1-1-2020

Can Reverse Nearest Neighbors Perceive Unknowns?

Payel Sadhukhan
Indian Statistical Institute, Kolkata

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Can Reverse Nearest Neighbors perceive unknowns?

PAYEL SADHUKHAN[1],
[1]Indian Statistical Institute, 203 B.T. Road Kolkata 700108 India (e-mail: payel0410@gmail.in)
Corresponding author: Payel Sadhukhan (e-mail: payel0410@gmail.com).

ABSTRACT A novel open set classifier is presented in this work, where the neighborhood of a test instance is determined using the principles of Reverse k-nearest neighbors (RkNN). The RkNN count of an instance can have any non-negative value less or equal to the size of the training set. While dealing with an open dataset, consisting of *known* and *unknown* classes, the zero count can provide a possible solution for detecting the unknown class. Positive RkNN count along with the nearest RkNN distance information are used to determine the known class classifications. Experiments are carried out on ten real world datasets, with various openness values on five state-of-the-art open set learners and the proposed scheme. Their performance is measured on three evaluating metrics namely *accuracy, average* F_1 *over known and unknown classes, and Known class* F_1. Empirical results indicate comparable to superior performance delivered by the proposed method over the state-of-the-art approaches on all but one dataset.

INDEX TERMS Open set classification, unknown class detection, known class classification, reverse nearest neighborhood

I. INTRODUCTION

A conventional classification task aims to assign the instances to any one of the *known* classes whereas *unknown* class detection deals with recognition of the instances belonging to *unknown* classes in addition to the known ones. An unknown class is discrepated from the known classes on the basis of the non-availability of its (unknown class's) instances during the training phase. Though classification and detection are performed simultaneously by humans, machines often fail to accomplish the latter efficaciously. Perception and consequent detection of unknowns pose a serious challenge for the machine, which is designed to operate in a 'closed' world. Classifier design and presumptions made by us primarily account for such a disparity. We grow and learn in an unknown world with an incrementally growing known subspace whereas our classifiers are trained in a 'closed' setting of known distributions and classes. Traditionally, it is considered ideal when the training set and the test set have as similar distributions as possible. On assuming the above, a classifier is forced to restrict its prediction into the set of training classes. While predicting a test set consisting of seen and unseen class instances, the unseen instances get camouflaged as seen instances and thus get misclassified.

The above mentioned problems can be generalized as follows. At the training phase we have instances belonging to any one of the c possible classes where $c \geq 1$. Unlike regular classification, during testing the instances can be a member of any one of the $c + u$ classes, $u \geq 1$, the c *known* classes are seen during the training as well as test phase while the remaining u classes which constitute the set of *unknown* class/es appear in the test phase only. Before proceeding with further details, we should distinguish an open world recognition from anomaly detection and outlier detection. While dealing with the latter, one has to detect the rare events or instances which deviate from the available population. In an open set scenario, we have an universe. During training, we are provided information about only a few aspects (known classes) of the universe but in the test phase we have to classify what we have seen before (known classes) and detect the ones that that we have not encountered earlier (unknown classes). We may also have some classes which we do not encounter in either training or test phases. Openness of a dataset is the degree of unknownness in the dataset. For quantifying this characteristic the following definition is provided by [1].

$$Openness = 1 - \sqrt[2]{\frac{2 * |Training\ classes|}{|Target\ classes| + |Test\ classes|}}$$

Target class consists of all the training and test classes as well as the leftover unknown classes that do not participate in the training and testing. The goal is to predict the *known*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

class correctly along with recognition and prediction of the unknown class test instances. In practical scenario, we cannot really quantify the degree of openness because the unknown remains unknown.

Extant classifiers predict 'closed' class-memberships in terms of the known classes only. A true open set solution has to possess the capability of saying 'no' or 'unknown' when a test point is coming from an unknown class. In this work, we attempt to answer this by raising a simple question. Instead of querying a test instance $\mathbf{p}$ about its nearest neighbors in a given search space, we query about the reverse nearest neighbors of $\mathbf{p}$. Reverse k-nearest neighbors of an instance $\mathbf{p}$ are all those points in a given search space whose k-nearest neighborhood contains $\mathbf{p}$. When n is the total number of training points, $\mathbf{p}$'s RkNN count can be anything between n and 0. When all the instances in search space has $\mathbf{p}$ as one of its k-NN, R-kNN count of $\mathbf{p}$ is n and it indicates sufficient belongingness of $\mathbf{p}$ to the given search space. On the contrary, when $\mathbf{p}$ does not lie in any point's neighborhood, it indicates significant disharmony between $\mathbf{p}$ and all others members of the search space. The latter situation is our motivation for rejecting and unfolding the prediction into the unknown class. In this paper, we present a novel reverse k-nearest based classification scheme which performs simultaneous classification into the known classes as well as to the unknown class. The key aspect of our work is the simplicity of the scheme. We do not require to provide any information other than the training or known class instances and their respective class labels. The proposed scheme does not require any distance based thresholding for demarcation of the known and unknown spaces, the only user-modulated parameter is neighborhood size k. In the next section, we present the literature review.

II. OPEN SET CLASSIFICATION

A closed set classifier makes it's prediction within the set of classes that it encounters in the training phase. It assumes that all classes of the test data (queries) were well represented at the training phase. Closed set classifiers, mostly built on Bayesian Optimal Posterior Probability model assumes that a fixed set of classes shares the real space and it (the classifier) has to predict to any of these classes according to the class boundaries. If the number of classes is c, it computes $\mathrm{P}(\mathrm{C}_j|\mathbf{x})$ for $j = 1, 2, \ldots, c$ and assigns the query instance to the class i which gives maximum value of $\mathrm{P}(\mathrm{C}_i|\mathbf{x}),\ i = 1,\ 2, \ldots, c$.

Open set classification is a type of classification problem where an instance belonging to the unknown class appears at the test phase. Unknown class denotes a class which had zero or no representation at the training time. An open set classifier can encounter instances from such unrepresented class/es at the test phase and should predict them as unknown instead of classifying them into the known classes. Open set classification is different from anomaly detection as well as incremental learning. In incremental learning, the scheme is to add the newly encountered classes to the database of seen classes on encountering it's instance. On the contrary, in open set classification it is not desirable for us to add the unknown/ unrepresented classes in the seen domain. What is unknown should remain unknown but should be recognized as unknown. Anomaly detection is a task in which a a rare event or observation like outliers is identified as different from the regular ones. In anomaly detection, we do not need to discriminate the known classes. Unlike anomaly detection, in open set classification, the classifier does not make any assumption about the cardinalities of the unknown/ unknowns, no information is available about the unknowns at the training time (as well as the test time). The problem of open set classification requires us to have provision for the unknown and unrepresented class besides discriminating between and correctly predicting the known classes. In an efficient open set classifier, two of these characteristics should be consequential of the scheme. Reverse nearest neighborhood provides an elegant way of solving these two issues simultaneously. Our scheme based on reverse nearest neighborhood principles is presented in Section V. We discuss the extant works on open set classification and the machine learning applications of reverse nearest neighbor principles in the next section.

III. LITERATURE REVIEW

This work deals with open set classification using the principles of *reverse nearest neighborhood*. Reverse k-nearest neighbor (RkNN) principle has been used in various applications but RkNN based classification has not been implemented or addressed in any existing piece of work so far. Keeping in mind these two aspects, the literature review of this work is presented in two contexts. First, we discuss extant works in the field of open set classification. In the second part, we present a brief discussion on works that have used principles of reverse nearest neighborhood to achieve some machine learning goals.

Open set recognition in a mixed bag of seen and unseen classes has appealed to the data science community for quite some time. Despite the number of works not being numerous till date, the techniques applied are quite diverse. [2] implemented unknown class recognition through estimation of prior probability of the known classes and posterior probabilities for the known as well as unknown classes. One class classifiers which try to model a class only through its positive instances has been one of the foremost solutions to deal with open world problem. Though it is sufficient to deal with a setup having one known class and the rest as unknown class, the need for more refined scheme which can tackle two or more known classes along with the unknown is natural. [1] addressed this issue by implementing open set recognition in the context of two known and the remaining as unknown class. They modified the conventional SVM for this. Besides drawing a decision boundary between the two known classes, [1] added one more hyperplane which separated the unknown class from the known subspace. The learning of the classifier model followed by incorporating Compact Abating Probability (CAP) is another solution. An amalgamation of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

TABLE 1: Categorization of different problems

Task	Objective	Training Data	Prediction of test data
Closed set classification	Discrimination between the classes	Data from all classes	To any one of the existing classes
Anomaly Detection	Detecting abnormal data like outlier	Adequate normal class data, few outliers	Classification as outlier-yes or no
Incremental Learning	Dynamic classifier modeling in a changing world	Adequate and sequential training and test data	To any one of the existing classes as well as classifier updation
Open set Recognition	**Discrimination between known classes and identifying unknown class instances**	**Data from all known classes only**	**To any one of the known class or to the unknown class**

the extreme value theory and the probabilistic CAP model is implemented in [3] to classify the instances from the known class/es and subsequently recognize the unknowns. CAP model considers a decreasing confidence of class membership as one moves away from a known class instance into the unmarked space. Regions beyond a thresholded radius are subsequently categorized as the unknown or open space. In [4], a posterior probability estimator is implemented for each training class. A test instance is predicted into a known class only if the maximum probability surpasses the threshold. If none is found, the point is recognized as unknown. Distribution learning of the known classes through Extreme Value Theory (EVT) and incremental learning are incorporated in [5] to implement open set classification. Object detection under openset constraints are solved using drop-out sampling approach in [6].

A few recent schema have incorporated neural networks to recognize samples from unseen classes along with classification of samples into seen or known classes. The scheme by [7] is based on an ensemble of Convolutional Neural Network with a provision for open set recognition. It separates plant images from unknown non-plant images. Open set recognition through weightless neural network has been explored in [8]. In [9], a neural network based classifier detects the unknown samples through comparison and computation of the similarity between the unknown data and the stored or bounded knowledge. [10] on the other hand proposed a theoretically sound method to estimate the 'sampling window' of the training data. Samples generated from regions outside the sampling window are used to represent the unknown world (class). They have trained a neural network to learn the known and unknown classes. In [11], Generative Adversarial Network (GAN) based approach is to separate the differential identity components of face to generate an-identity preserving open set face synthesizer.

[12] has tweaked traditional k-NN based classifier to facilitate open set recognition. It has proposed two schema. In the first variant, an instance is classified as unknown on non-agreement in class labels of its first two neighbors, agreement assigns the instance to its first (as well second) neighbor's class. The second considers looks at the distances of the test instance's from its two nearest neighbors belonging to different classes and calculates their ratio (nearer/ farther). If the ratio is beyond a threshold, the instance is classified as unknown and vice versa. [13] has employed a data fusion technique by integrating open-set graph-based optimum-path forest (OSOPF) classifier with genetic programming (GP) and majority voting fusion techniques for open set recognition. [14] explores the technique of classification-reconstruction learning for open set recognition.

Reverse nearest neighborhood might just seem a flip side of the k-nearest neighborhood, but it has been used to solve a number of data mining subtasks. Outlier detection in an unsupervised context and in data streams is implemented using reverse-nearest neighborhood by [15] and [16] respectively. Efficient reverse nearest search in metric spaces is achieved by [17]. [18] explored reverse nearest neighbor principles for protein information mining in bioinformatics. Problems on spatial data search is also addressed by the same in [19]. Reverse-nearest neighbor based algorithms have solved spatio-temporal query and range queries in [20]. A work by [21] has implemented data clustering algorithm via RkNN. RkNN explores the locality of the instances to obtain meaningful data mining. In recent years, the techniques of local information exploration, feature embedding and lower rank and sparse subspace recovery have been used as a backbone in a number of diversified domains. In [22], a technique of adaptive embedded label propagation with weight learning is used for classification of real-world image datasets. For efficient classification of images, [23] integrates incorporation of embedded low-rank and sparse principal features with feature coding error and classification error. [24] uses analysis-based trained dictionary learning model for retrieval of query images. [25] is another important work on the same context. It introduces a structured and scalable dictionary learning framework to handle image analysis.

A technical elaboration of the backbone of our work, the *reverse-nearest neighbor principles* is presented in the next

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

section.

IV. REVERSE NEAREST NEIGHBORHOOD

Definition 1: Given a set of instances $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ ($\mathrm{X} \subset IR^N$) and a point $\mathbf{p}$ ($\mathbf{p} \in IR^N$), a Reverse Nearest Neighbor query concerning $\mathbf{p}$ in search space X retrieves all the points $\mathbf{x}_\mathrm{i} \in X$ that have $\mathbf{p}$ as their nearest neighbor. Thus, a Reverse k-Nearest Neighbor (RkNN) search returns all those points $\mathbf{x}_\mathrm{i} \in X(\mathrm{i}=1, 2, \ldots, \mathrm{n})$ whose k-nearest neighborhood contain $\mathbf{p}$.

Extant neighborhood estimators estimate the neighborhood of a query instance $\mathbf{p}$ through the distribution of the neighboring instances around $\mathbf{p}$. Neighborhood demarcation is made via a surrounding hypersphere or through the encompassment of a fixed number of nearest instances around the query point. They do not take into account the locale of the query instance $\mathbf{p}$ in the neighborhood of the other search points. Reverse k-nearest neighborhood realizes the neighborhood paradigm in the latter light. To obtain reverse nearest neighbors of a query point $\mathbf{p}$, all points in a given search space are queried about their k-nearest neighbors to find if $\mathbf{p}$ is one of them. It is interesting to note that unlike k-NN (where a query point has exactly k neighbors), the number of RkNNs of query instance $\mathbf{p}$ can be anything between 0 and n (the search space cardinality). Depending on the data distribution, a query instance $\mathbf{p}$ can remain absent from the k-nearest neighborhood of all the instances in the training data, subsequently the RkNN count of $\mathbf{p}$ would be zero, if $distance(\mathbf{p}, \mathbf{x}_\mathrm{i}) >$*distance of* $\mathbf{x}_i$ *from its* k^{th}*-nearest neighbor*, $\forall i$. The other extreme case arises when the query point $\mathbf{p}$ has the RkNN count of n, the size of search space by virtue of its presence within the k-nearest neighborhood of all the instances in the search space. $0 \leq$ RkNN count $\leq$ n is the possible range of RkNN values. For $\mathbf{p}$, its RkNNs constitute its neighborhood in the given search space X. More the RkNN count of $\mathbf{p}$ in X, more is its agreement with the instances in X. A zero RkNN cardinality indicates a significant disharmony between the query point $\mathbf{p}$ and the instances in the training set, and it will be fair to assume that $\mathbf{p}$ comes from an entirely different distribution. This is our principal motivation for predicting the unknown class instances (along with the usual prediction for the known classes) in a mixed bag of known and unknown class instances.

A. KNOWN AND UNKNOWN SPACE MODULATION

According to our scheme, a region of positive RkNN count constitutes the known subspace (subspace covered by the known classes). We have a search space X (as defined in the previous subsection) and a query instance $\mathbf{p}$. Let $d_k(\mathbf{x}_\mathrm{i})$ be the distance of $\mathbf{x}_i$ from its k^{th} nearest neighbor in the given search space X (excluding itself). A hypersphere $S_{k\mathbf{x}_\mathrm{i}}$ of radius $d_k(\mathbf{x}_i)$, centered at $\mathbf{x}_i$ is assumed as the k^{th}-nearest neighborhood of $\mathbf{x}_\mathrm{i}$. $S_{k\mathbf{x}_\mathrm{i}}$ constitutes the known space corresponding to instance $\mathbf{x}_\mathrm{i}$. If $\mathbf{p}$ lies inside $S_{k\mathbf{x}_\mathrm{i}}$, $\mathbf{x}_\mathrm{i}$ becomes RkNN of $\mathbf{p}$. Let $d(\mathbf{p}, \mathbf{x}_\mathrm{i})$ be the distance between $\mathbf{p}$ and $\mathbf{x}_\mathrm{i}$. $\mathbf{p}$ can lie within $S_{k\mathbf{x}_\mathrm{i}}$ if $d_k(\mathbf{x}_\mathrm{i}) > d(\mathbf{p}, \mathbf{x}_\mathrm{i})$. Let S be the subspace that is covered by the known class.
$\mathrm{S} = \bigcup_{\mathbf{x}_\mathrm{i} \in X} S_{k\mathbf{x}_\mathrm{i}}$.

If $\mathbf{x}_\mathrm{i}$ is a vector in IR^N, then S is a subset of IR^N. Here, S implicitly defines the sampling window of the training data and hence can be viewed as defining the boundary of the known classes. The volume of $S_{k\mathbf{x}_\mathrm{i}}$ or the known subspace spanned by $\mathbf{x}_\mathrm{i}$ is dependent on $d_k(\mathbf{x}_\mathrm{i})$. In Figure 1, we scatter-plot 100 points each from two Gaussian distributions $\mathrm{N}_1(\mu_1, \Sigma_1)$ and $\mathrm{N}_2(\mu_2, \Sigma_2)$ where $\mu_1 = [50, 50]$, $\mu_2 = [20, 15]$, $\Sigma_1 = \left(\begin{smallmatrix} 49 & 0 \\ 0 & 49 \end{smallmatrix}\right)$ and $\Sigma_2 = \left(\begin{smallmatrix} 9 & 0 \\ 0 & 9 \end{smallmatrix}\right)$. The points of N_1 are labeled in red while the ones from N_2 are labeled in blue. The k^{th} nearest neighbor distance or $d_k(\mathbf{x}_\mathrm{i})$ for points in N_1 are usually greater than that of points in N_2. Accordingly, the points from N_1 spans a larger volume of known space than that of N_2. Thus, the RkNN gives an automatic modulation of the known class spaces depending on the local distribution of the training data points. In Figure 1, the spaces marked with yellow color corresponds to the unknown region. It is auto-adaptive to the class boundaries which vary from class to class. This is desirable property while dealing with variable data distributions.

B. PRINCIPLES OF REVERSE NEAREST NEIGHBORHOOD AND CLASSIFICATION

Mathematically, the principles of reverse-nearest neighborhood is another way of quantifying the neighborhood of the points. But reverse k-nearest neighbor principles have not been used for handling problems of classification. k-nearest neighborhood principles has a framework of classifying test data points. In k-nearest neighborhood based classifier, the confidence of the contending classes is calculated from the class membership of the k nearest neighbors. A test point is likely to belong to a class which has the highest number of it's (test point's) neighbors. The working principles of reverse k-nearest neighborhood is analogous to that of k-nearest neighbor's. We can easily extend a similar classification protocol using reverse nearest neighborhood. For a certain k value, we can find the reverse k-nearest neighbors of a test point $\mathbf{p}$ and classify $\mathbf{p}$ to the class with highest number of reverse nearest neighbors. It is indeed true that getting an reverse nearest neighborhood is also possible. A RkNN based classifier has to possess proper strategies for handling the zero neighborhood count according to the devoir of the problem. In our case, the zero RkNN count allows us to solve the issue of open set recognition in a natural manner, hence we allow it as it is in our scheme.

The approach and its algorithm is elaborated in the next section.

