

[18] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, and L. Sánchez, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework", *J. Mult. Log. Soft Comput.*, 2011.