A Scalable Classifier Ensemble Suitable for Multiclass Classification by Use of Pairwise Classifier Ensembles

Hamid Parvin[1], Alizadeh Hosein[2], Sajad Parvin[1]
[1]Nourabad Mamasani Branch
Islamic Azad University
Nourabad Mamasani, Iran
[2]Mahdi Shahr Branch
Islamic Azad University
Mahdi Shahr, Iran
hamidparvin@mamasaniiau.ac.ir, {halizadeh, s.parvin}@iust.ac.ir

ABSTRACT: *To reach the best classification, there is a way to use many inaccurate or weak classifiers; each of them is specialized for a sub-space in the problem space and using their consensus vote as the final classifier. Many methods have been proposed for classifier ensemble in pattern recognition such as Random Forest which uses a host of decision trees as base classifiers.*

The paper proposes a heuristic classifier ensemble to improve the performance of learning in the classification. It specially deals with multiclass problems which their aim is to learn the boundaries of each class among many classes. Based on the concept of multiclass problems, the classifiers are divided into two different categories: pairwise classifiers and multiclass classifiers. The aim of a pairwise classifier is to separate one class from another one. Because the pairwise classifiers are just trained to discriminate between two classes, the decision boundaries learned by them are simpler and more effective than those learned by the multiclass classifiers.

The two proposed methods are similar to Random Forest method in employing many decision trees and neural networks as base classifiers. For evaluating the proposed weighting methods, both cases of decision tree and neural network classifiers are applied in experimental results. The two proposed ensemble methods are tested on a huge Persian dataset of handwritten digits and it has been shown that the proposed ensemble methods outperform some other state-of-art ensemble methods.

Keywords: Genetic Algorithm, Optical Character Recognition, Pairwise Classifier, Multiclass Classification, Artificial Neural Networks, Decision tree

Received: 11 September 2011, Revised 28 November 2011, Accepted 4 December 2012

1. Introduction

Usage of recognition systems has found many applications in almost all fields. However, most of the classification algorithms have obtained good performance for specific problems; but they have not enough robustness for other problems. Combination of multiple classifiers can be considered as a general solution method for any pattern recognition problems. It has been shown

that combination of classifiers can usually operate better than single classifier provided that its components are independent or they have diverse outputs. It has shown that the necessary diversity of an ensemble can be achieved by manipulation of data set features. Parvin et al. have proposed some methods of creating this diversity [12]-[13].

In practice, there may be problems that one single classifier can't deliver a satisfactory performance [7]-[9]. In such situations, employing an ensemble of classifying models instead of a single classifier can reach the model to a better learning [6]. Although obtaining the more accurate classifier is often targeted, there is an alternative way to reach for it. Indeed one can use many inaccurate or weak classifiers, each of which is specialized for a few data items in the problem space and then he can employ their consensus vote as the classification. This can lead to better performance due to reinforcement of the classifier in error-prone problem spaces.

Based on the concept of multiclass problem, classifiers are divided into two different categories: pairwise classifiers and multiclass classifiers. While the aim of multiclass problems is to learn the boundaries of each class from many other classes, the aim of a pairwise classifier is to separate one class from another one. Because pairwise classifiers are just trained to learn the boundary between two classes, decision boundaries produced by them are simpler and more effective than those produced by multiclass classifiers.

Pairwise discrimination between classes has been suggested in [16]-[18]. In this model there are c*(c-1)/2 possible pairwise classifications, one for each pair of classes. The class label for an input *x* is inferred from the similarity between the code words and the outputs of the classifiers. The code word for class *q* will contain "*don't care* " symbols to denote the classifiers that are not concerned with this class label. This method is impractical for a large *c* as the number of classifiers becomes prohibitive.

In General, it is ever-true sentence that "*combining the diverse classifiers any of which performs better than a random results in a better classification performance*" [2], [6] and [10]. Diversity is always considered as a very important concept in classifier ensemble methodology. It is considered as the most effective factor in succeeding an ensemble. The diversity in an ensemble refers to the amount of differences in the outputs of its components (classifiers) in deciding for a given sample. Assume an example dataset with two classes. Indeed the diversity concept for an ensemble of two classifiers refers to the probability that they may produce two dissimilar results for an arbitrary input sample. The diversity concept for an ensemble of three classifiers refers to the probability that one of them produces dissimilar result from the two others for an arbitrary input sample. It is worthy to mention that the diversity can converge to 0.5 and 0.66 in the ensembles of two and three classifiers respectively. Although reaching the more diverse ensemble of classifiers is generally handful, it is harmful in boundary limit. It is very important dilemma in classifier ensemble field: the ensemble of accurate/diverse classifiers can be the best. It means that although the more diverse classifiers, the better ensemble, it is provided that the classifiers are better than random.

An Artificial Neural Network (ANN) is a model which is to be configured to be able to produce the desired set of outputs, given an arbitrary set of inputs. An ANN generally composed of two basic elements: (a) neurons and (b) connections. Indeed each ANN is a set of neurons with some connections between them. From another perspective an ANN contains two distinct views: (a) topology and (b) learning. The topology of an ANN is about the existence or nonexistence of a connection. The learning in an ANN is to determine the strengths of the topology connections. One of the most representatives of ANNs is MultiLayer Perceptron. Various methods of setting the strength of connections in an MLP exist. One way is to set the weights explicitly, using a prior knowledge. Another way is to 'train' the MLP, feeding it by teaching patterns and then letting it change its weights according to some learning rule. In this paper the MLP is used as one of the base classifiers.