V. PROPOSED WORK

A. APPROACH

While classifying a test instance, classifiers operating on principles of density estimation predict the class having the highest density estimate (that is the class with the highest number of neighbors) as the test class. Now, let us assess their potential to address an unknown class classification task. For

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

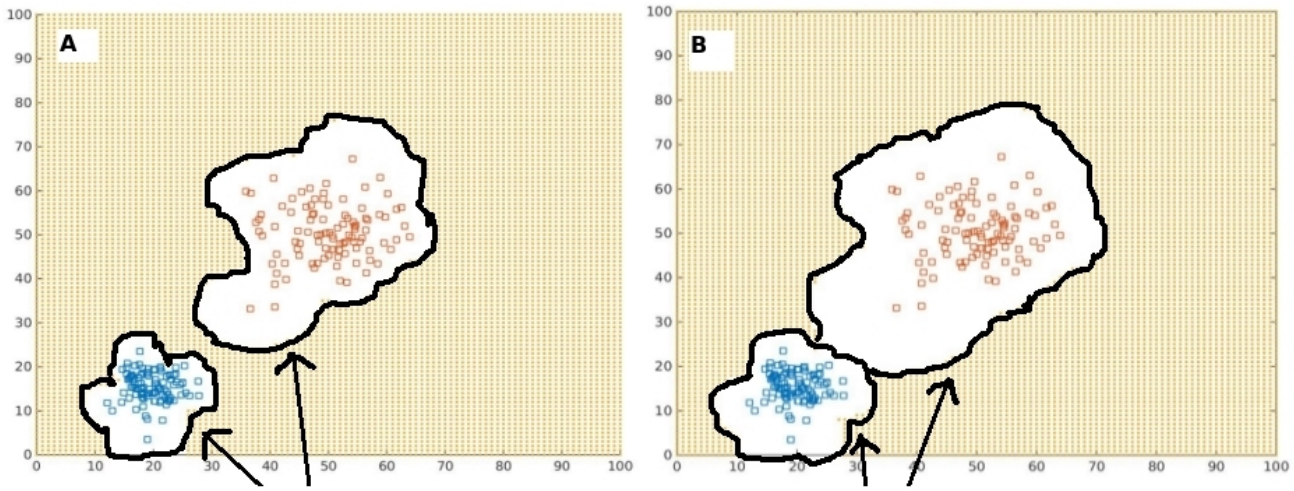


FIGURE 1: This figure shows known-unknown subspace for a toy example.

Red points and blue points denote two different known classes. Let red points and blue points denote class 1 and class 2 respectively. Class 1 points come from a Gaussian distribution with $\mu = [50, 50]$ and $\Sigma = \left(\begin{smallmatrix} 49 & 0 \\ 0 & 49 \end{smallmatrix}\right)$. Similarly, Class 2 come from a Gaussian distribution with $\mu = [20, 15]$ and $\Sigma = \left(\begin{smallmatrix} 9 & 0 \\ 0 & 9 \end{smallmatrix}\right)$. 100 points from each of these classes are shown in the figures. White space denotes the known subspace and the yellow colored region denotes the unknown subspace. It can be noted that spread of known subspace around each class increases with the sparsity of the distribution. Class 2, being a dense class with lower value of sigma spans a smaller area representing the known subspace. On the other hand, known space volume around class 1 points is high since the relative distribution of the points is sparser. 1 Fig A and Fig B shows the known-unknown subspace delimitation at $k = 5$ and $k = 10$ respectively for the same set of data points. It can be noted that the known space volume increases with increasing the k value. At k value 10, known subspaces of the two classes expand and we get an overlap between the two.

a window based classification paradigm [26], the number of neighbors inside the window volume can vary from zero to maximum cardinality of the search space. Though a zero neighbor count can be used for unknown class detection, when the density distribution is highly skewed, a single volume threshold is not expected to work well across the entire dataset. In addition to this, the volume thresholding is not automatic and needs empirical and manual modulations. In k-NN based classification motivated by [27], the k-nearest neighbors of a query point are searched in the training space. Consequently, k-NN classifier can predict only one of the known classes. There is no provision for unknown class detection in this scheme unless some thresholding is involved.

An efficient neighborhood based solution of open set detection should detect test instances which falls into zero neighborhood zones of a given known space and subsequently reject them as unknown class instances. On a similar note, a positive neighbor count of a test instance indicates a finite known class membership and should be predicted to a class from the training instances. It is desirable that both these tasks (unknown class rejection and known class classification) should be consequential of the scheme and without any thresholding. The scheme should be uniform as well as robust to non-uniform class distributions in a dataset.

In order to design a scheme satisfying the said requirements, we propose a neighborhood based classifier where the neighborhood definition is a bit different from the one assumed in the above paragraph. Reverse k-nearest neighbors (RkNN) of a query instance p is searched in the training space X. When the RkNN count of p is zero, we classify p to the unknown class. In other words, if $\mathbf{p} \in \mathrm{S^C}$ (the complement of the known subspace or sampling window, S), then $\mathbf{p}$ is coming from some unknown class. When RkNN count of p is > 0, then the class-specific membership scores are computed. Membership score of p for a class depends on the number of RkNNs count from that class and the distance between p and the nearest RkNN in that class. The membership value increases with increase in the RkNN count and a decrease in the distance of the nearest RkNN. The instance p is assigned to the training class with the highest membership score.

B. THE PROPOSED METHOD

We have an open instance set D, consisting of two mutually exclusive partitions $\mathrm{D_{tr}}$ and $\mathrm{D_{te}}$. $\mathrm{D_{tr}}$ and $\mathrm{D_{te}}$ represent the training set and test set respectively. The respective number of classes in $\mathrm{D_{tr}}$ and $\mathrm{D_{te}}$ are c and $c+u$. The extra u classes in $\mathrm{D_{te}}$ remain unseen during the training. We consider u unseen classes together as a single unknown class resulting in $c + 1$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

classes for the test set, D_{te}. Classes $1, 2, \ldots, c$ correspond to the known classes and $c+1^{th}$ class correspond to the unknown class. We also assume that the neighborhood size is a fixed positive integer k.

We will classify a test instance $\mathbf{p} \in D_{te}$ in IR^N into any one of the known classes $1, 2, \ldots, c$ or to the unknown class, $c+1$ on the basis of the training set D_{tr} only.

Let $D_{tr}=\{(\mathbf{x}_i, \mathbf{y}_i) |\ 1 \leq i \leq n\}$, where $\mathbf{x}_i$ is a training instance vector in IR^N and $\mathbf{y}_i$ is its corresponding class label. Hence, the number of training instances is n. The instances in D_{tr} belong to the known classes only, hence their memberships lie in $\{1, 2, \ldots, c\}$. Next we provide a stepwise description of the algorithm. Algorithm 1 depicts the same.

Step 1: We find the RkNN of $\mathbf{p}$ in D_{tr}. The outputs of the lookup is stored in $R_{\mathbf{p}}(.)$ and $M_{\mathbf{p}}(.)$.

$$R_{\mathbf{p}}(i) = \begin{cases} 1, & \text{if } \mathbf{x}_i \text{ is a } RkNN \text{ of } \mathbf{p} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$M_{\mathbf{p}}(i) = \begin{cases} \text{distance}(\mathbf{p}, \mathbf{x}_i), & \text{if } \mathbf{x}_i \text{ is a } RkNN \text{ of } \mathbf{p} \\ \infty, & \text{otherwise} \end{cases} \quad (2)$$

Remarks: $R_{\mathbf{p}}(i)$ is a vector which can take only two values 0 or 1. $M_{\mathbf{p}}$ is a vector in R^n.

Step 2: Now, we obtain the class-wise RkNN statistics for $\mathbf{p}$ in $N_{\mathbf{p}}(j)$ and $\text{Mem}_{\mathbf{p}}(j)$. By 'class' only the seen training classes are meant. We calculate the distance of $\mathbf{p}$ from its nearest RkNN in class j and store the same in $N_{\mathbf{p}}(j)$. When $\mathbf{p}$ does not find a RkNN in class j, $\mathbf{p}$ is considered unreachable from the entire class j and $N_{\mathbf{p}}(j)$ is set to ∞. Next, we compute $\text{Mem}_{\mathbf{p}}(j)$. It indicates the overall membership of $\mathbf{p}$ to class j. $\text{Mem}_{\mathbf{p}}(j)$ depends on the RkNN count from class j as well as $N_{\mathbf{p}}(j)$, the distance from the nearest RkNN of $\mathbf{p}$ from class j.

$$N_{\mathbf{p}}(j) = \{min(M_{\mathbf{p}}(i)); i = 1, 2, \ldots, n, \mathbf{y}_i = j, R_{\mathbf{p}}(i) = 1\} \quad (3)$$

For each class j, $j = 1, 2, .., c$, class membership score of $\mathbf{p}$, $Mem_{\mathbf{p}}(j)$ is calculated.

$$Mem_{\mathbf{p}}(j) = \frac{1}{N_{\mathbf{p}}(j)} \sum_{\substack{i \\ \mathbf{y}_i=j}}^{n} R_{\mathbf{p}}(i) \quad (4)$$

Remarks : A higher value of class-specific RkNN count and smaller distance between $\mathbf{p}$ and the nearest class-specific RkNN indicates higher confidence of $\mathbf{p}$ to that class. A zero RkNN count from a class results in zero confidence of the instance to that class. Note that $\text{Mem}_{\mathbf{p}}(j)$ could be greater than 1. By RkNN principles, even for the same k value, the neighbor count of different points vary (depending on their configurations). In such a scenario, it is difficult to adopt the distribution of their distances (as the number of neighbors would vary widely). So, in $\text{Mem}_{\mathbf{p}}(j)$, we have considered the nearest neighbor distance from class j only.

A toy example of $\text{Mem}_{\mathbf{p}}(j)$ calculation: Let us have two classes A and B. Let the test point be $\mathbf{p}$. We have the information about $\mathbf{p}$'s RkNN counts and it's respective nearest neighbor distances from class A and class B also. Let the RkNN count from class A and class B be 2 and 3 respectively. Let the nearest neighbor distances $N_{\mathbf{p}}(A)$ and $N_{\mathbf{p}}(B)$ be 0.5 and 1 respectively.

$$\text{Mem}_{\mathbf{p}}(A) = \frac{1}{0.5} \times 2 = 4$$

$$\text{Mem}_{\mathbf{p}}(B) = \frac{1}{1} \times 3 = 3$$

This indicates the importance of nearest neighbor distance in our scheme. Though the RkNN count from class B is higher than that of class A, $\mathbf{p}$'s class-membership to A is greater than that of B by virtue of the smaller distance. Besides reverse nearest neighbor configuration, the nearest neighbor's proximity from a class plays a decisive factor in computing the class-memberships.

Step 3: In this step, we will classify $\mathbf{p}$ to any one of the known classes $1, 2, .., c$ or to the unknown class on the basis of class membership scores. Max_Mem($\mathbf{p}$) value 0 indicates a zero RkNN count from entire set of known (training) classes. It indicates remoteness of $\mathbf{p}$ from the training classes and $\mathbf{p}$ is classified to the unknown class. Max_Mem($\mathbf{p}$) $>$ 0 signifies the presence of $\mathbf{p}$ within some known class space and $\mathbf{p}$ is assigned to the class with Max_Mem($\mathbf{p}$). Class_prediction($\mathbf{p}$) gives the final prediction for $\mathbf{p}$, it can be the unknown class or any one of the known classes.

$$\text{Max\_Mem}(\mathbf{p}) = \max_{j}\ \text{Mem}_{\mathbf{p}}(j), j = 1, 2, \ldots, c \quad (5)$$

$$\text{Max\_Mem\_class}(\mathbf{p}) = \text{argmax}_j\ \text{Mem}_{\mathbf{p}}(j) \quad (6)$$

$$\text{Class\_prediction}(\mathbf{p}) = \begin{cases} \text{unknown class} & , \text{Max\_Mem}(\mathbf{p}) = 0 \\ \text{Max\_Mem\_class}(\mathbf{p}), & \text{otherwise.} \end{cases} \quad (7)$$

General Remarks:

1) *Not in the neighborhood of any*: Our scheme classifies an instance to the unknown class only when the instance does not possess any RkNN in the known training space. In other words, it does not lie in the k^{th}-nearest neighborhood of any training instance.
2) *Do we need to search the training space for each test instance?* A training instance $\mathbf{x}_i \in D_{tr}$ can be a RkNN of a test instance $\mathbf{p}$ only if $d(\mathbf{x}_i, \mathbf{p})$ is less than the k^{th} nearest neighbor distance of $\mathbf{x}_i$. Here, we assume that k^{th} nearest neighbor search of each $\mathbf{x}_i$ is done in the training space only once and stored for computations in the later stages. We conduct a single k-nearest neighbor search of the entire training set D_{tr} and find the distance of the k^{th} nearest neighbor of each training point

$\mathbf{x}_\mathrm{i}$, , i=1,2, . . . ,n. Next, we just to need to find and compare $d(\mathbf{x}_i, \mathbf{p})$ with k^{th}-nearest neighbor distance of $\mathbf{x}_\mathrm{i}$. If the former is lesser, $\mathbf{x}_\mathrm{i}$ becomes a RkNN of $\mathbf{p}$ and vice versa. Hence, the RkNN lookup of the entire test instance set requires just one k-NN search of the training set (in context of itself).

Algorithm 1 Reverse-nearest neighborhood based classification

Input: Training set $\mathrm{D_{tr}}$ with c known classes, Test point $\mathbf{p}$ which may belong to one of the c known classes or the unknown class (c+1).

Output: Class prediction of $\mathbf{p}$.

Search for RkNN of $\mathbf{p}$ in $\mathrm{D_{tr}}$

$$R_{\mathbf{p}}(\mathbf{i}) = \begin{cases} 1, & \text{if } \mathbf{x}_\mathrm{i} \text{ is a } RkNN \text{ of } \mathbf{p} \\ 0, & \text{otherwise} \end{cases}$$

$$M_{\mathbf{p}}(\mathbf{i}) = \begin{cases} \text{distance}(\mathbf{p}, \mathbf{x}_\mathrm{i}), & \text{if } \mathbf{x}_i \text{ is a } RkNN \text{ of } \mathbf{p} \\ \infty, & \text{otherwise} \end{cases}$$

for each class j from $1, 2, .., c$ **do**

  Calculate $N_j(\mathbf{p}) = min(M(\mathbf{p}, \mathbf{x}_\mathrm{i})), i = 1, 2, \ldots, n,$ $\mathbf{y}_\mathrm{i} = j$, $\mathbf{x}_i$ *is a RkNN of* $\mathbf{p}$

  Calculate $Mem_j(\mathbf{p}) = \frac{1}{N_j(\mathbf{p})} \sum_{\substack{i \\ \mathbf{y}_\mathrm{i}=j}}^{n} R_{\mathbf{p}}(\mathbf{x}_\mathrm{i})$

end for

$Max\_Mem(\mathbf{p}) = \max_j \ Mem_j(\mathbf{p})$

$Max\_Mem\_class(\mathbf{p}) = argmax_j \ Mem_j(\mathbf{p}),$

if $Max\_Mem(\mathbf{p})$=0 **then**

  Classify $\mathbf{p}$ as unknown (c+1).

else

  Classify $\mathbf{p}$ to $Max\_Mem\_class(\mathbf{p})$ (known class).

end if

End

VI. EXPERIMENTAL SETUP

In this section,we propose a setup to make a comprehensive assessment of the proposed and competing method's performance on classification of the known classes and detection of the unknown class. A brief outline on the four essentials, namely *Datasets*, *Comparing methods*, *Parameter Optimization* and *Evaluating Metrics* are presented in the following subsections in order.

A. DATASETS

We have employed ten real-world multi-class datasets to evaluate the relative efficacies of the proposed and the comparing methods. Table 2 summarizes the basic statistics of their attributes. MNIST dataset is obtained from https://pjreddie.com/projects/mnist-in-csv/ while the source of the remaining ones is Keel Dataset Repository [28]. MNIST dataset has 784 features and we obtain a Reduced-MNIST version by extracting the top features whose eigenvalue value summation covers 90% feature variance. Reduced MNIST dataset has 79 features. We present the results of both MNIST and Reduced MNIST datasets individually in this work. These datasets are obtained in closed form that is they do not possess any openness and the class information of all the instances are known. In order to accommodate them for the purpose of open set recognition, we have generated open version of each dataset following the same protocol as done by [3]. The first step is to set the cardinalities of the known and unknown classes. For MNIST and Letter datasets, we have followed the recommended partition (by [3]) of 6 known, 1-4 unknown classes and 15 known, 1-11 unknown classes, respectively. For the remaining datasets, the following protocol is adopted.

Let the non-open or regular instance set be denoted by D. $\mathrm{D} = \{(\mathbf{x}_i, \mathrm{y_i})|, \mathbf{x}_i \in \mathrm{X}, \mathrm{y_i} \in \mathrm{C}\}$, $\mathrm{X} \subset IR^N$ and $\mathrm{C} = \{c_1, c_2, \ldots, c_n\}$. Hence the number of classes in the dataset is n. We randomly equi-partition D into a training set D_{tr} and D_{te}. $\mathrm{D}_{tr} \cup \mathrm{D}_{te} = \mathrm{D}$ and $\mathrm{D}_{tr} \cap \mathrm{D}_{te} = \phi$. We will generate open training-test tuple $(\mathrm{D}^o_{tr}, \mathrm{D}^o_{te})$ from D_{tr} and D_{te}. We will select the sets of known classes and unknown classes, C_k and C_u respectively from C. The instances belonging to C_k will appear in both D^o_{tr} and D^o_{te} whereas the instances belonging to the unknown class set, C_u will appear in D^o_{te} only. The cardinality of C_k, denoted by c_k is fixed to $\lfloor 0.5 \times n \rfloor$. The cardinality of C_u, c_u is varied from 1 to $\lceil 0.5 \times n \rceil$. Here, we describe a procedure for generating $(\mathrm{D}^o_{tr}, \mathrm{D}^o_{te})$ at one particular openness.

1) $\mathrm{C}_k = \{\text{A set of } c_k \text{ classes from C}\}$.
   $\bar{C}_k = \mathrm{C} - \mathrm{C}_k$.

2) For a given c_u, $\mathrm{C}_u = \{\text{A set of } c_u \text{ classes from } \bar{C}_k\}$.

3) $\mathrm{D}^o_{tr} = \{(\mathbf{x}_\mathrm{i}, \mathrm{y_i}), |\mathbf{x}_\mathrm{i} \in \mathrm{D}_{tr} \text{ and } \mathrm{y_i} \in \mathrm{C}_k\}$.
   The instances in D_{tr} which belong to C_k goes to the open training set.

   $\mathrm{D}^k_{te} = \{(\mathbf{x}_\mathrm{i}, \mathrm{y_i}), |\mathbf{x}_\mathrm{i} \in \mathrm{D}_{te} \text{ and } \mathrm{y_i} \in \mathrm{C}_k\}$.
   D^k_{te} is the collection of test instances which belong to the known class/es.

   $\mathrm{D}^u_{te} = \{(\mathbf{x}_\mathrm{i}, \mathrm{y_i}), |\mathbf{x}_\mathrm{i} \in \mathrm{D}_{te} \text{ and } \mathrm{y_i} \in \mathrm{C}_u\}$.
   D^u_{te} is the collection of test instances which belong to the unknown class/es. We relabel the instances in D^u_{te} to the unknown class (instead of their actual class labels).

   $\mathrm{D}^o_{te} = \mathrm{D}^k_{te} \cup \mathrm{D}^u_{te}$.
   D^o_{te}, the open test set consists of the test instances belonging to the known classes as well as the unknown class.

By varying c_u in step 2, we vary the openness in D^o_{te}. In our experiments, for each dataset, we repeat step 1 and step 2

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

five times to generate 25 folds of open data (at each openness). After generating the open set versions, we calculate the openness value of each such partition using the formula proposed in [1].

$Openness = 1 - \sqrt[2]{\frac{2*|Training\ classes|}{|Target\ classes| + |Test\ classes|}}$. Target class consists of all the training and test classes as well as the leftover unknown classes that do not participate in the training and testing. An example is illustrated below in **Openness Calculation Example**.

Remarks: As said earlier, we have followed openness generation protocol similar to the state-of-the-art methods. We may note that the number of opennesses generated for a dataset depends on the the number of classes it originally has. Following this, *Vehicle*, a dataset with 4 classes has 2 openness values (0.244 and 0.293) while *Texture*, a 11 class dataset has six openness values in the range 0.233-0.326. Refer to the openness calculation formula stated in Introduction.

Openness calculation example: Let us consider *Dermatology* dataset which has 6 classes. The number of target class for this dataset is always 6. Following the above-mentioned protocol, we have 3 known classes and we will have 1,2 or 3 unknown classes at each stage.

3 known classes, 1 unknown class: Number of training classes=3. Number of test classes (known+unknown)= 4. Number of target classes=6. Following formula, openness =0.225.

3 known classes, 2 unknown class: Number of training classes=3. Number of test classes (known +unknown) = 5. Number of target classes=6. Following formula, openness =0.261.

3 known classes, 1 unknown class: Number of training classes=3. Number of test classes (known+unknown)= 6. Number of target classes=6. Following formula, openness =0.293.

B. PARAMETER OPTIMIZATION

Most of the open set learners, including ours involve parameters whose values have to be determined empirically. The optimized values of these parameters are determined via cross-validation on the training set. We carve out a cross-validation training set, T and validation set V from D^o_{tr} only. For open set classification, we introduce openness in V following the same protocol as described in the above section. If m is the number of classes in D^o_{tr}, we fix the known class and unknown class cardinalities at $\lfloor 0.5 \times m \rfloor$ and $\lceil 0.5 \times m \rceil$ respectively.

Let us illustrate this with an example. Let there be 6 classes and 100 instances in D^o_{tr}. We randomly partition the D^o_{tr} into cross-validation training set, T and validation set, V. Each of T and V has 50 instances. We randomly choose 3 classes as known classes and the remaining 3 classes fall into the unknown class. We remove the instances from unknown classes in the training set T. In the validation set, instances from the known classes as well unknown classes are present.

To optimize N parameters, we perform an N-dimensional grid search on the training set validation set tuple (T,V) and select the parameter value/s giving the best output on the validation set. *Accuracy* is used for evaluation of the performance.

C. COMPARING METHODS

Open set recognition and classification have been accomplished efficaciously by a number of works in the past few years. For comparative assessment of performance of our scheme, we have selected five methods which are briefly described next.

- *1-vs-set, [1]*: It is a baseline method in the field of open set recognition. The recommended version of "1-vs-all" is chosen for comparison.
- *WSVM, [3]*: This is possibly the best performing open set learner so far. But LETTER and MNIST datasets are run on the recommended values of $C = 2$, $\gamma = 2$,$\delta = 0.1$ and $C = 2$, $\gamma = 0.03125, \delta = 0.1$ respectively. For the remaining datasets, γ and C values are selected via two-dimensional grid search. As recommended in the paper, threshold probability, P is set to *0.5*openness*.
- *Multi-class probability of inclusion, PI-SVM [4]*: Probability of inclusion or into the class probability is the foundation of this work. '1-vs-rest' binary SVM with threshold probability, P value *0.5*openness* is considered for execution. Similar to [3], tuning of γ and C are required for this scheme. LETTER and MNIST datasets are run on the recommended values of $C = 2$, $\gamma = 2$,$\delta = 0.1$ and $C = 2$, $\gamma = 0.03125, \delta = 0.1$ respectively. For the remaining datasets, parameters are fixed through grid search.

  Nearest neighbor distance-ratio open set classifier by [12] has addressed open set recognition through a tweaked knn classifier. They proposed two slightly different schema which stand apart from each other in terms of performance. Since the interest of this work lies with classification through nearest neighborhood, we consider both the versions for comparison.
- *Nearest neighbor distance-ratio open set classifier (OSNN-CV)* : An instance is classified as unknown on getting a class mismatch between its two nearest neighbors. No user defined parameter is involved.
- *Nearest neighbor distance-ratio open set classifier (OSNN-NDR)*: The distance between two nearest neighbors belonging to different classes are noted for a test instance. If the ratio of the distance (nearer to farther) is sufficiently large, the instance is classified as unknown. The ratio of the two distances (nearer/ farther) is computed and compared with a threshold, namely T. For unknown class classification, T threshold range suggested by the authors is between *0.5* and *1*. Through parameter optimization, a single value is selected from $0.5, 0.55, \ldots, 1$ for each dataset.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

TABLE 2: Description of datasets. N, f and C denote the number of instances, features and total number of classes in order. c_k and c_u denote the cardinalities of the known and unknown classes respectively.

Datasets	N	f	C	c_k	c_u
Dermatology	358	34	6	3	1-3
Letter	20000	16	26	15	1-11
MNIST	70000	784	10	6	1-4
Reduced-MNIST	70000	79	10	6	1-4
Optdigits	5620	64	10	5	1-5
Penbased	10992	16	10	5	1-5
Segment	2310	19	7	3	1-4
Shuttle	58000	9	7	3	1-4
Texture	5500	40	11	5	1-6
Vehicle	846	18	4	2	1-2
Vowel	990	13	11	5	1-6

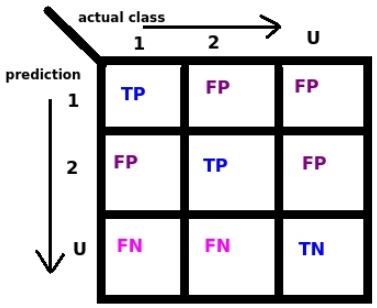


FIGURE 2: It depicts TP, TN, FP, FN for a toy scenario which has 2 known classes and an unknown class. Class 1 and class 2 It depicts TP, TN, FP, FN for a toy scenario which has 2 known classes and an unknown class. Class 1 and class 2 constitute the set of known classes and U denote the unknown class. The first two diagonal elements correspond to the correct predictions for class 1 and class 2 and belong to the TP set. The 3^{rd} diagonal cell corresponds to the correct predictions for the unknown class U and hence counted as TN. Remaining elements of row 1 and 2 corresponds to the FPs or false predictions into the known classes. For example, cell(2,1) counts the cases where the actual class is 1 but the prediction has been class 2. For cell(2,U) the actual class of the instances is unknown class U but class 2 is predicted. Non-diagonal elements of row 3 correspond to the cases where prediction as been made into the unknown class U but actual class is a known class (1 or 2).

- The proposed method: The proposed scheme requires tuning of the neighborhood k. The value of k is chosen via cross-validation on the training set.

D. EVALUATING INDICES

In this piece of work, we deal with learners which detect *unknown* class instances alongside the usual classification of instances into one of the known classes. *Accuracy*, *Average F_1 over known and unknown classes (AKUF$_1$)* and *Known class F_1* are employed to provide insight into known class classification as well as unknown class detection. Before going into the details, we describe a few notations.

The class of *known* classes (known or training classes taken together) is considered the positive class and the set of classes absent during training or the *unknown* class is dubbed as negative. Let the known classes set be, K = $\{1, 2, \ldots, c\}$ and the unknown class label be $c + 1$. A *true positive* prediction denotes that the classifier prediction is correct and the actual class is any one among $1, 2, \ldots, c$. In a similar fashion, a *true negative* is a correct prediction and the actual class is $c+1$, the unknown class. A *False positive* prediction is incorrect and the prediction is between 1 and c. There can be two cases of a false positive prediction — true class is the *unknown* class but the learner has misclassified into a *known* class. The other possible case is when the true class is some known class 1 (say) but the prediction has been made into some other known class 3 (say). *False negative* denotes that an instance from a *known* class has been incorrectly classified into the *unknown* class. The total and individual counts of *true positive, true negative, false positive* and *false negative* are represented as *TP, TN, FP* and *FN* respectively. An example is illustrated in Fig 2.

- *Accuracy*: For evaluating the classification performance of a learner, accuracy is the primary choice. It measures the fraction of correct predictions against the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

  Intricate details like individual class performance of a learner cannot be deduced from accuracy alone. The next metric is employed to address the same.

- *Average F_1 over known and unknown classes (AKUF$_1$)*: In order to address the limitation of the above and provide a better glimpse of the class performances, AKUF$_1$ is computed. F_1 is measured for a single class where the possible classes can be more than one. F_1 calculates the harmonic mean of precision and recall for the concerned class. Below, the F_1 calculation for the positive class is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

demonstrated.

$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN} \quad (9)$$
$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

In the context of open set recognition, classes are broadly classified into *known* and *unknown* and the two are equally significant. F_1 is individually calculated on the *known* (positive) class as well as the *unknown* (negative) class. Mean of the above two are computed as the AKUF$_1$ and interpreted to evaluate the overall performance of the learners. A similar metric has been used by [12].

- *Known class* F_1: This particular measure estimates the efficiency of the schemes in correct classification of the known class instances in a mixed bag of known and unknown instances.

VII. RESULTS AND DISCUSSION

This section of the paper is devoted to the summarization and comparative analysis of the experimental results. Before proceeding to the discussion, we would like to clarify the figurative layout. The empirical results are obtained with different openness values where the range of openness varies across datasets depending on the number of classes. For LETTER and MNIST datasets, we have set the known class and unknown cardinalities according the experimental protocol of [3]. For a proper presentation, we have provided *three* graphical layouts for each dataset, one each for three evaluating metrics, namely *Accuracy*, *Average* F_1 *over known and unknown classes* (AKUF$_1$) and *Known class* F_1. Results on AKUF$_1$ are presented in Figure 4 to Figure 14). Figures 15-25 and Figures 26-36 show the *Accuracy* and *Known class* F_1 plots respectively. Additionally, we have presented the summarized results in three tables 3, 4 and 5. Each table is dedicated to a metric and reports the number of best results obtained by each method on each dataset. A total of 50 scenarios or opennesses have arisen for the 10 datasets and the corresponding results are reported in the tables. In the following three paragraphs, we discuss the comparative performance of the methods on accuracy, AKUF_1, and Known class F_1 in order with reference to their corresponding plots and tables.

Accuracy is a primary choice when one has to evaluate a classifier. Table 3 records the number and percentage of best performance delivered by each of the comparing methods. Out of the 50 cases, the proposed method delivers best results on 39 scenarios (78%), followed by 6 (12%), 4 (8%) and 1 (2%) scenarios by WSVM, PI-SVM and 1-vs-Set respectively. On MNIST and Dermatology datasets, the proposed method achieves best result on 1 (25%) out of 4 opennesses and 1 (50%) out of 2 opennesses. On all other datasets, performance of the proposed method is better at more than 50% of the cases. For lower values of opennesses, the known class/es play a key role in determining accuracy value while the growing unknown class contributes more at higher opennesses. From the figures, it can be seen that the accuracy achieved by the proposed method is fairly constant across increasing openness. This indicates the robustness of the proposed scheme to variable opennesses. Figures 15 to 25 portray the graphical portrayal of accuracy performances delivered by the methods against increasing openness. These plots show the accuracy scores of the methods at different opennesses. Let us analyze Fig 15, which corresponds to the *Dermatology* dataset. At openness value 0.2257, performance of 1-vs-Set is best among the lot, for the remaining opennesses (openness values 0.2614 and 0.2929) the performance of proposed method is better than all the comparing methods. The betterment in performance by the proposed method is more at 0.2614 than at 0.2929.

AKUF_1 performance indicates a similar show. Figures 4 to 14 present the AKUF_1 performance of the methods over increasing openness. These figures plot the actual outputs given by the proposed and comparing methods. Let us analyze Fig. 4 (*Dermatology* dataset). It can be observed that the performance of the proposed method lies above all others at all three openness values. But the degree of improvement over the other methods is more pronounced at openness values 0.2257 and 0.2929. Similar analysis for all datasets can be be made by consulting the remaining figures. Table 4 shows the overall statistics of best AKUF_1 performance by the methods. The proposed method delivers the best performance on 50% or more cases for all but one dataset.Out of the 50 cases, the proposed method wins in 40 cases (80%) followed by 6 (12%) and 4 (8%) cases by WSVM and PI-SVM respectively. These figures indicate the capability of the proposed scheme in correctly predicting the known classes as well as the unknown class.

Now, we analyze the relative capability of the proposed method to correctly predict the known class instances or *Known class* F_1. In practical scenario, this metric holds significance since its mimics the real world where we predict known things in a known and unknown world. Table 5 records the data of best outcomes on each dataset and its respective opennesses. Similar to the previous two measures, the proposed method gets the major share 76% (38 out of 50) best outcomes. Remaining 24% is shared by WSVM (8%, 4 out of 50), PI-SVM (10%, 5 out 50) and OSNN-CV (6%, 3 out of 5). Detailed known class F_1 values are available in Figures 26-36. Known class F_1 performance on *Dermatology* dataset is shown in Figure 26. At openness values 0.2257 and 0.2614, the proposed method performs best. PI-SVM scores best on the remaining openness (0.2929).

Comparative results presented in the above three paragraphs indicate the efficaciousness of the proposed scheme in both known and unknown aspects of open set learning. The proposed method maintains its superior performance on datasets with lesser number of classes (*Dermatology*,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

TABLE 3: Performance on Accuracy. The table gives the summary of the best performances obtained by each method on each dataset.

Dataset (Openness)	Methods											
	Proposed Method		1-vs-Set		WSVM		PI-SVM		OSNN-CV		OSNN-NDR	
	# of Wins	Win%	# of Wins	Win%	# of Wins	Win%	#of Wins	Win%	# of Wins	Win%	#of Wins	Win%
Dermatology (3)	**2**	**66.67%**	1	33.33%	0	0	0	0	0	0	0	0
Letter (11)	**8**	**72.72%%**	0	0	3	27.27%	0	0	0	0	0	0
MNIST (4)	1	25%	0	0	**2**	**50%**	1	25%	0	0	0	0
Optdigits (5)	**4**	**80%**	0	0	0	0	1	20%	0	0	0	0
Penbased (5)	**4**	**80%**	0	0	0	0	1	20%	0	0	0	0
Segment (4)	**4**	**100%**	0	0	0	0	0	0	0	0	0	0
Shuttle (4)	**4**	**100%**	0	0	0	0	0	0	0	0	0	0
Texture (6)	**5**	**83.33%**	0	0	1	16.67%	0	0	0	0	0	0
Vehicle (2)	**1**	**50%**	0	0	0	0	**1**	**50%**	0	0	0	0
Vowel (6)	**6**	**100%**	0	0	0	0	0	0	0	0	0	0
Total	**39/50**	**78%**	1	2%	6/50	12%	4/50	8%	0	0	0	0

TABLE 4: Performance on Average F_1 over known and unknown classes (AKUF_1). The table gives the summary of the best performances obtained by each method on each dataset.

Dataset (Openness)	Methods											
	Proposed Method		1-vs-Set		WSVM		PI-SVM		OSNN-CV		OSNN-NDR	
	# of Wins	Win%	# of Wins	Win%	# of Wins	Win%	#of Wins	Win%	# of Wins	Win%	#of Wins	Win%
Dermatology (3)	**3**	**100%**	0	0	0	0	0	0	0	0	0	0
Letter (11)	**9**	**81.81%%**	0	0	2	18.18%	0	0	0	0	0	0
MNIST (4)	0	0%	0	0	**3**	**75%**	1	25%	0	0	0	0
Optdigits (5)	**4**	**80%**	0	0	0	0	1	20%	0	0	0	0
Penbased (5)	**4**	**80%**	0	0	0	0	1	20%	0	0	0	0
Segment (4)	**4**	**100%**	0	0	0	0	0	0	0	0	0	0
Shuttle (4)	**2**	**50%**	0	0	1	25%	1	25%	0	0	0	0
Texture (6)	**6**	**100%**	0	0	0	0	0	0	0	0	0	0
Vehicle (2)	**2**	**100%**	0	0	0	0	0	0	0	0	0	0
Vowel (6)	**6**	**100%**	0	0	0	0	0	0	0	0	0	0
Total	**40/50**	**80%**	0	0	6/50	12%	4/50	8%	0	0	0	0

TABLE 5: Performance on Known class F_1. The table gives the summary of the best performances obtained by each method on each dataset.

Dataset (Openness)	Methods											
	Proposed Method		1-vs-Set		WSVM		PI-SVM		OSNN-CV		OSNN-NDR	
	# of Wins	Win%	# of Wins	Win%	# of Wins	Win%	#of Wins	Win%	# of Wins	Win%	#of Wins	Win%
Dermatology (3)	**2**	**66.67%**	0	0	0	0	1	33.33%	0	0	0	0
Letter (11)	**7**	**63.63%%**	0	0	2	11.11%	0	0	2	18.18%	0	0
MNIST (4)	1	25%	0	0	**2**	**50%**	0	0	1	25%	0	0
Optdigits (5)	**4**	**80%**	0	0	0	0	1	20%	0	0	0	0
Penbased (5)	**4**	**80%**	0	0	0	0	1	20%	0	0	0	0
Segment (4)	**4**	**100%**	0	0	0	0	0	0	0	0	0	0
Shuttle (4)	**4**	**100%**	0	0	0	0	0	0	0	0	0	0
Texture (6)	**6**	**100%**	0	0	0	0	0	0	0	0	0	0
Vehicle (2)	0	0	0	0	0	0	**2**	**100%**	0	0	0	0
Vowel (6)	**6**	**100%**	0	0	0	0	0	0	0	0	0	0
Total	**38/50**	**76%**	0	0	4/50	8%	5/50	10%	3/50	6%	0	0

Vehicle) as well as on datasets with large number of classes (*LETTER*, *Vowel*, *Texture*). The intrinsic multi-class framework of the proposed scheme accounts for this robustness.

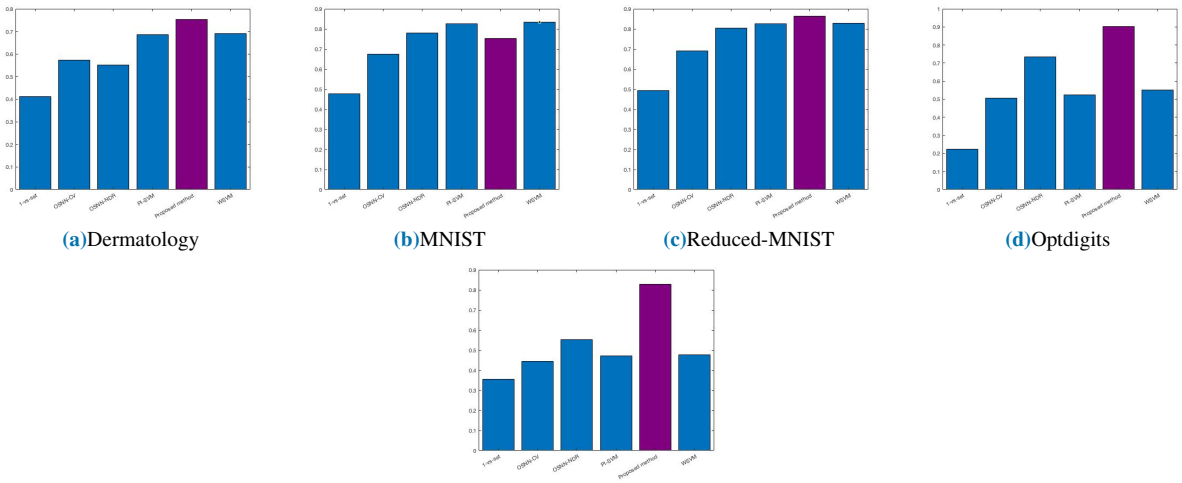
The performance of the proposed method on MNIST dataset is not as good as compared to a couple of methods (namely WSVM and PI-SVM). Moreover, it also shows a deviation from it's own (proposed method's) performance on the remaining datasets. We investigated the loss of performance on MNIST dataset and our findings direct to the high-dimensionality of this dataset. Our method is based on RkNN principles where distance and neighborhood relations are the only information that we cultivate for classification. Our method suffers from curse of dimensionality at 784 features and failed to perform as competently as on the remaining datasets. To validate our findings, we have generated outputs on a reduced version of MNIST dataset. The Reduced-MNIST version is obtained by extracting the top features which covers 90% feature variance. Reduced MNIST dataset has 79 features. Fig 7 shows the AKUF_1 performance of proposed and comparing methods on Reduced MNIST. It shows that the performance of the proposed method is better than that of all others. The results are also superior to that of the best performing methods (WSVM and PI-SVM) on regular MNIST (with all features) of 784 features (Refer to Figure 6 (for MNIST) and 7 (for Reduced MNIST)). Figures 18 and 29 show the accuracy and known class F_1 results of these experiments. The results are in congruence with AKUF_1 performance.