Decision Tree (DT) is considered as one of the most versatile classifiers in the machine learning field. DT is considered as one of the unstable classifiers. It means that it can converge to different solutions in successive trainings on same dataset with same initializations. It uses a tree-like graph or model of decisions. The kind of its knowledge representation is appropriate for experts to understand what it does [11].

Its intrinsic instability can be employed as a source of the diversity which is needed in classifier ensemble. The ensemble of a number of DTs is a well-known algorithm called Random Forest (RF) which is considered as one of the most powerful ensemble algorithms. The algorithm of RF was first developed by Breiman [1].

In a previous work, Parvin et al. have only dealt with the reducing the size of classifier ensemble [9]. They have shown that one can reduce the size of an ensemble of pairwise classifiers. Indeed they propose a method for reducing the ensemble size in the

best meaningful manner. Here we inspire from their method, we propose a framework based on that a set of classifier ensembles are produced that its size order is not important. Indeed we propose an ensemble of binary classifier ensembles that has the order of c, where c is number of classes.

This paper proposes a framework to develop combinational classifiers. In this new paradigm, a multiclass classifier in addition to a few ensembles of pairwise classifiers creates a classifier ensemble. At last, to produce final consensus vote, different votes (or outputs) are gathered, after that a heuristic classifier ensemble algorithm is employed to aggregate them. The main idea behind the proposed methods is to focus classifier in the erroneous spaces of problem and use of pairwise classification concept instead of multiclass classification concept. Even the usage of pairwise classification concept instead of multiclass classification concept is not new, we propose a new pairwise classifier ensemble with a very lower order. In this paper, first the most confused classes are determined and then some ensembles of classifiers are created. The classifiers of each of these ensembles jointly work using majority weighting votes. The results of these ensembles are combined to decide the final vote in a weighted manner. Finally the outputs of these ensembles are heuristically aggregated. The proposed frameworks are evaluated on a very large scale Persian digit handwritten dataset and the experimental results show the effectiveness of the algorithm.

This paper focuses on Persian handwritten digit recognition (PHDR), especially on Hoda dataset [4]. Although there are well works on PHDR, it is not rational to compare them with each other, because there was no standard dataset in the PHDR field until 2006 [4]. The contribution is only compared with those used the same dataset used in this paper, i.e. Hoda dataset.

Rest of the paper is as following. Section 2 is about artificial neural networks. In the section 3, decision tree is explained. Section 4 deals with k-nearest neighbor classifier. The proposed methods are explained in section 5 and 7. Theoretical background is explored in section 6. Section 8 presents the experimental study. Finally, section 9 concludes the paper.

2. Artificial Neural Network

A first wave of interest in ANN (also known as 'connectionist models' or 'parallel distributed processing') emerged after the introduction of simplified neurons by McCulloch and Pitts in 1943. These neurons were presented as models of biological neurons and as conceptual components for circuits that could perform computational tasks. Each unit of an ANN performs a relatively simple job: receive input from neighbors or external sources and use this to compute an output signal which is propagated to other units. Apart from this processing, a second task is the adjustment of the weights. The system is inherently parallel in the sense that many units can carry out their computations at the same time. Within neural systems it is useful to distinguish three types of units: input units (indicated by an index i) which receive data from outside the ANN, output units (indicated by an index o) which send data out of the ANN, and hidden units (indicated by an index h) whose input and output signals remain within the ANN. During operation, units can be updated either synchronously or asynchronously. With synchronous updating, all units update their activation simultaneously; with asynchronous updating, each unit has a (usually fixed) probability of updating its activation at a time t, and usually only one unit will be able to do this at a time. In some cases the latter model has some advantages.

An ANN has to be configured such that the application of a set of inputs produces the desired set of outputs. Various methods to set the strengths of the connections exist. One way is to set the weights explicitly, using a priori knowledge. Another way is to 'train' the ANN by feeding it teaching patterns and letting it change its weights according to some learning rule. For example, the weights are updated according to the gradient of the error function. For further study the reader must refer to an ANN book such as Haykin's book on theory of ANN [3].

3. Decision Tree Learning

DT as a machine learning tool uses a tree-like graph or model to operate deciding on a specific goal. DT learning is a data mining technique which creates a model to predict the value of the *goal* or *class* based on input variables. Interior nodes are the representative of the input variables and the leaves are the representative of the target value. By splitting the source set into subsets based on their values, DT can be learned. Learning process is done for each subset by recursive partitioning. This process continues until all remain features in subset has the same value for our goal or until there is no improvement in *Entropy*. Entropy is a measure of the uncertainty associated with a random variable.

Data comes in records of the form: $(x,Y) = (x_1, x_2, x_3, \ldots, x_n, Y)$. The dependent variable, Y, is the target variable that we are trying

to understand, classify or generalize. The vector **x** is composed of the input variables, x_1, x_2, x_3 etc., that are used for that task. To clarify that what the DT learning is, consider Figure 1. Figure 1 has 3 *attributes Refund, Marital Status* and *Taxable Income* and our goal is cheat status. We should recognize if someone cheats by the help of our 3 attributes. To do learn process, attributes split into subsets. Figure 2 shows the process tendency. First, we split our source by the *Refund* and then *MarSt* and *TaxInc*.