A. REPORTING AVERAGE (OVER ALL OPENNESSES OF A DATASET) AKUF_1 RESULTS OF FIVE DATASETS

For five datasets (Dermatology, MNIST Reduced-MNIST, Optdigits and Segment), we calculate the average of AKUF_1 scores over the various openness values (of each dataset). In

Figure 3, we plot the average $AKUF_1$ results of the proposed method and the five competing methods. The results indicate the certain superiority of the proposed method over all five comparing methods (including neighborhood based openset classifiers OSNN-CV and OSNN-NDR) in all datasets except MNIST.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access



(a) Dermatology **(b)**MNIST **(c)**Reduced-MNIST **(d)**Optdigits

(e) Segment

FIGURE 3: The results indicate the certain superiority of the proposed method over all competing methods on four out of five datasets. On MNIST dataset, the proposed method suffers from the issue of high dimensionality of features. The enhanced performance of the proposed method on Reduced-MNIST (with reduced feature set) dataset affirms this fact.

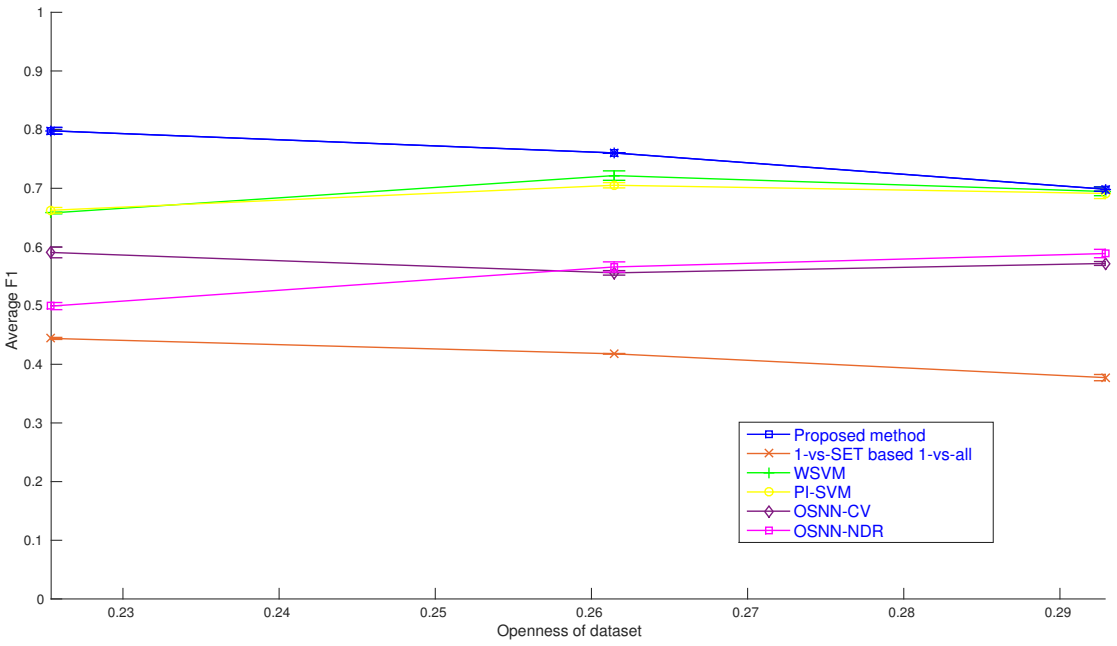


FIGURE 4: AKUF_1 results on *Dermatology* on *three* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

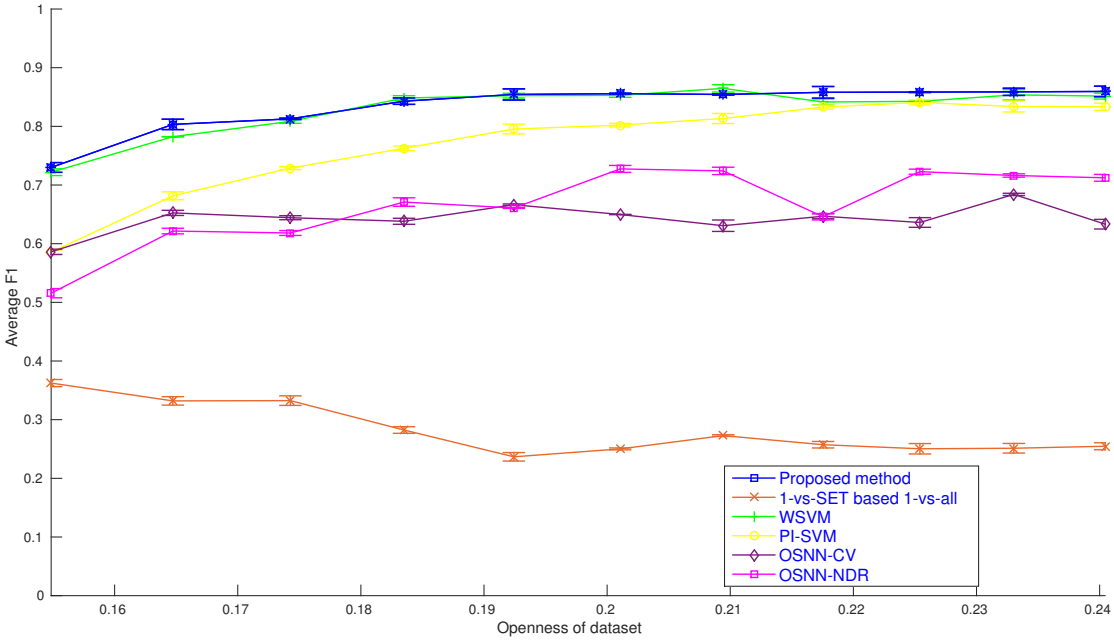


FIGURE 5: AKUF_1 results on *Letter* on *eleven* openness values.

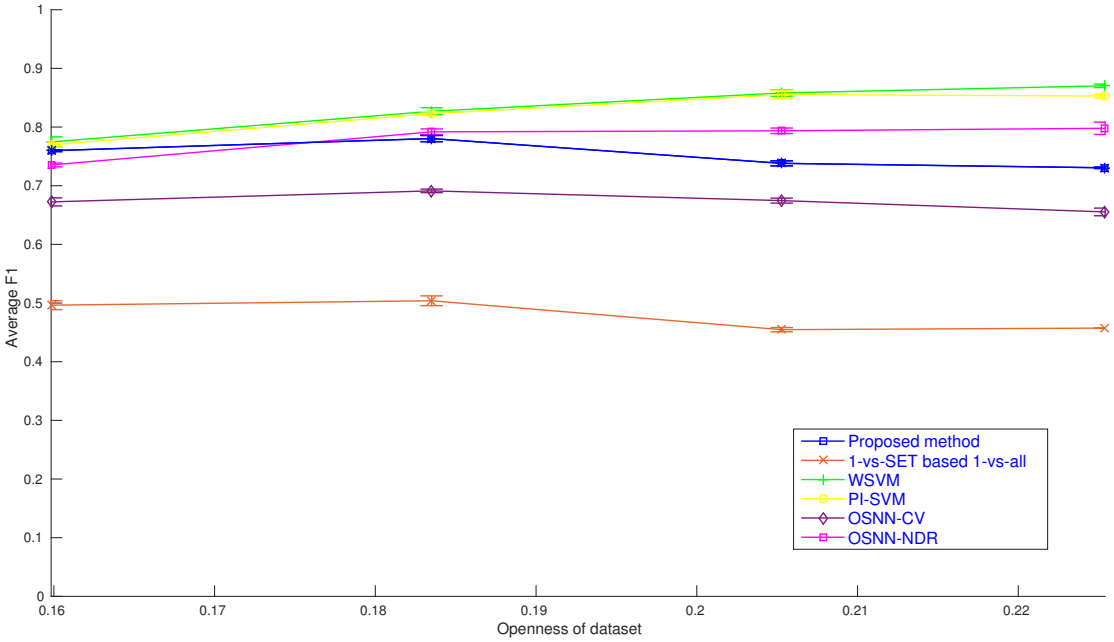


FIGURE 6: AKUF_1 results on *MNIST* on *four* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

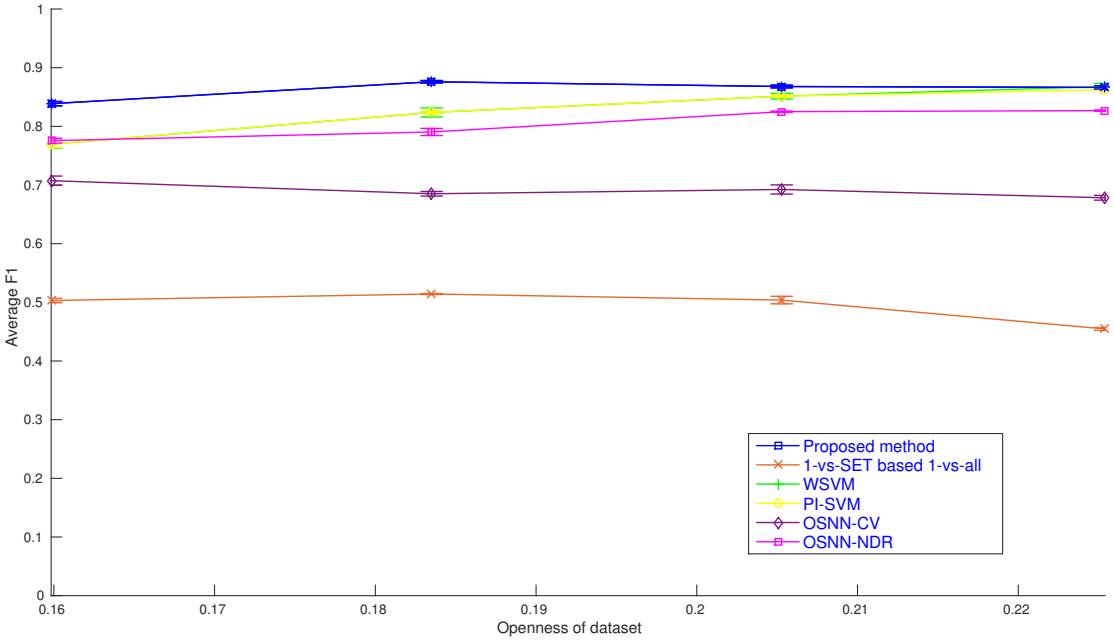


FIGURE 7: AKUF_1 results on *Reduced-MNIST* on *four* openness values.

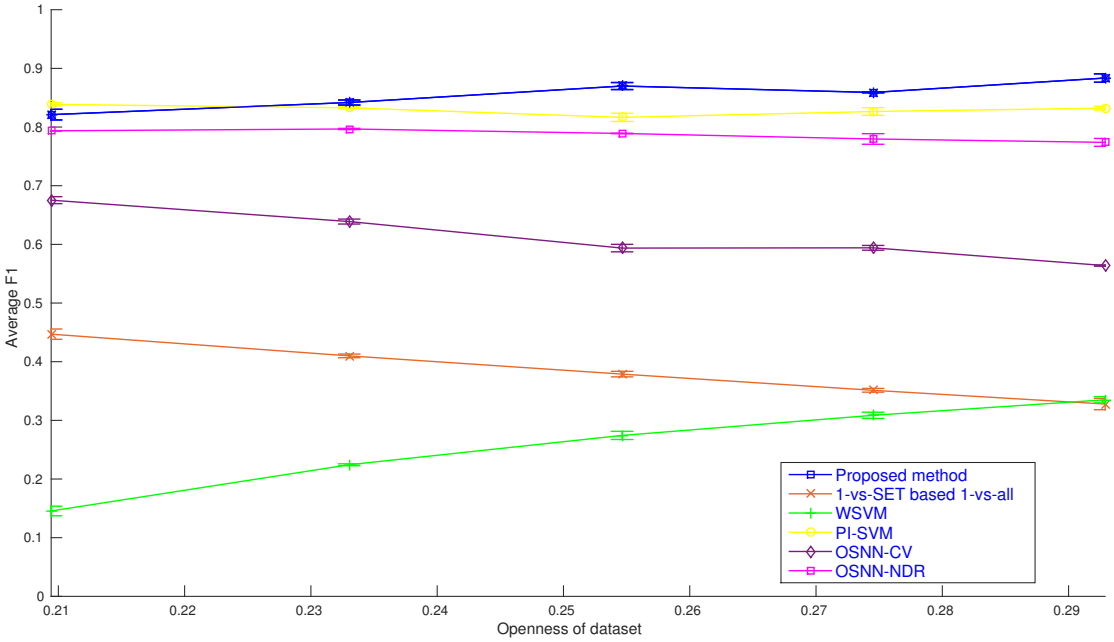


FIGURE 8: AKUF_1 results on *Optdigits* on *five* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

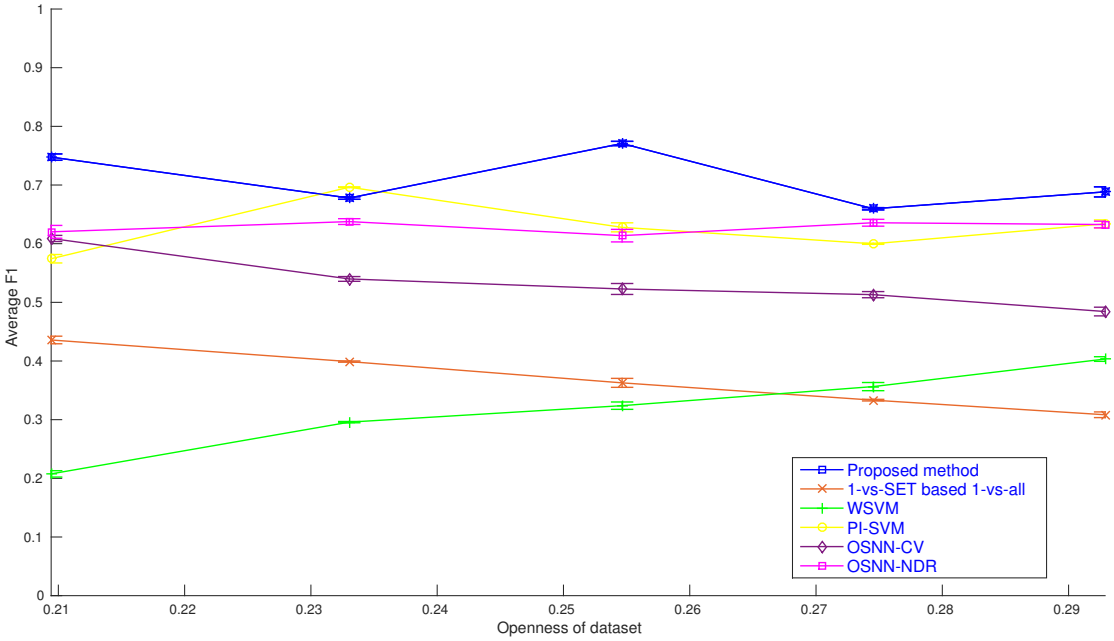


FIGURE 9: AKUF_1 results on *Penbased* on *five* openness values.

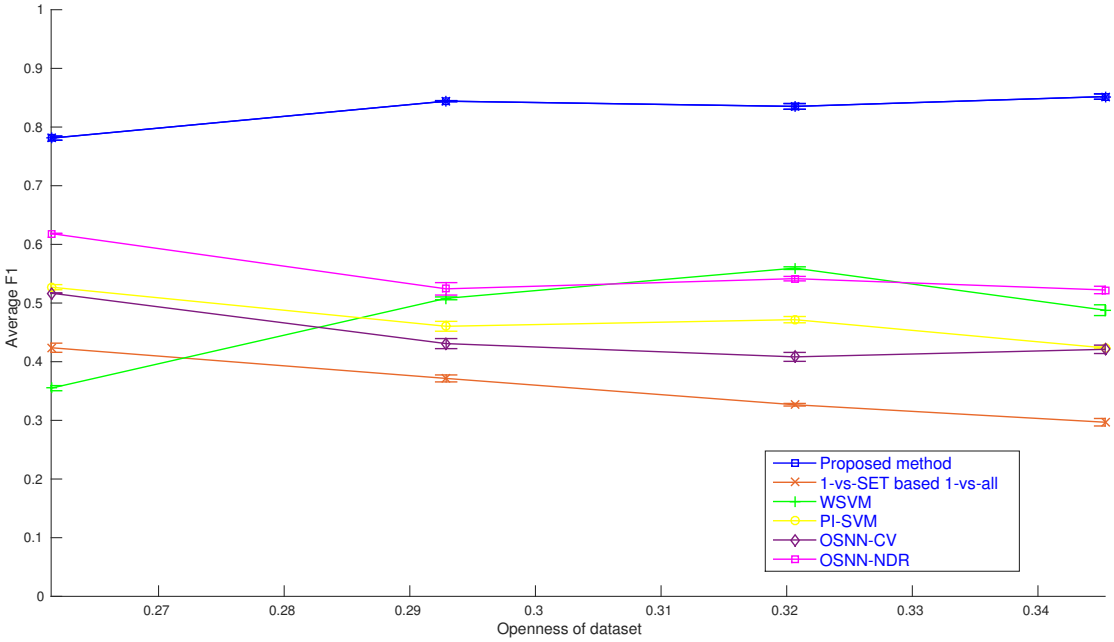


FIGURE 10: AKUF_1 results on *Segment* on *four* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

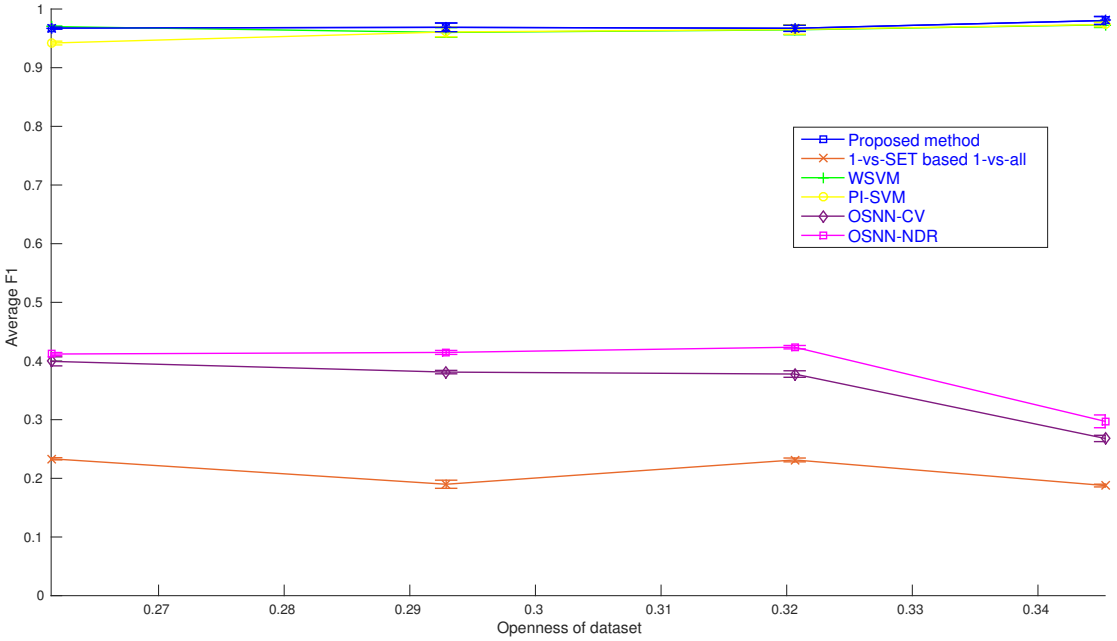


FIGURE 11: AKUF_1 results on *Shuttle* on *four* openness values.

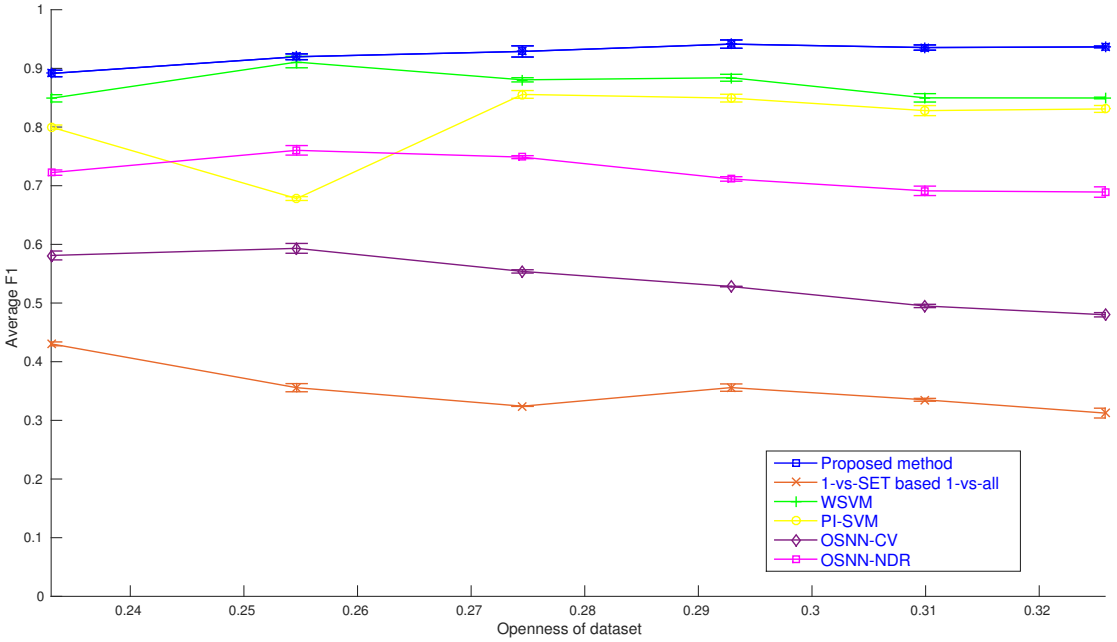


FIGURE 12: AKUF_1 results on *Texture* on *six* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

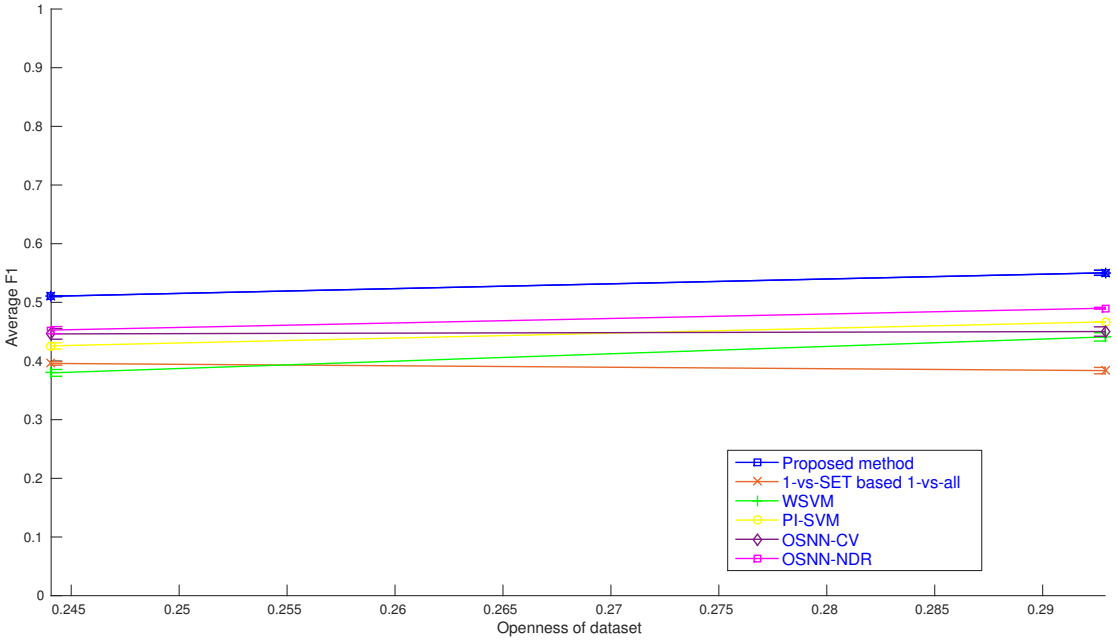


FIGURE 13: AKUF_1 results on *Vehicle* on *two* openness values.

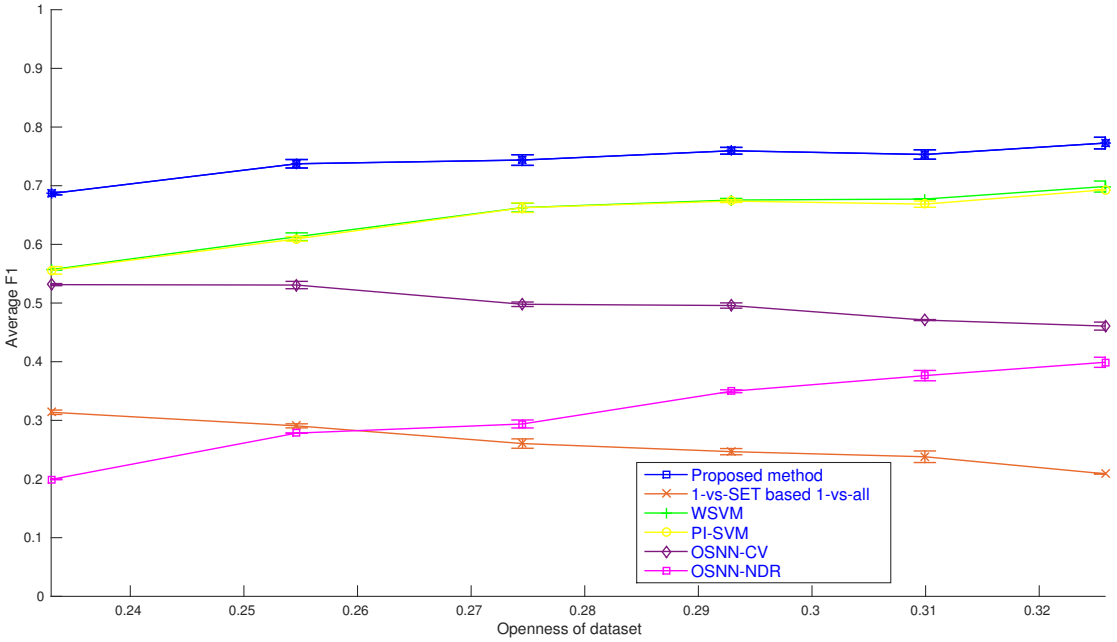


FIGURE 14: AKUF_1 results on *Vowel* on *six* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

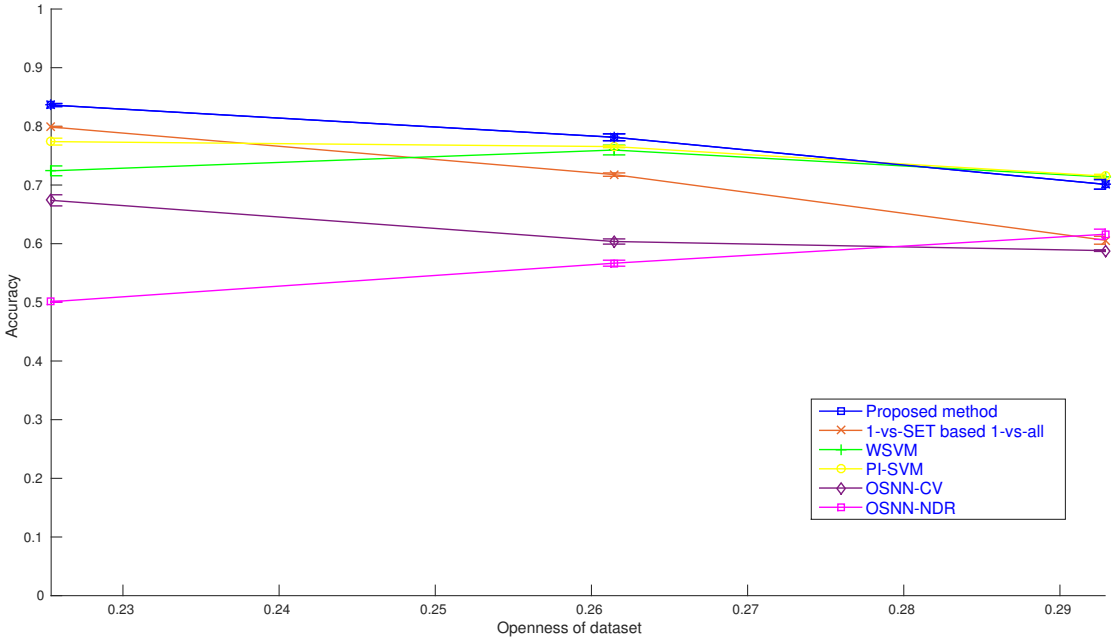


FIGURE 15: Accuracy results on *Dermatology* on *three* openness values.

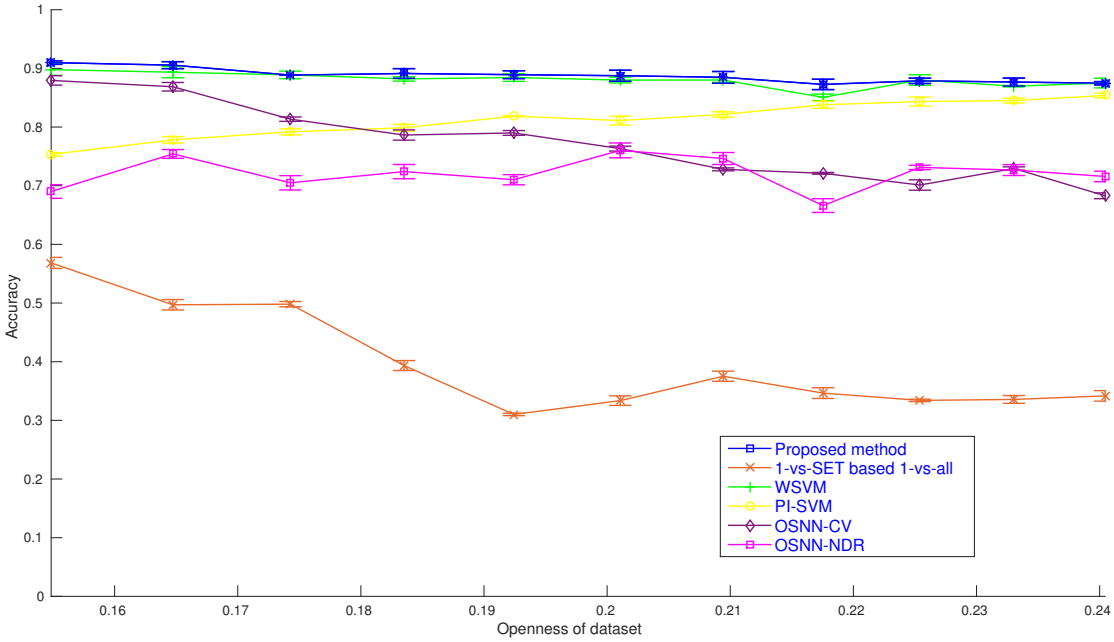


FIGURE 16: Accuracy results on *Letter* on *eleven* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

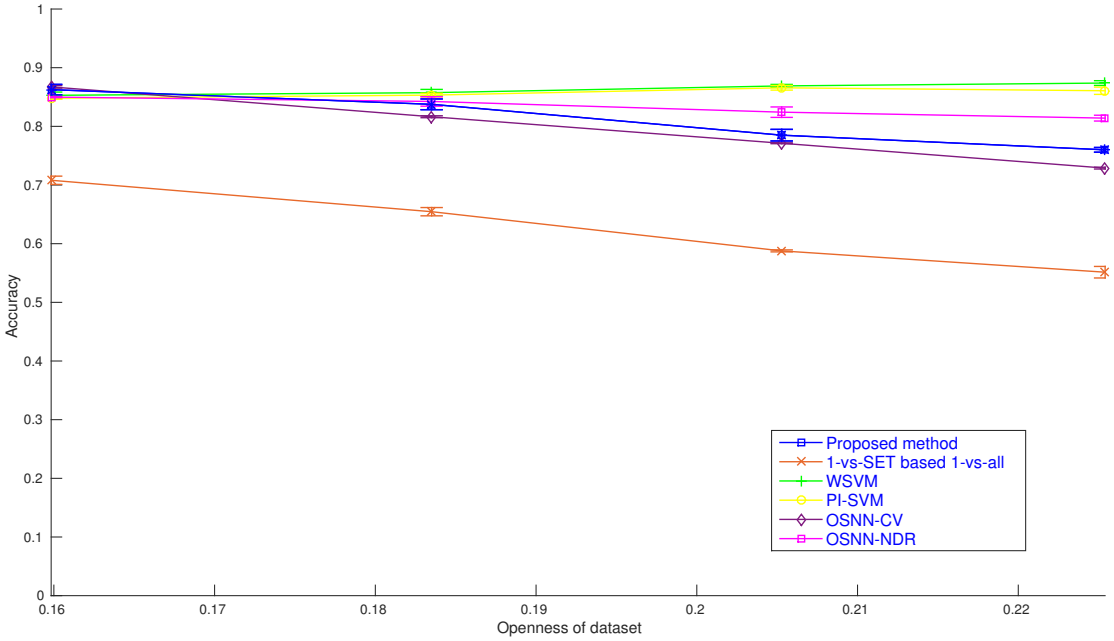


FIGURE 17: Accuracy results on *MNIST* on *four* openness values.

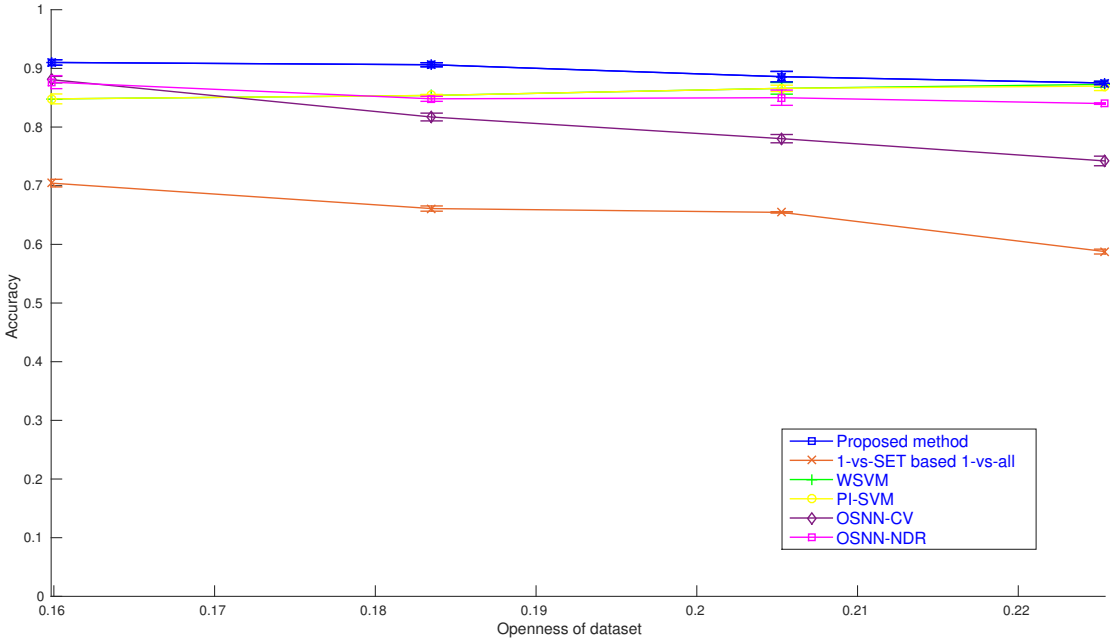


FIGURE 18: Accuracy results on *Reduced-MNIST* on *four* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

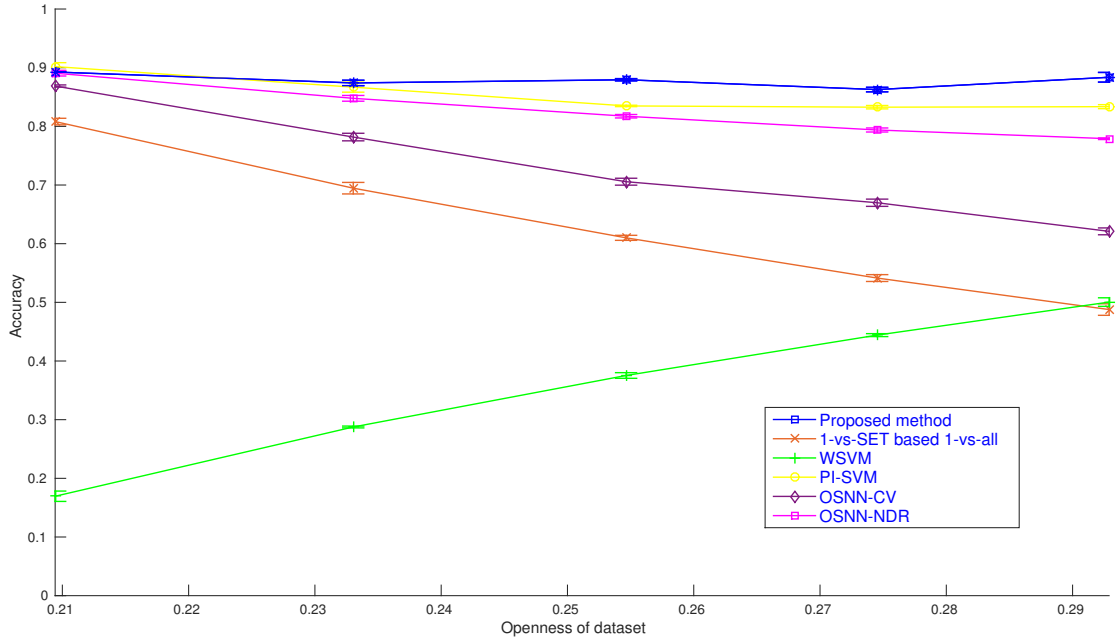


FIGURE 19: Accuracy results on *Optdigits* on *five* openness values.