For making rules from a decision tree, we must go upward from leaves as our antecedent to root as our consequent. For example consider Figure 2. Rules such as following are apprehensible. We can use these rules such as what we have in Association Rule Mining.

T id	Refund	Material Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Figure 1. An exemplary raw data

- Refund = Yes ⇒ cheat = No
- TaxInc < 80, MarSt = (Single or Divorce), Refund = No ⇒ cheat = No
- TaxInc > 80, MarSt = (Single or Divorce), Refund = No ⇒ cheat = Yes
- Refund = No, MarSt = Married ⇒ cheat = No

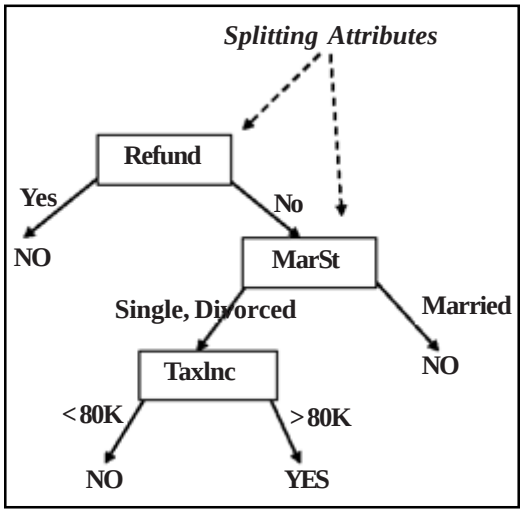

Figure 2. The process tendency for Figure 1

4. K-Nearest Neighbor Algorithm

K-nearest neighbor algorithm (k-NN) is a method for classifying objects based on closest training examples in the feature space.

k-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. The k-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its *k* nearest neighbors (*k* is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of its nearest neighbor.

As it is obvious, the k-NN classifier is a stable classifier. A stable classifier is the one converge to an identical classifier apart from its training initialization. It means the 2 consecutive trainings of the k-NN algorithm with identical *k* value, results in two classifiers with the same performance. This is not valid for the MLP and DT classifiers. We use 3-NN as a base classifier in the paper. It is then inferred that using a k-NN classifier in an ensemble is not a good option.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	969	0	0	4	1	14	2	0	0	1
1	4	992	1	0	2	4	1	1	1	15
2	1	1	974	18	9	1	4	4	0	1
3	0	0	13	957	12	0	3	2	0	1
4	5	0	3	17	973	3	2	2	0	3
5	15	0	0	0	0	977	1	0	0	0
6	2	6	2	1	3	0	974	5	1	3
7	3	0	3	1	0	1	1	986	0	0
8	0	1	0	1	0	0	2	0	995	0
9	1	0	4	1	0	0	10	0	3	976

Table 1. Unsoft confusion matrix pertaining to the Persian handwritten OCR

5. First Proposed Algorithm

The main idea behind the first proposed method is to use a number of pairwise classifiers to reinforce the main classifier in the error-prone regions of the problem space. Figure 3 depicts the training phase of the first proposed method schematically.

In the first proposed algorithm, a multiclass classifier is first trained. Its duty is to produce a confusion matrix over the validation set. Note that this classifier is trained over the total train set. At next step, the pair-classes which are mostly confused with each other and are also mostly error-prone are detected. After that, a number of pairwise classifiers are employed to reinforce the drawbacks of the main classifier in those error-prone regions. A simple heuristic is used to aggregate their outputs.

At the first step, a multiclass classifier is trained on all train data. Then, using the results of this classifier on the validation data, confusion matrix is obtained. This matrix contains important information about the functionalities of classifiers in the dataset localities. The close and Error-Prone Pair-Classes (EPPC) can be detected using this matrix. Indeed, confusion matrix determines the between-class error distributions. Assume that this matrix is denoted by *a*. Item a_{ij} of this matrix determines how many instances of class c_j have been misclassified as class c_i.

Table 1 shows the confusion matrix obtained from the base multiclass classifier. As you can see, digit 5 (or equivalently class 6) is incorrectly recognized as digit 0 fifteen times (or equivalently class 1), and also digit 0 is incorrectly recognized as digit 5 fourteen times. It means 29 misclassifications have totally occurred in recognition of these two digits (classes). The mostly erroneous pair-classes are respectively (2, 3), (0, 5), (3, 4), (1, 4), (6, 9) and so on according to this matrix. Assume that the *i*-th mostly EPPC is denoted by $EPPC_i$. So $EPPC_1$ will be (2, 3). Also assume that the number of selected EPPC is denoted by *k*.