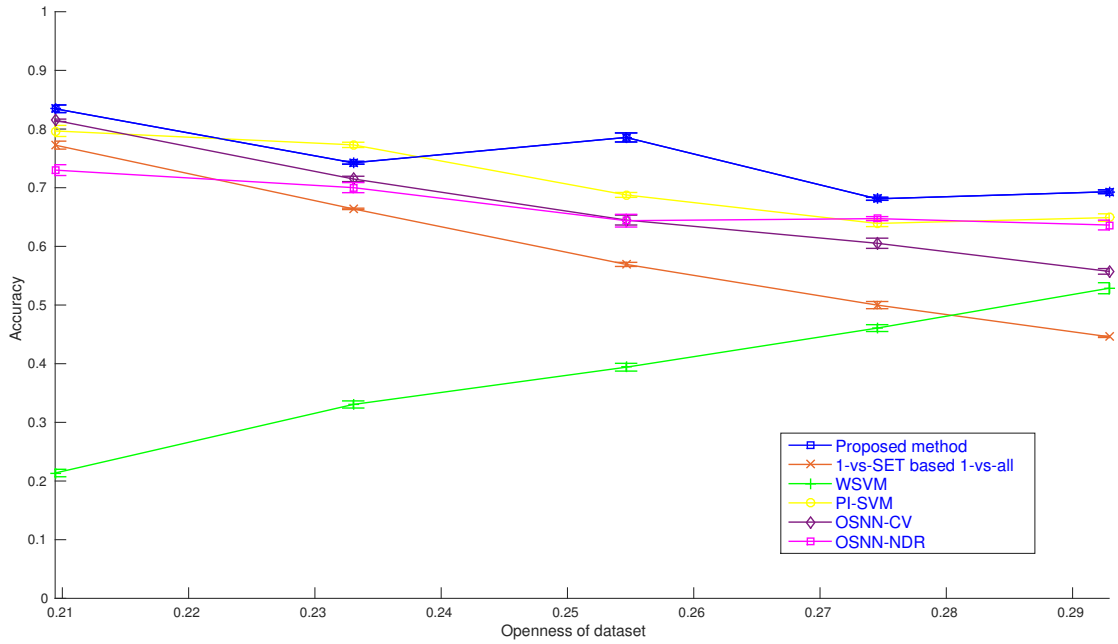


FIGURE 20: Accuracy results on *Penbased* on *five* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

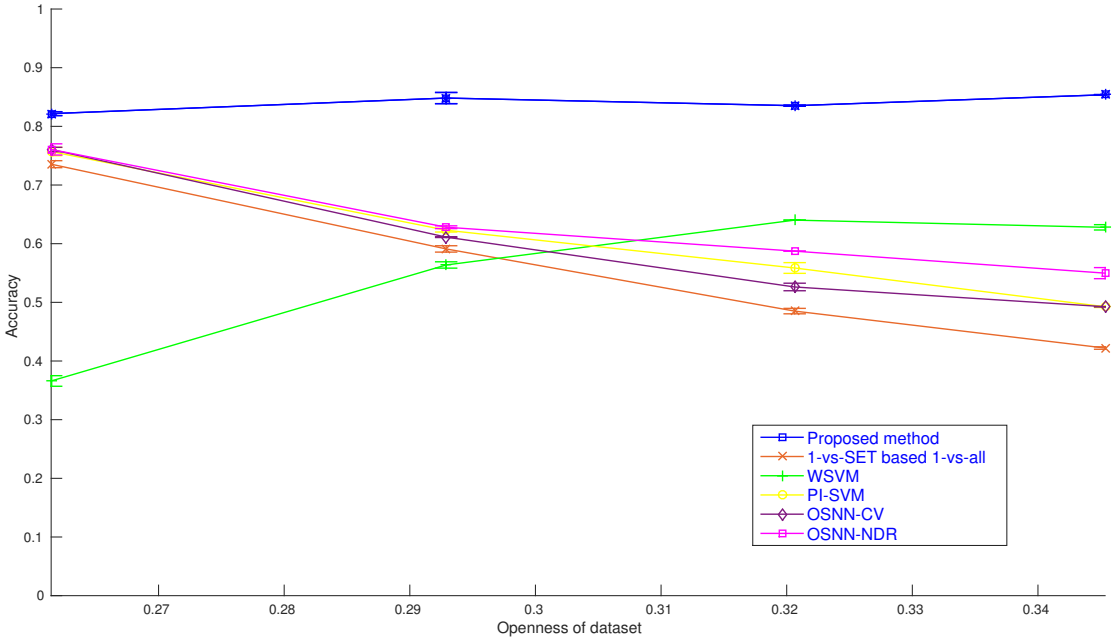


FIGURE 21: Accuracy results on *Segment* on *four* openness values.

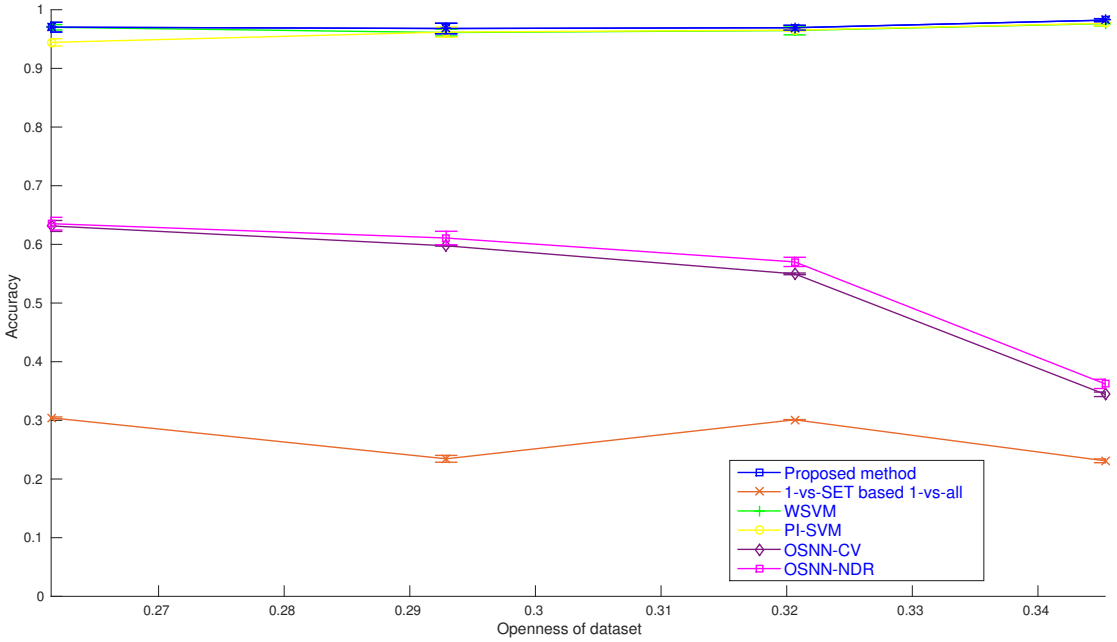


FIGURE 22: Accuracy results on *Shuttle* on *four* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

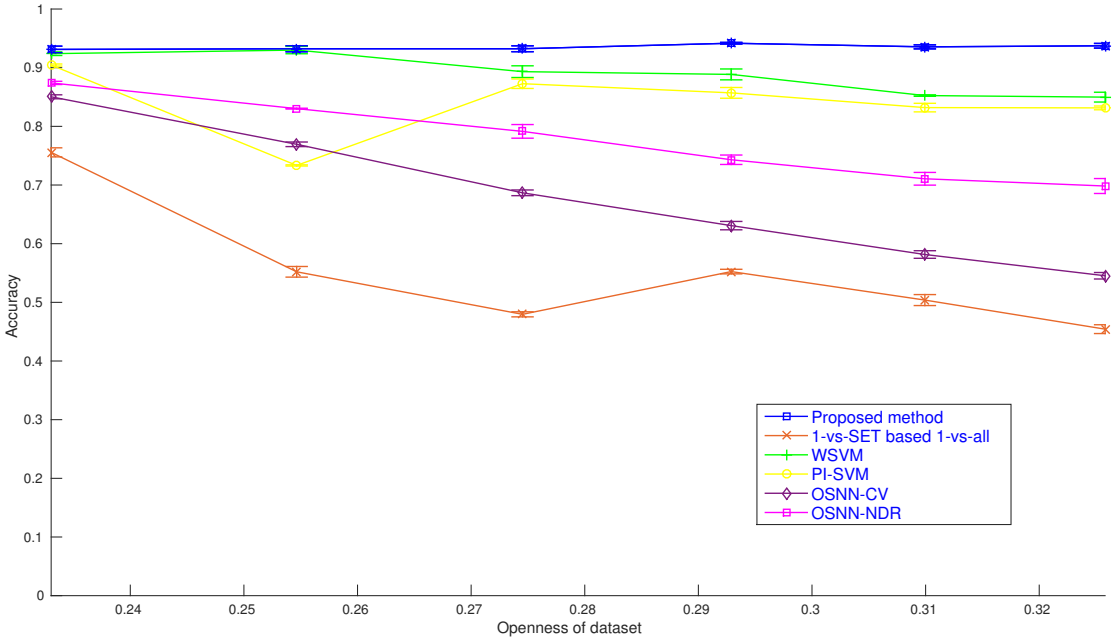


FIGURE 23: Accuracy results on *Texture* on *six* openness values.

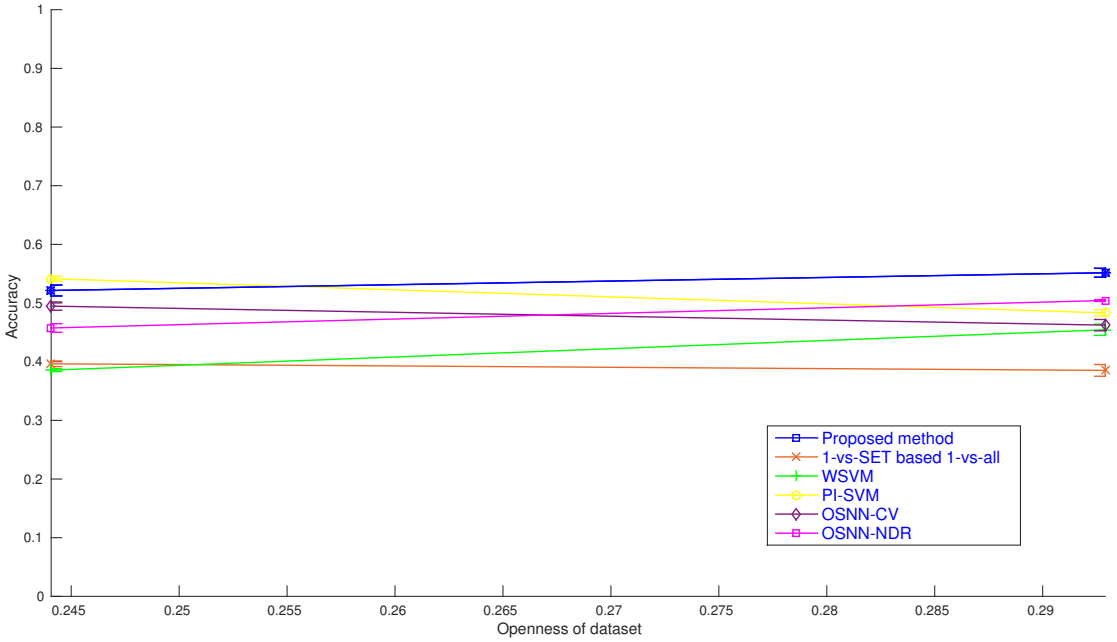


FIGURE 24: Accuracy results on *Vehicle* on *two* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

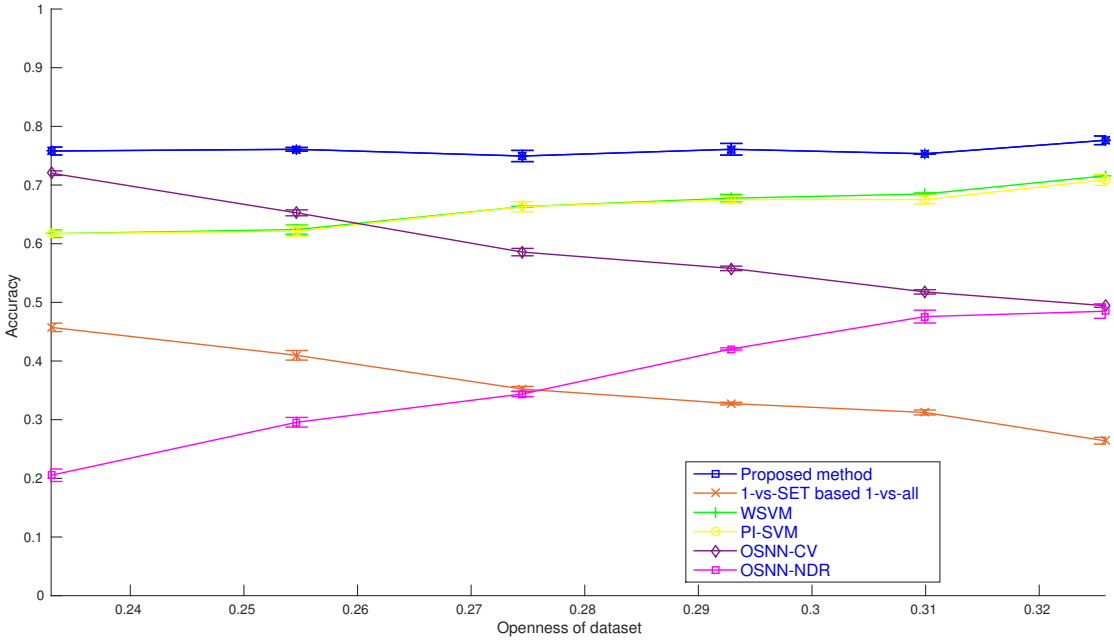


FIGURE 25: Accuracy results on *Vowel* on *six* openness values.

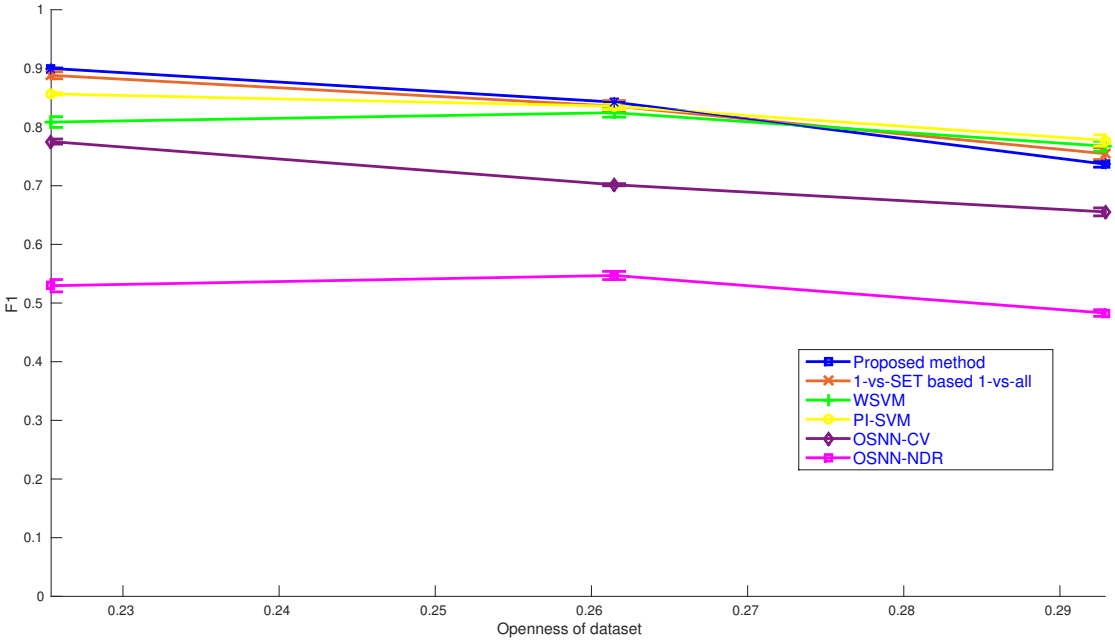


FIGURE 26: F_1 results on *Dermatology* on *three* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

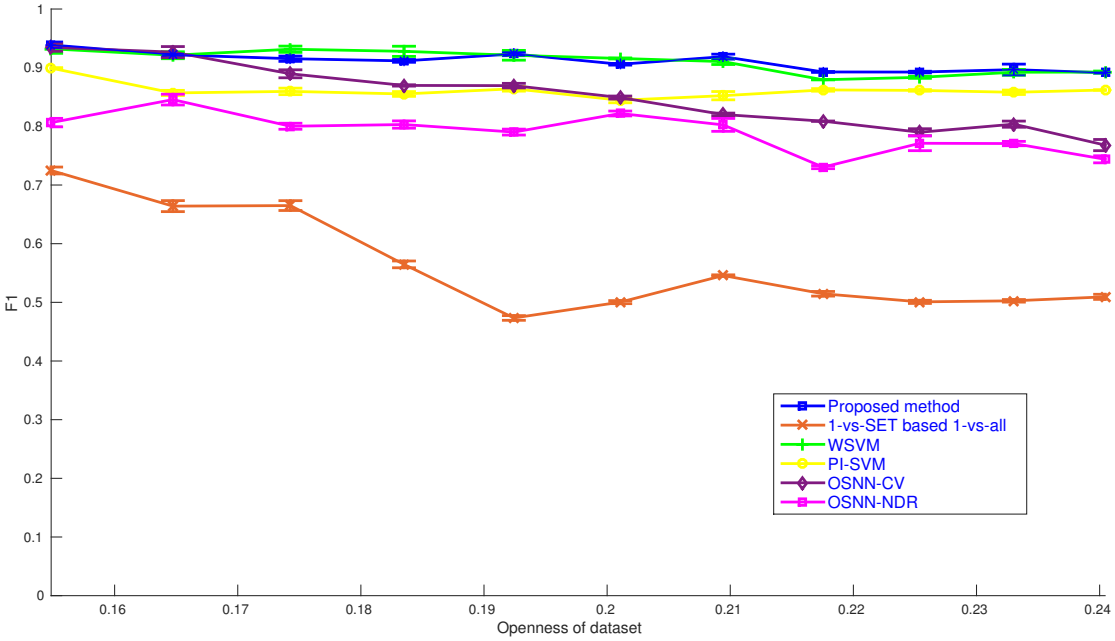


FIGURE 27: F_1 results on *Letter* on *eleven* openness values.

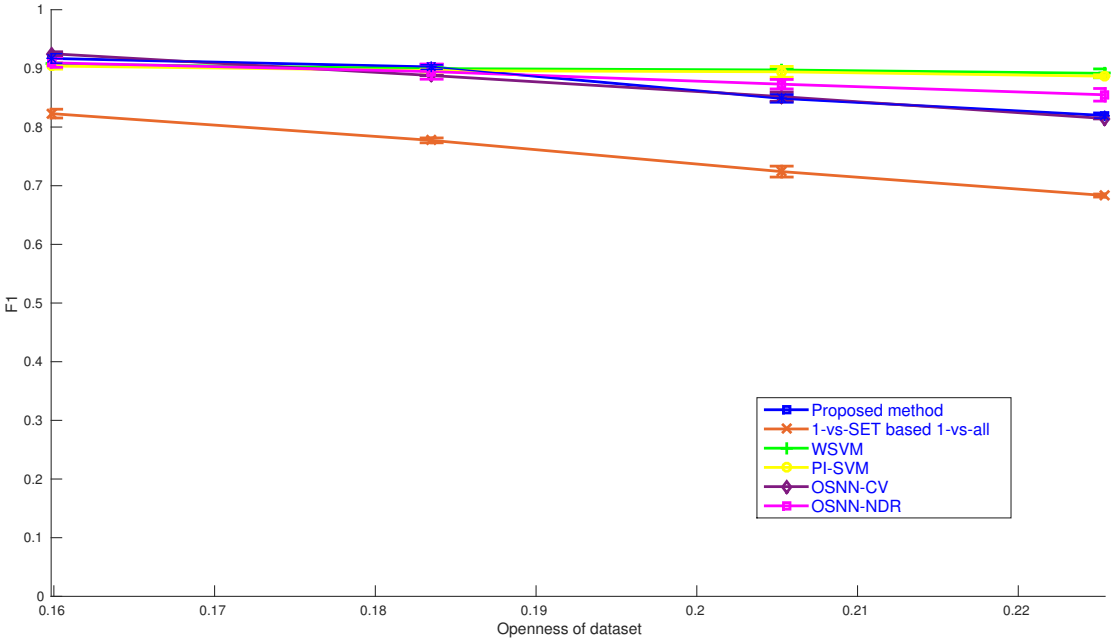


FIGURE 28: F_1 results on *MNIST* on *four* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

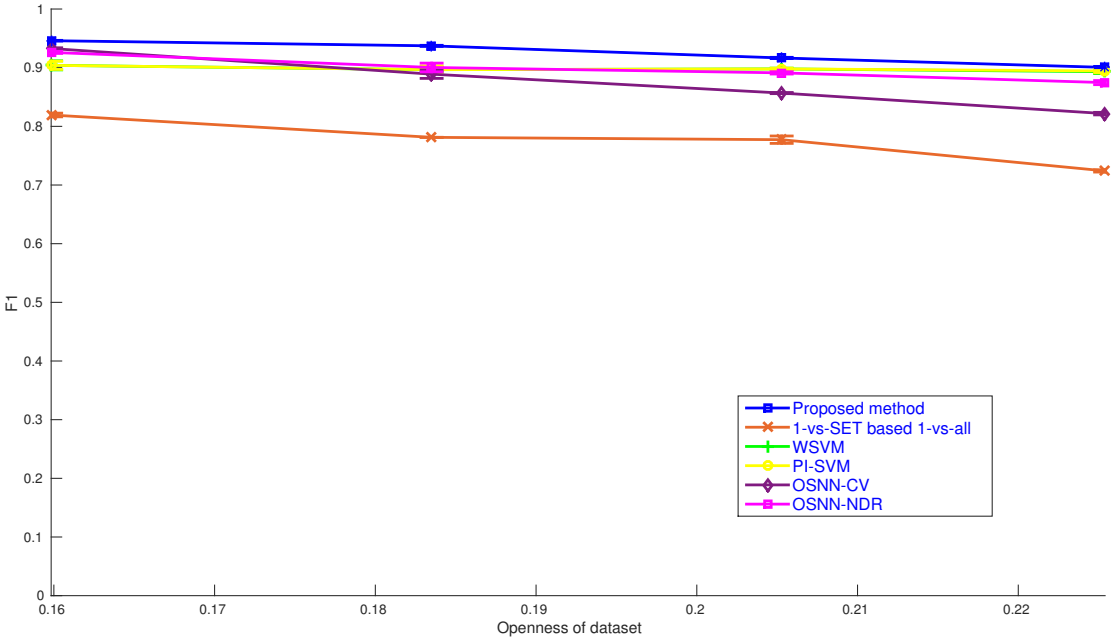


FIGURE 29: F_1 results on *Reduced-MNIST* on *four* openness values.

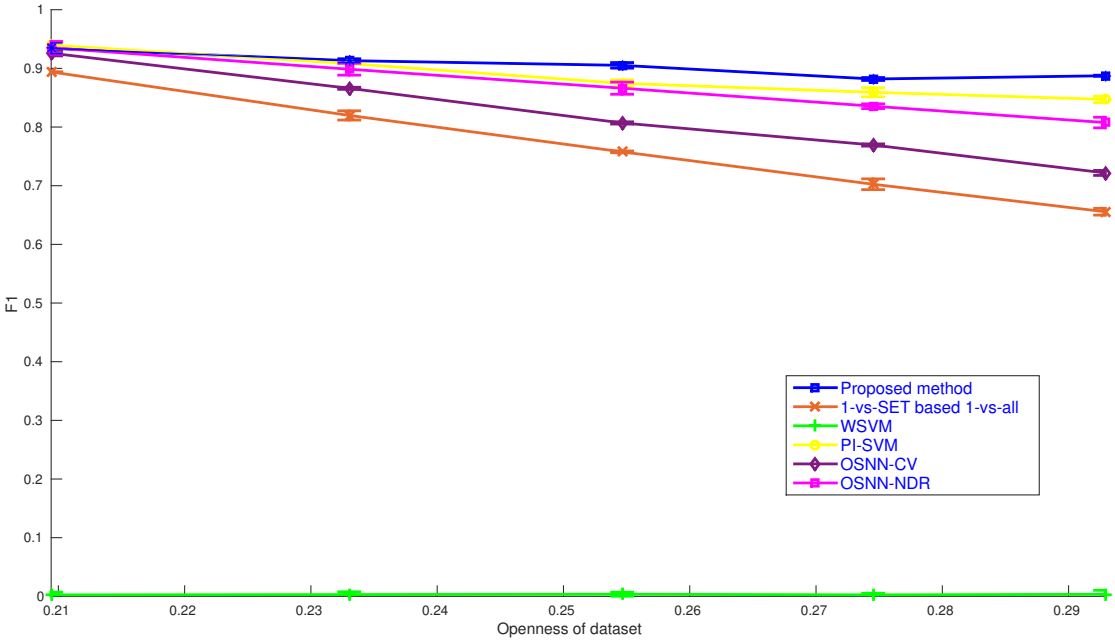


FIGURE 30: F_1 results on *Optdigits* on *five* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

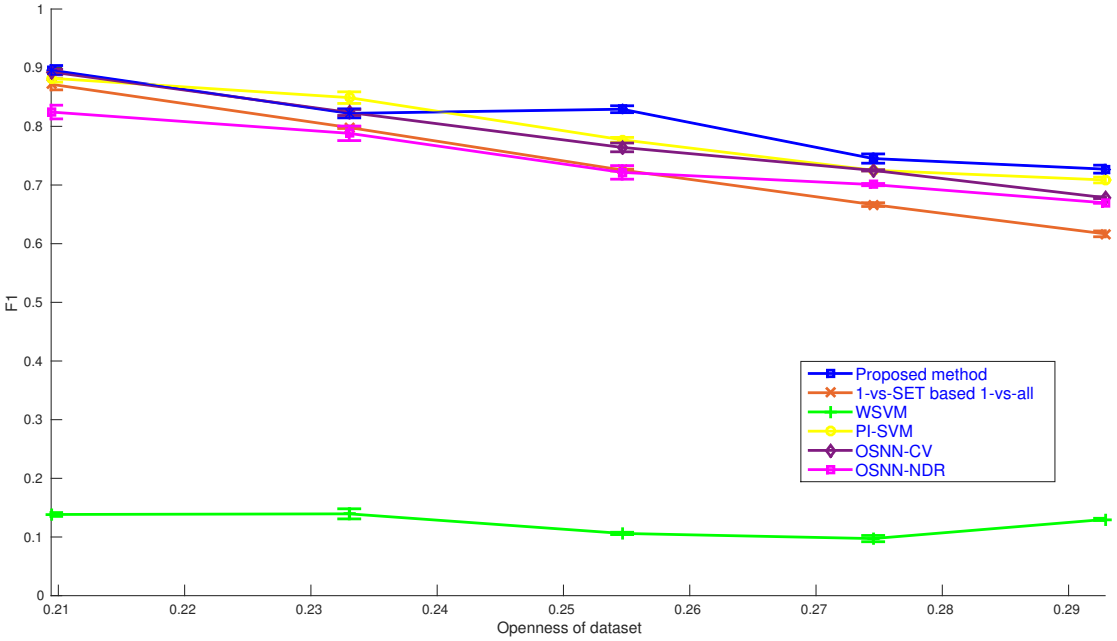


FIGURE 31: F_1 results on *Penbased* on *five* openness values.

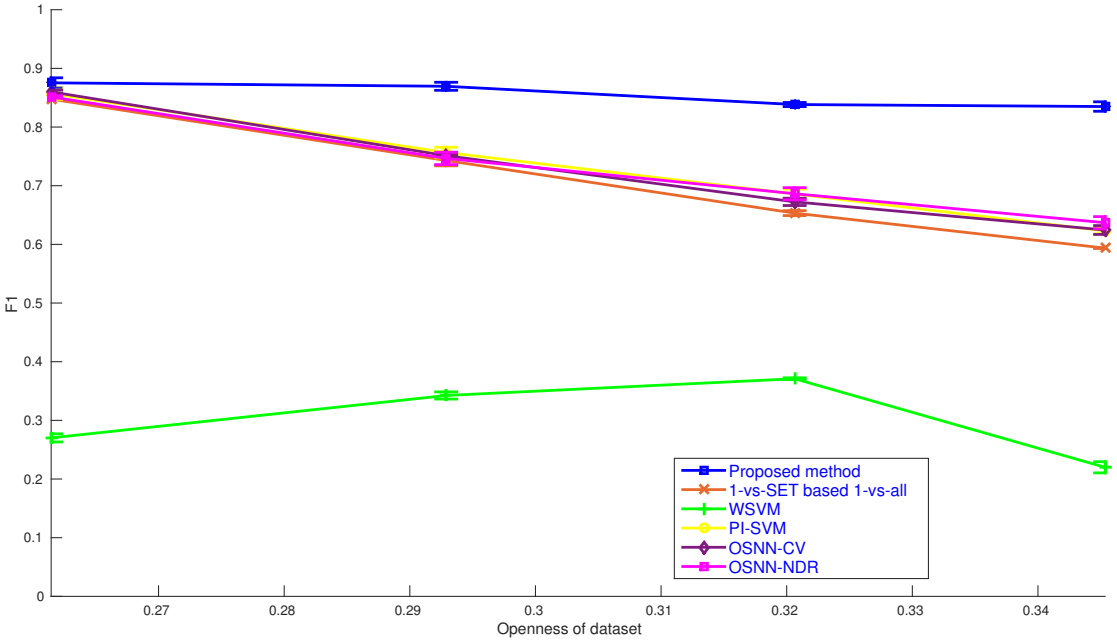


FIGURE 32: F_1 results on *Segment* on *four* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

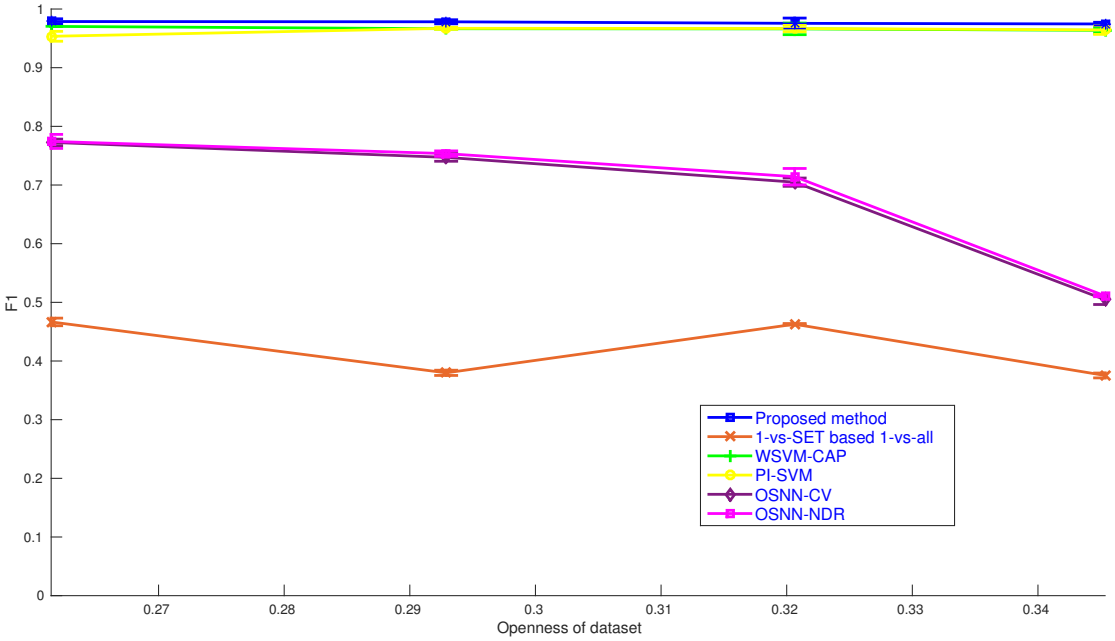


FIGURE 33: F_1 results on *Shuttle* on *four* openness values.

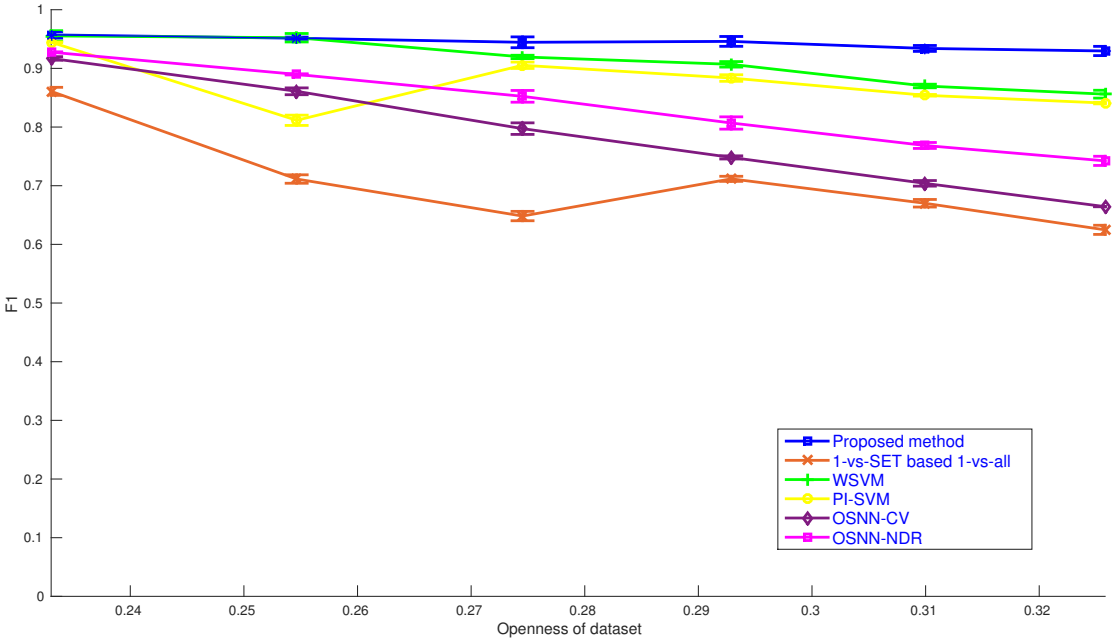


FIGURE 34: F_1 results on *Texture* on *six* openness values.

B. LIMITATIONS OF THE PROPOSED SCHEME

The proposed work deals with RkNN in which distance and neighborhood relation is the only information that is interpreted. Like any other distance-based scheme, our method suffers from the curse of dimensionality at higher dimensions. The same phenomenon was observed for the original MNIST dataset with 784 features. To curb this problem, we suggest a reduction in feature dimension of a dataset with $\geq$ 100 features through feature extraction or selection before proceeding with the RkNN-based learning and classification. The improvement in performance on Reduced-MNIST dataset (Figures 6, 17, 28) over original MNIST dataset (Figures 5, 16, 27) manifests the same.

VIII. EXPERIMENT ON PARAMETER TUNING

On *four* datasets, namely, *Dermatology, Vehicle, Segment and Vowel*, we have conducted a parameter tuning experiment. Neighborhood size 'k' is the only tunable parameter of our scheme. From our detailed experimental study and analysis, we have seen that a k value in the range 2, 3, 4, 5, 6 works well for all the datasets that we have used. Accordingly, we have reported the accuracy results of the four mentioned datasets across these five k values. Figures 37-40 shows the same. It is interesting to note that a single 'k' value may not work well on a dataset. So, it is advisable to tune the k value across different opennesses of a single dataset. The detailed procedure for parameter optimization is given Section VI. B

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

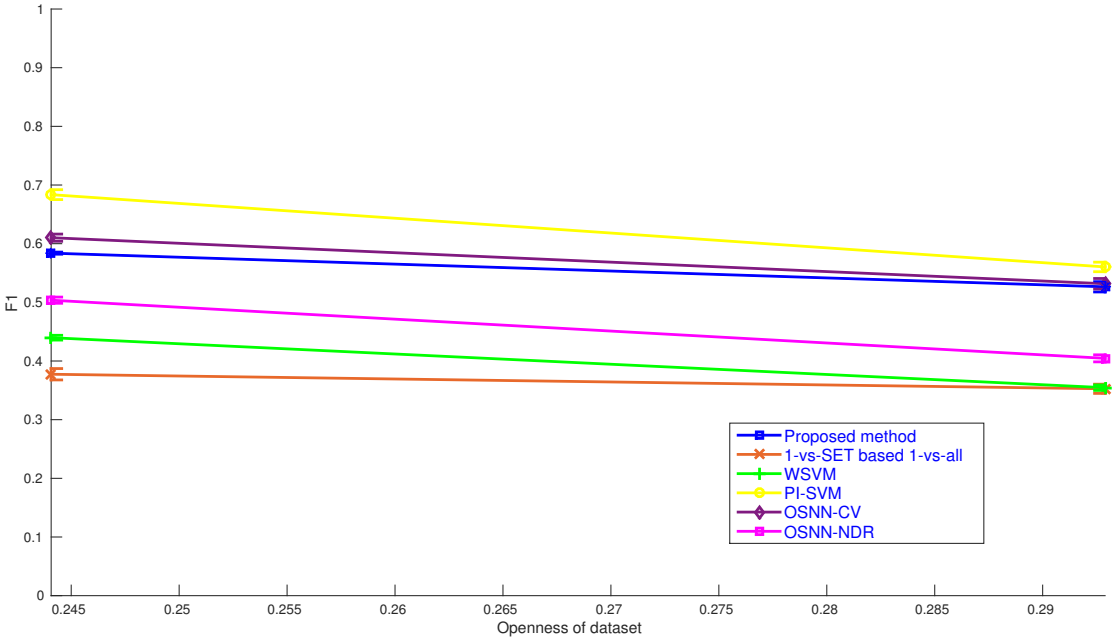


FIGURE 35: F_1 results on *Vehicle* on *two* openness values.