After determining the mostly erroneous pair-classes, or EPPCs, a set of *m* ensembles of binary classifiers is to be trained to jointly, as an ensemble of binary classifiers, reinforce the main multiclass classifier in the region of each EPPC. So as it can be

inferred, it is necessary to train *k* ensembles of *m* binary classifiers. Assume that the ensemble which is to reinforce the main multiclass classifier in the region of $EPPC_i$ is denoted by PWC_i. Each binary classifier contained in PWC_i, is trained over a bag of train data like RF. The bags of train data contain only *b* percept of the randomly selected of train data. It is worthy to be mentioned that pairwise classifiers which are to participate in PWC_i are trained only on those instances which belongs to $EPPC_i$. Assume that the *j*-th classifier binary classifier of PWC_i is denoted by $PWC_{i,j}$. Because there exists *m* classifiers in each of PWC_i and also there exists *k* EPPC, so there will be *k* * *m* binary classifiers totally. For example in the Table 1 the EPPC (2, 3) can be considered as an erroneous pair-class. So a classifier is necessary to be trained for that EPPC using those data items of train data that belongs to class 2 or class 3. As mentioned before, this method is flexible, so we can add arbitrary number of PWC_i to the base primary classifiers. It is expected that the performance of the first proposed framework outperforms the primary base classifier. It is worthy to note that the accuracies of $PWC_{i,j}$ can easily be approximated using the train set. Because $PWC_{i,j}$ is trained only on *b* percept of the train set with labels belong to $EPPC_i$, provided that *b* is very small rate, then the accuracy of $PWC_{i,j}$ on the train set with labels belong to $EPPC_i$ can be considered as its approximated accuracy. Assume that the mentioned approximated accuracy of $PWC_{i,j}$ is denoted by $P_{i,j}$.

It is important to note that each of PWC_i acts as a binary classifier. As it mentioned each PWC_i contains *m* binary classifiers with an accuracy vector, P_i. It means of these binary ensemble can take a decision with weighed sum algorithm illustrated in [5]. So we can combine their results according to weighs computed by the equation 1.

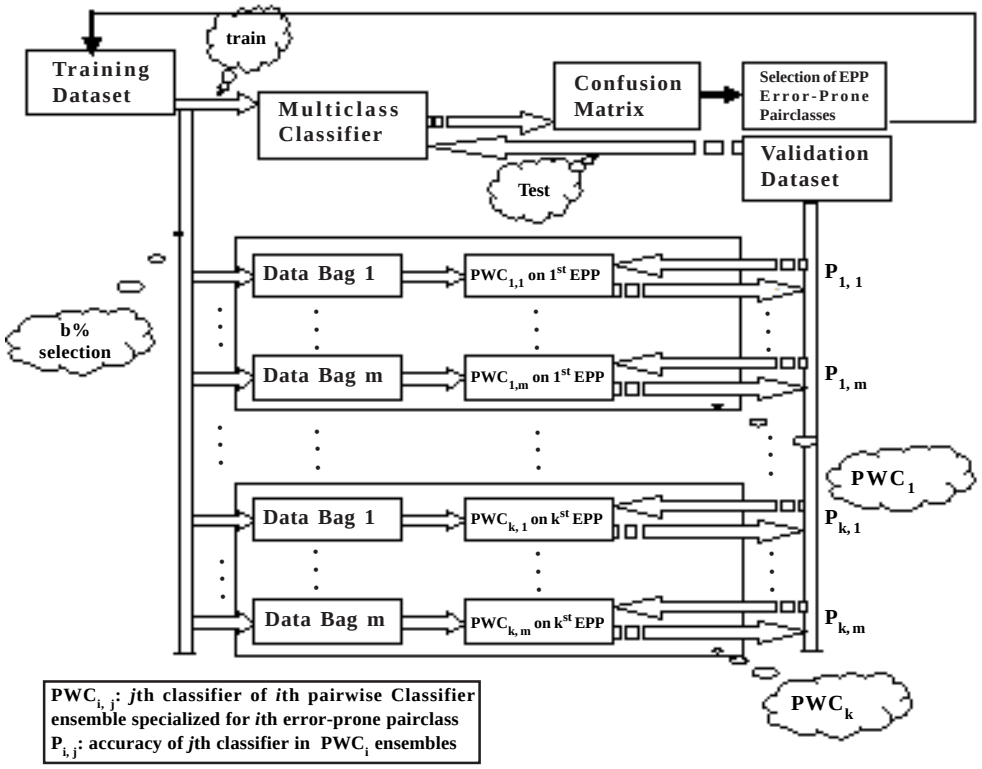

Figure 3. The first training phase of the first and second proposed method

$$w_{i,j} = \log\left(\frac{p_{i,j}}{1-p_{i,j}}\right) \quad (1)$$

where $w_{i,j}$ is the accuracy of *j*-th classifier in the *i*-th binary ensemble. It is proved that the weights obtained according to the equation 1 are optimal weights in theory. Now the two outputs of each PWC_i are computed as equation 2.

$$PWC_i(x \mid h) = \sum_{j=1}^{m} w_{i,j} * PWC_i(x \mid h), \qquad h \in EPPC_i \quad (2)$$

where x is a test data.

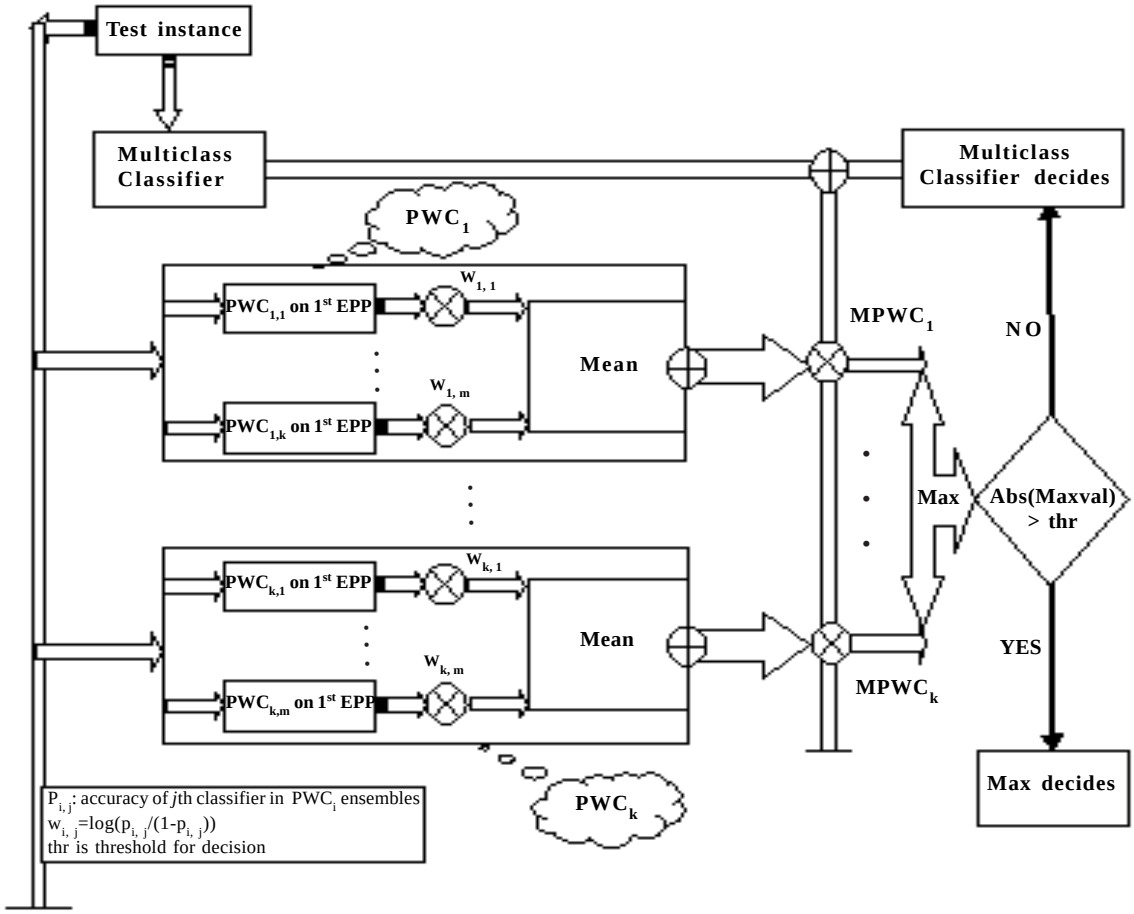

Figure 4. Heuristic test phase of the first proposed method test

The last step of the first proposed framework is to combine the results of the main multiclass classifier and those of PWC_i. It is worthy to note that there are $2 * k$ outputs from the binary ensembles plus c outputs of the main multiclass classifier. So the problem is to map a $2 * k + c$ intermediate space to a c space each of which corresponds to a class. The results of all these classifiers are fed as inputs in the aggregators. The Output i of aggregator is the final joint output for class i. Here, the aggregation is done using a special heuristic method. This process is done using a heuristic based ensemble which is illustrated in the Figure 4. As Figure 4 shows, after producing the intermediate space, the outputs of i-th ensemble of binary classifier are multiplied in a q_i number. This q_i number is equal to the sum of the main multiclass classifier's confidences for the classes belong to $EPPC_i$. Assume that the results of the multiplication of q_i by the outputs of PWC_i are denoted by $MPWC_i$. It is important to note that $MPWC_i$ is a vector of two confidences; the confidences of the classifier framework to the classes belonging to PWC_i.

After calculating the $MPWC_i$, the max value is selected between all of them. If the framework's confidence for the most confident class is satisfactory for a test data, then it is selected for final decision of framework, else the main multiclass classifier decides for the data. It means that the final decision is taken by equation 3.

$$Decision\ (x) = \begin{cases} MaxDecision\ (x) & \max\limits_{h \in EPPC_{Sc}} (MPWC_{sc}(h \mid x)) > thr \\ \max\limits_{h \in \{1,\ldots c\}} (MCC(h \mid x)) & otherwise \end{cases} \quad (3)$$

where $MCC\ (h \mid x)$ is the confidence of the main multiclass classifier for the class h given a test data x. $MPWC_{sc}\ (h \mid x)$ is the confidence of the sc-th ensemble of binary classifiers for the class h given a test data x. *MaxDecision* is calculated according to equation 4.

$$MaxDecision\,(x) = arg \max_{h \in EPPC_{Sc}} (MPWC_{sc}(h \mid x)) \tag{4}$$

where *sc* is:

$$sc\,(x) = arg \max_{i} (\max_{h \in EPPC_i} (MPWC_i(h \mid x))) \tag{5}$$

Because of the reinforcement of the main classifier by some ensembles in erroneous regions, it is expected that the accuracy of this method outperforms a simple MLP or unweighted ensemble. Figure 3 along with Figure 4 stands as the structure of the ensemble framework.