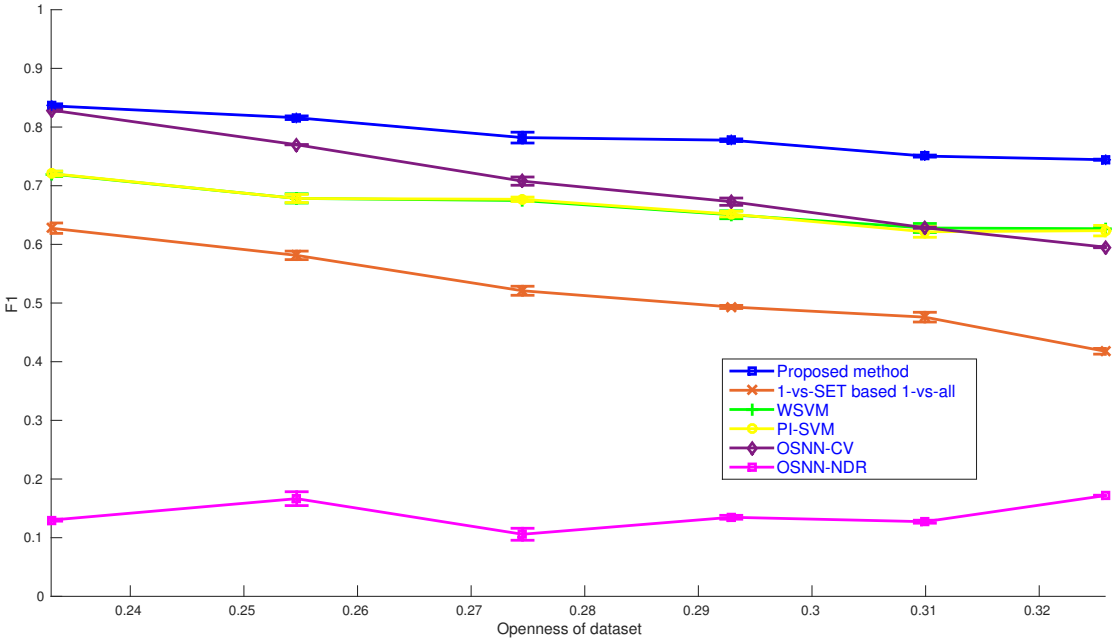


FIGURE 36: F_1 results on *Vowel* on *six* openness values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

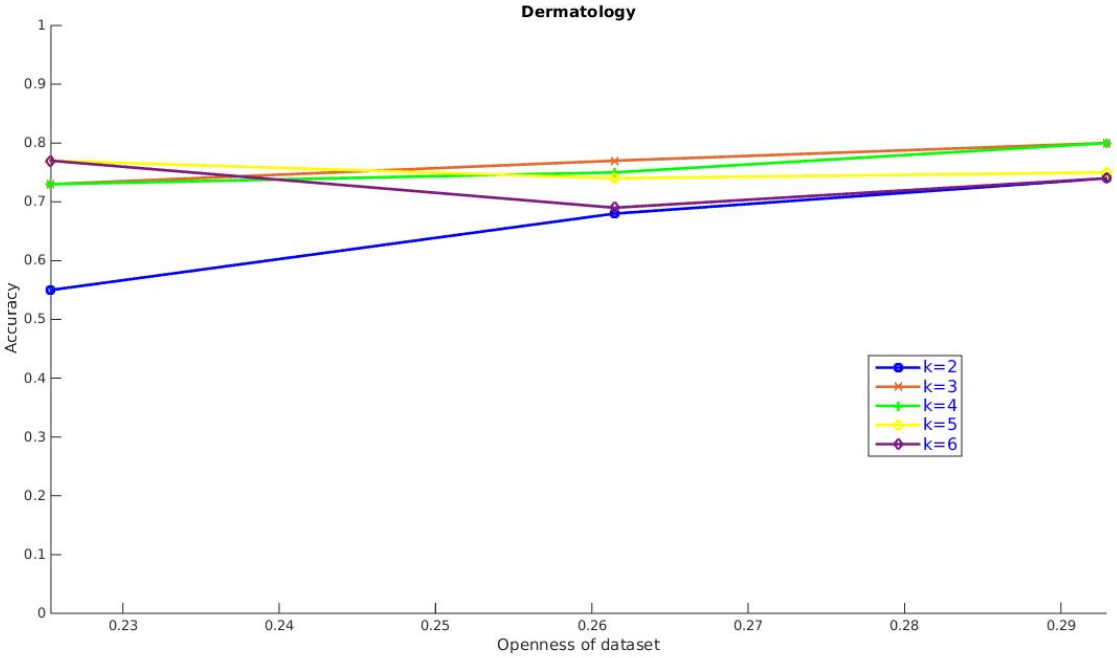


FIGURE 37: $Accuracy$ results on $Dermatology$ on $three$ openness values and varying k values.

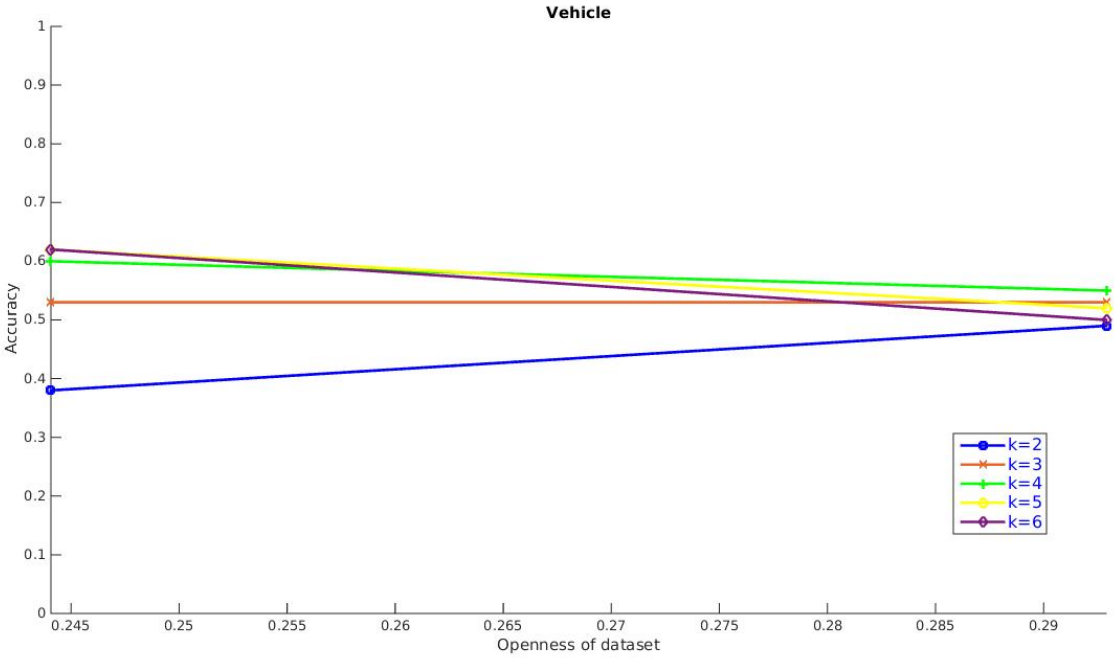


FIGURE 38: $Accuracy$ results on $Vehicle$ on two openness values and varying k values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

IX. CONCLUSION

In this paper, we have presented a novel reverse k-nearest neighbor based classifier. The elegance of this classifier lies with it's innate ability to address open set classification. RkNN based neighborhood identification does the task of unknown class detection besides the regular known class classification naturally. Choice of k or neighborhood size is dataset dependent and it is determined through cross-validation on the training set. Apart from that, no thresholding or parameters are involved to distinguish the known and unknown subspaces. A unique attribute of the proposed scheme is that it estimates and explores the sampling window implicitly. The RkNN process itself adaptively adjusts the class boundaries, depending on the local sparseness of the training data and this contributes to the simplicity and efficiency of the scheme. The proposed classifier also operates on an intrinsic multi-class framework. A comprehensive empirical study affirms the capability of the proposed scheme deliver competent to superior performance on open set backdrop the competing learners.

X. ACKNOWLEDGEMENTS

Besides the author of this paper, Late Prof. C. A. Murthy and Prof. Nikhil R. Pal of Indian Statistical Institute, Kolkata, India have made contribution to this work. The work was started by Payel Sadhukhan and Prof. Murthy and at its budding stage (on March 2018), Prof. Murthy died in a fatal road accident. Prof. Pal has participated in the writing and polishing of this work in mid 2018. I am immensely indebted to Prof. Murthy and Prof. Pal for helping me in completion of this work.

REFERENCES

[1] W. Scheirer, A. Rocha, A. Sapkota, and T. Boult, "Toward open set recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 7, pp. 1757–1772, Jul. 2013.

[2] B. Gorte and N. Gorte-Kroupnova, "Non-parametric classification algorithm with an unknown class," in Proceedings of International Symposium on Computer Vision - ISCV, Nov 1995, pp. 443–448.

[3] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability models for open set recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 11, pp. 2317–2324, Nov 2014.

[4] L. P. Jain, W. J. Scheirer, and T. E. Boult, "Multi-class open set recognition using probability of inclusion," in ECCV, 2014.

[5] E. M. Rudd, L. P. Jain, W. J. Scheirer, and T. E. Boult, "The extreme value machine," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 3, pp. 762–768, March 2018.

[6] D. Miller, L. Nicholson, F. Dayoub, and N. SÃijnderhauf, "Dropout sampling for robust object detection in open-set conditions," in 2018 IEEE International Conference on Robotics and Automation (ICRA), May 2018, pp. 3243–3249.

[7] M. Mehdipour-Ghazi, B. A. Yanikoglu, and E. Aptoula, "Open-set plant identification using an ensemble of deep convolutional neural networks," in CLEF, 2016.

[8] D. O. Cardoso, J. a. Gama, and F. M. França, "Weightless neural networks for open set recognition," Mach. Learn., vol. 106, no. 9-10, pp. 1547–1567, Oct. 2017.

[9] D. O. Cardoso, F. FranÃğa, and J. Gama, "A bounded neural network for open set recognition," in 2015 International Joint Conference on Neural Networks (IJCNN), July 2015, pp. 1–7.

[10] B. Karmakar and N. R. Pal, "How to make a neural network say âĂIJdonâĂŹt knowâĂİ," Information Sciences, vol. 430-431, pp. 444 – 466, 2018.

[11] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, "Towards open-set identity preserving face synthesis," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[12] P. R. Mendes Júnior, R. M. de Souza, R. d. O. Werneck, B. V. Stein, D. V. Pazinato, W. R. de Almeida, O. A. B. Penatti, R. d. S. Torres, and A. Rocha, "Nearest neighbors distance ratio open-set classifier," Machine Learning, vol. 106, no. 3, pp. 359–386, Mar 2017.

[13] M. A. CÃşrdova Neira, P. Ribeiro Mendes JÃžnior, A. Rocha, and R. Da Silva Torres, "Data-fusion techniques for open-set recognition problems," IEEE Access, vol. 6, pp. 21 242–21 265, 2018.

[14] R. Yoshihashi, W. Shao, R. Kawakami, S. You, M. Iida, and T. Naemura, "Classification-reconstruction learning for open-set recognition," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[15] M. RadovanoviÄĞ, A. Nanopoulos, and M. IvanoviÄĞ, "Reverse nearest neighbors in unsupervised distance-based outlier detection," IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 5, pp. 1369–1382, May 2015.

[16] D. Pokrajac, A. Lazarevic, and L. J. Latecki, "Incremental local outlier detection for data streams," in 2007 IEEE Symposium on Computational Intelligence and Data Mining, March 2007, pp. 504–515.

[17] Y. Tao, M. L. Yiu, and N. Mamoulis, "Reverse nearest neighbor search in metric spaces," IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 9, pp. 1239–1252, Sept 2006.

[18] K. Ning, H. K. Ng, S. Srihari, H. W. Leong, and A. I. Nesvizhskii, "Examination of the relationship between essential genes in ppi network and hub proteins in reverse nearest neighbor topology," BMC bioinformatics, vol. 11, p. 505, October 2010. [Online]. Available: http://europepmc.org/articles/PMC3098085

[19] C. Ji, H. Hu, Y. Xu, Y. Li, and W. Qu, "Efficient multi-dimensional spatial rknn query processing with mapreduce," in 2013 8th ChinaGrid Annual Conference, Aug 2013, pp. 63–68.

[20] J. Lu, Y. Lu, and G. Cong, "Reverse spatial and textual k nearest neighbor search," in SIGMOD Conference, 2011.

[21] A. Bryant and K. Cios, "Rnn-dbscan: A density-based clustering algorithm using reverse nearest neighbor density estimates," IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 6, pp. 1109–1121, 2018.

[22] Z. Zhang, F. Li, L. Jia, J. Qin, L. Zhang, and S. Yan, "Robust adaptive embedded label propagation with weight learning for inductive classification," IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 8, Aug 2018.

[23] Z. Zhang, F. Li, M. Zhao, L. Zhang, and S. Yan, "Joint low-rank and sparse principal feature coding for enhanced robust representation and visual classification," IEEE Transactions on Image Processing, vol. 25, no. 6, pp. 2429–2443, June 2016.

[24] Z. Zhang, W. Jiang, J. Qin, L. Zhang, F. Li, M. Zhang, and S. Yan, "Jointly learning structured analysis discriminative dictionary and analysis multiclass classifier," IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 8, pp. 3798–3814, Aug 2018.

[25] Z. Zhang, W. Jiang, Z. Zhang, S. Li, G. Liu, and J. Qin, "Scalable block-diagonal locality-constrained projective dictionary learning," in Proceedings of the 28th International Joint Conference on Artificial Intelligence, ser. IJCAI'19. AAAI Press, 2019, pp. 4376–4382. [Online]. Available: http://dl.acm.org/citation.cfm?id=3367471.3367650

[26] E. Parzen, "On estimation of a probability density function and mode," Ann. Math. Statist., vol. 33, no. 3, pp. 1065–1076, 09 1962.

[27] D. O. Loftsgaarden and C. P. Quesenberry, "A nonparametric estimate of a multivariate density function," Ann. Math. Statist., vol. 36, no. 3, 06 1965.

[28] J. AlcalÃą-fdez, A. FernÃąndez, J. Luengo, J. Derrac, S. GarcÃ■a, L. SÃąnchez, and F. Herrera, "a member of the old city publishing group. keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," 2011.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963471, IEEE Access

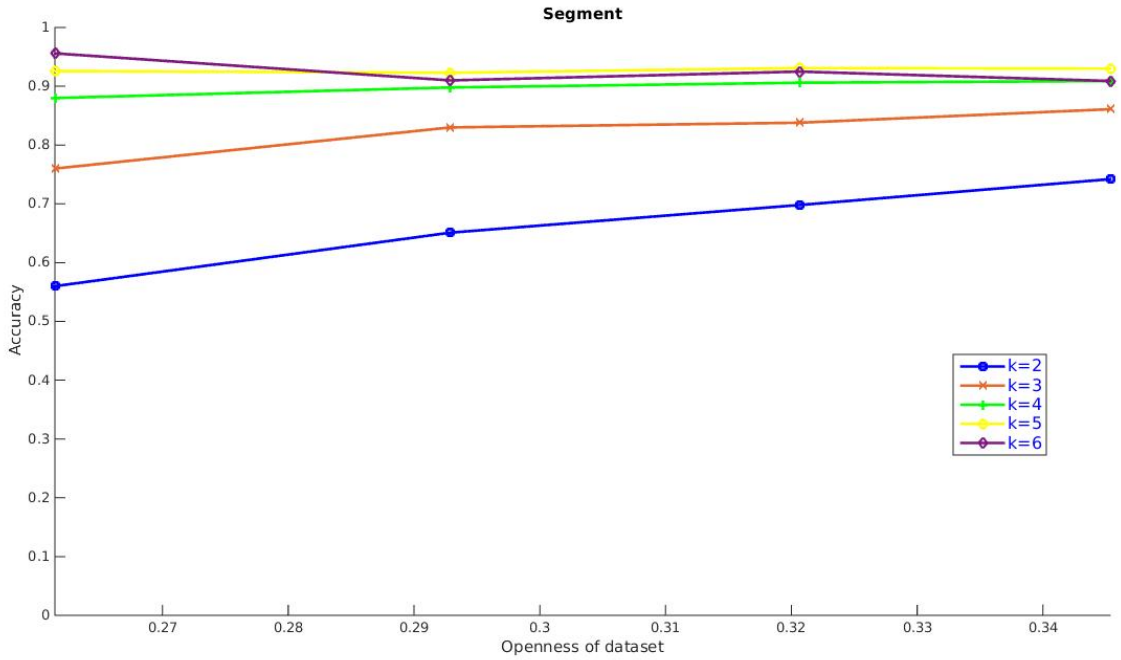


FIGURE 39: *Accuracy* results on *Segment* on *four* openness values and varying k values.

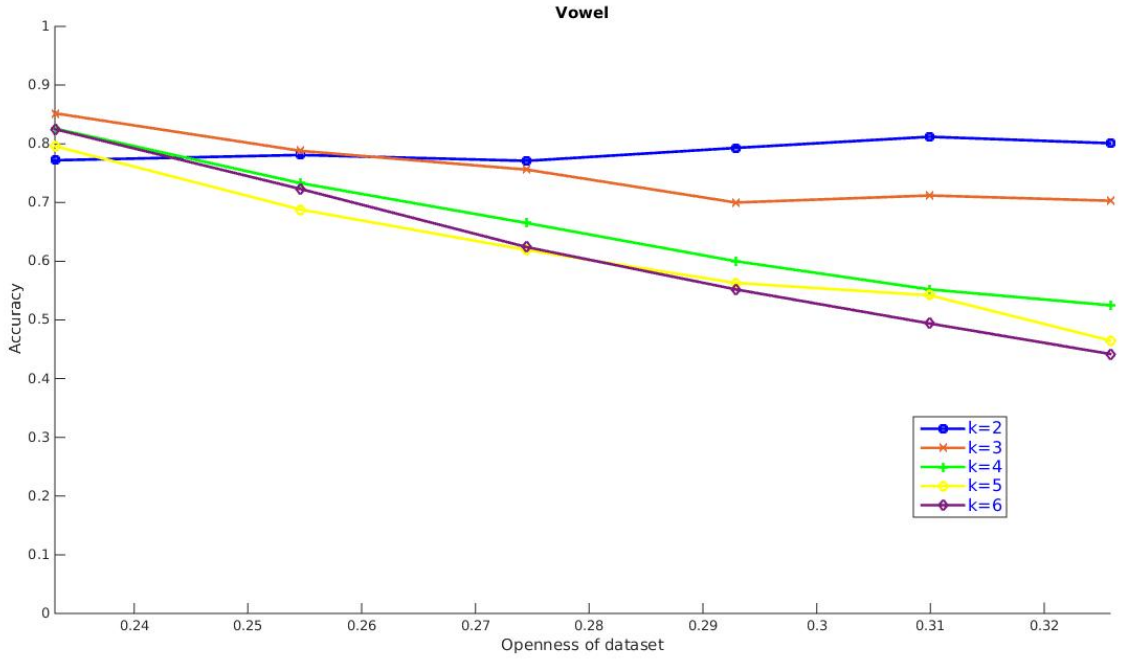


FIGURE 40: *Accuracy* results on *Vowel* on *six* openness values and varying k values.