6. Theoretical Background

As we presume in the paper, it is aimed to add as many as pairwise classifiers to compensate a predefined error rate, *PDER* * *EF* (*MCL*, *DValidation*), where *PDER* is a predefined error rate and *EF* (*MCL*, *DValidation*) is error frequency of multiclass classifier, MCL, over the validation data, *DValidation*. Assume we add |*EPS*| pairwise classifiers to the main *MLC*. It is as in the equation below.

$$\begin{aligned} &\sum_{i=1}^{|eps|} (p(\hat{w} = EPPC_i.x \mid w = EPPC_i.y, x) + p(\hat{w} = EPPC_i.y \mid w = EPPC_i.x, x)) \\ &= PDER * EF(MCL, DValidation, DTrain) \end{aligned} \tag{6}$$

Now assume that a data instance ***x*** which belongs really to class *q* is to be classified by the first proposed algorithm; it has the error rate which can be obtain by equation 12. First assume p^{p}_{max} is probability for the first proposed classifier ensemble to take decision by one of its binary classifiers that is able to distinguish two classes: *q* and *p*. Also assume p^{pr}_{max} is probability for the first proposed classifier ensemble to take decision by one of its binary classifiers that is able to distinguish two classes: *r* and *p*. They can be is obtained by equation 7 and 8 respectively.

$$p^{pr}_{\max}(EPPC = (p,r) \mid x \in q) = (MCC(p \mid x) + MCC(r \mid x)) * \max(PWC(p \mid x), PWC(r \mid x)) \tag{7}$$

$$p^{p}_{\max}(EPPC = (p,q) \mid x \in q) = (MCC(p \mid x) + MCC(q \mid x)) * \max(PWC(p \mid x), PWC(q \mid x)) \tag{8}$$

We can assume equation 9 without losing generality.

$$\forall r \neq q \mid \max(PWC(p \mid x \in q), PWC(r \mid x \in q)) \cong \mu << \max(PWC(p \mid x \in q), PWC(q \mid x \in q)) = \lambda \tag{9}$$

where μ is a fixed value and then we have:

$$p^{pr}_{max}(EPPC = (p,r) \mid x \in q) \cong (MCC(p \mid x) + (MCC(r \mid x)) \times \mu \propto (b_{p,q} + b_{r,q}) \times \mu \tag{10}$$

$$p^{p}_{max}(EPPC = (p,q) \mid x \in q) = (MCC(p \mid x) + (MCC(q \mid x)) \times \lambda = (b_{p,q} + b_{q,q}) \times \lambda \tag{11}$$

As it is inferred from the algorithm in the same condition, its error can be formulated as follow.

$$error(x \mid w = q) = \sum_{EPPC=(p,q)} p^{p}_{max}(EPPC \mid x) * p_{pair}(p \mid x) + \sum_{EPPC=(p,r)} p^{pr}_{max}(EPPC \mid x) + (1 - p^{p}_{max} - p^{pr}_{max})(1 - b_{q,q}) \tag{12}$$

where p_{pair} is probability of taking correct decision by binary classifier and $b_{j,q}$ is defined as follow.

$$b_{j,q} = \frac{confusion_{j,p}}{\sum_{i=1}^{c} confusion_{i,p}} \tag{13}$$

So we can reformulate equation 12 as follow:

$$error(x \mid w=q) = \sum_{EPPC=(p,q)} p^{p}_{\max}(EPPC \mid x) * p_{pair}(p \mid x) + \sum_{EPPC=(p,r)} p^{pr}_{\max}(EPPC \mid x) + (1 - p^{p}_{\max} - p^{pr}_{\max})(1 - b_{q,q}) \cong \sum_{EPPC=(p,q)} p^{p}_{\max}(EPPC \mid x) * p_{pair}(p \mid x) + (1 - p^{p}_{\max} - p^{pr}_{\max})(1 - b_{q,q}) \quad (14)$$

Note that in equation 14 if p^{pr}_{max} and p^{r}_{max} are zero for an exemplary input the error of classification will be still equal to the main multiclass classifier. If they are not zero for an exemplary input the misclassification rate will still be reduced because of reduction in second part of equation 14. Although the first part increases the error in equation 14, but if we assume that the binary classifiers are more accurate than the multiclass classifier, then the increase is nullified by the decrease part.

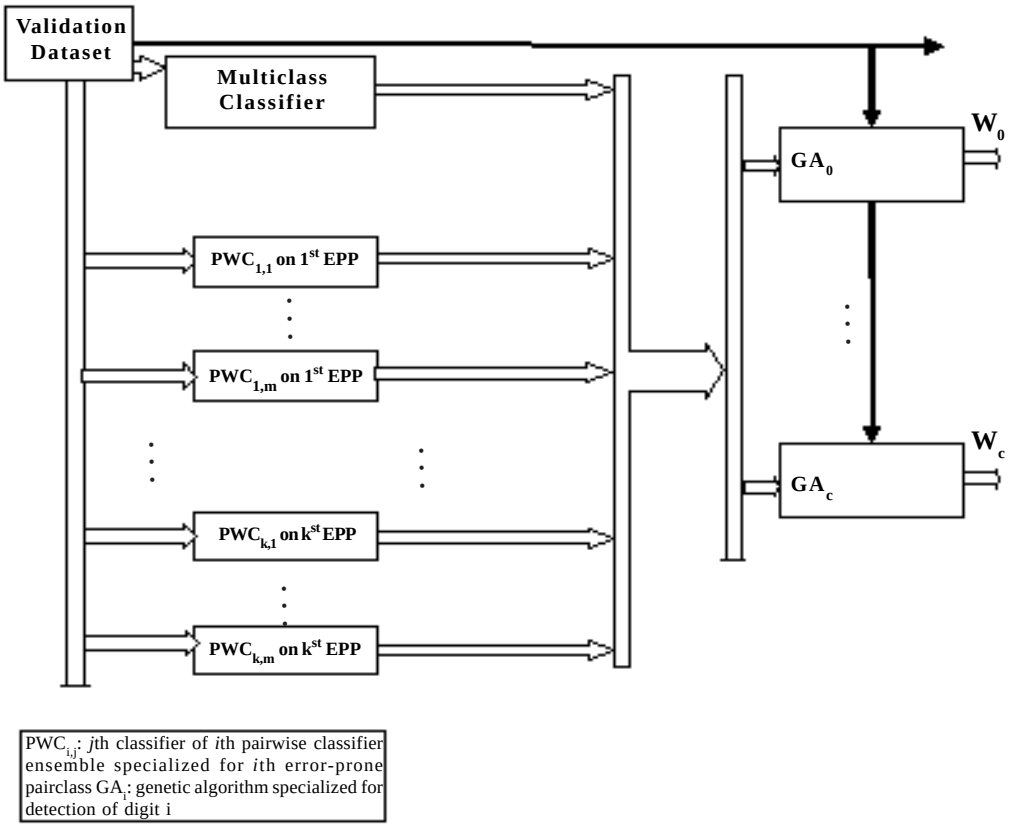

Figure 5. The second training phase of the second proposed method based on GA

7. Second Proposed Algorithm

After Figure 3, the last step of the second proposed framework is to combine the results of the main multiclass classifier and those of PWC_i. It is worthy to note that there are $2 * k$ outputs from the binary ensembles plus c outputs of the main multiclass classifier. So the problem is to map a $2 * k + c$ intermediate space to a c space each of which corresponds to a class. The results of all these classifiers are fed as inputs for the aggregators. Note that there are c aggregators, one per each class. The Output of aggregator i is the final joint output for class i. Here, the aggregation is done using a special weighting method. The problem here is how one can optimally determine these weights. In this paper, GA is employed to find these weights.

Because of the capability of the GA in passing local optimums, it is expected that the accuracy of this method outperforms a simple MLP or unweighted ensemble. Figure 3 along with Figure 5 and Figure 6 depicts the structure of the second ensemble framework.

As it is shown in Figure 5, in the second proposed framework, the number of times that GA is invoked is equal to c, which is the

number of digits (classes). This GA-based algorithm is overall illustrated by Figure 5.

In fact, each GA creates an ensemble to detect one digit (class), by considering the $2 * k + c$ intermediate space obtained by the multiclass classifier plus the binary classifier ensembles as new feature space. Each GA uses one hyper-line in this new intermediate feature space, by assigning a weight to each dimension. The chromosome representation of GA_i is a vector of real numbers. The function of GA_i is calculated as equation 15.

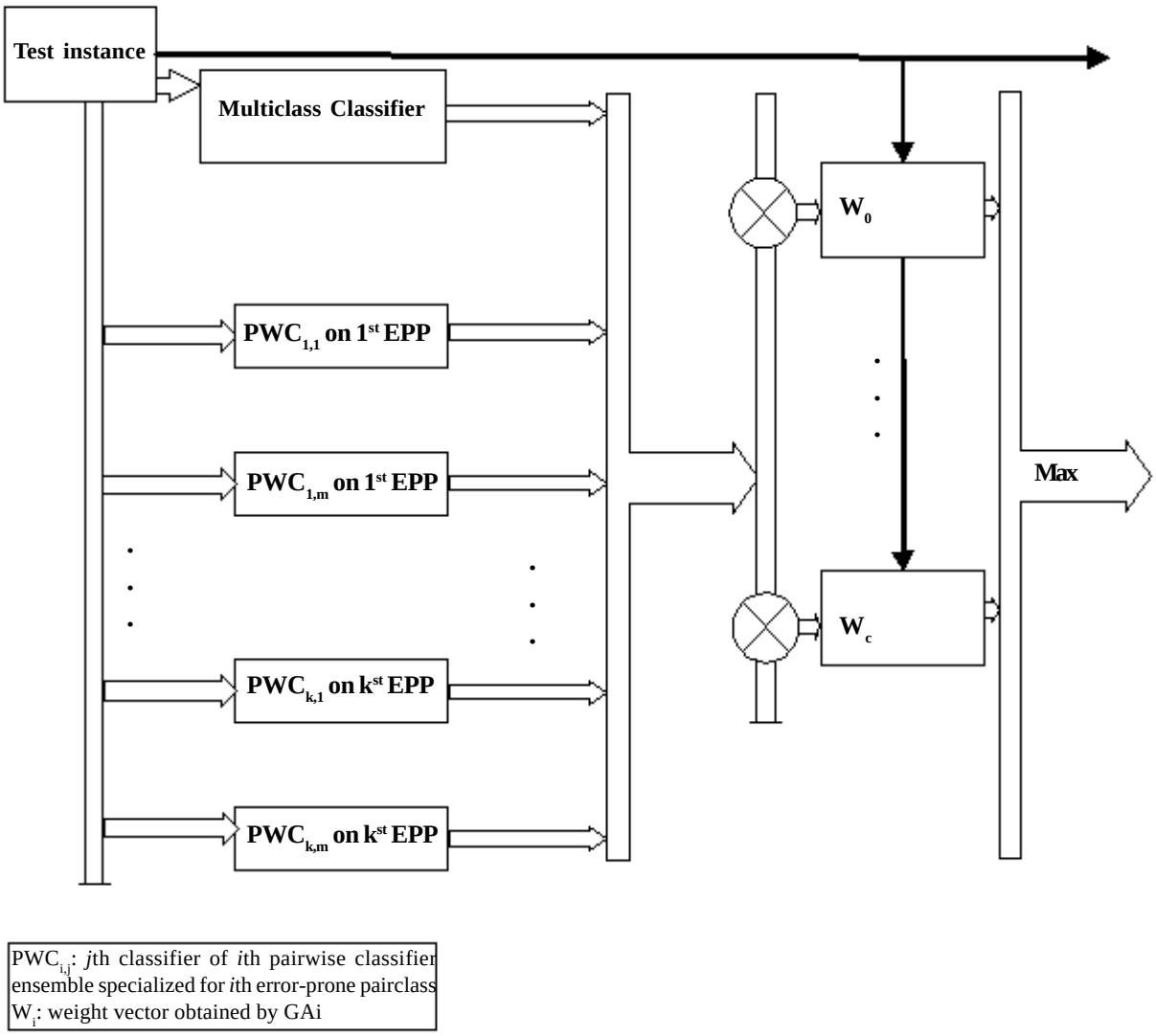

Figure 6. Test phase of the second proposed method based on GA

$$Fitness\,(W_i, Valset) = \sum_{x \in Valset} f(x, W_i) \tag{15}$$

where the function $f(x, W_i)$ is computed as equation 16.

$$f(x, W_i) = sign\,(x, i) * (BinOuts\,(x, W_i) + MultiOuts\,(x, W_i)) \tag{16}$$

where the function $sing\,(x, i)$ is computed as equation 17.

$$singn\,(x, i) = \begin{cases} 1 & lable\,(x) = i \\ -1 & lable\,(x) \neq i \end{cases} \tag{17}$$

and *ValSet* in equation 16 is validation set. In the equation 16, *BinOuts* is the weighted sum of the outputs of the binary ensembles, given an input sample *x*, which is computed as equation 18, and *MultiOuts* is the weighted sum of the outputs of the main multiclass classifier, given an input sample *x*.

$$BinOuts(x, W_i) = \sum_{j=1}^{k} \sum_{h=1}^{2} W_i(s) * PWC_{i,j}(x \mid EPPC^{h}_{\lceil j/2 \rceil}) \quad (18)$$

where *s* is computed as equation 19.

$$s = (j - 1) * 2 \quad (19)$$

and *MultiOuts* is also computed as equation 20.

$$MultiOuts(x, W_i) = \sum_{j=1}^{c} W_i(2 * k + j) * MCC(x \mid j) \quad (20)$$

Indeed GA_i try to better discriminate the class *i* from other classes. Finally, the most voted class is selected as final decision of the framework as depicted in the Figure 6. This is simply done using a max function as it is obvious from the Figure 6. It means that the final decision is taken by equation 21.

$$FinalDecision(x) = arg\max_{i} f(x, W_i) \quad (21)$$

8. Experimental Study

In this section, we present experiments comparing both proposed methods with the leading ensemble methods.

8.1 Benchmark

This section evaluates the results of applying both proposed frameworks on a Persian handwritten digit dataset named Hoda [4]. This dataset contains 102,364 instances of digits 0-9. Dataset is divided into 3 parts: train, evaluation and test sets. Train set contains 60,000 instances. Evaluation and test datasets are contained 20,000 and 22,364 instances. The 106 features from each of them have been extracted which are described in [4].

8.2 Experimental Setting

In this paper, MLP, 3-NN and DT are used as base primary classifier. We use an MLPs with 2 hidden layers including respectively 10 and 5 neurons in the hidden layer 1 and 2, as the base Multiclass classifier. Confusion matrix is obtained from its output. Also DT's measure of decision is taken as Gini measure. The classifiers' parameters are kept fixed during all of their experiments. It is important to take a note that all classifiers in the algorithm are kept unchanged. It means that all classifiers are considered as MLP in the first experiments. After that the same experiments are taken by substituting all MLPs whit DTs.

The parameter *k* is set to 11. So, the number of pairwise ensembles of binary classifiers added equals to 11 in the experiments. The parameter *m* is also set to 9. So, the number of binary classifiers per each EPPC equals to 9 in the experiments. It means that 99 binary classifiers are trained for the pair-classes that have considerable error rates. Assume that the error number of each pair-class is available. For choosing the most erroneous pair-classes, it is sufficient to sort error numbers of pair-classes. Then we can select an arbitrary number of them. This arbitrary number can be determined by try and error which it is set to 11 in the experiments.

As mentioned 9 *11 = 99 pairwise classifiers are added to main multiclass classifier. As the parameter *b* is selected 20, so each of these classifiers is trained on only *b* precepts of corresponding train data. It means each of them is trained over 20 percept of the train set with the corresponding classes. The cardinality of this set is calculated by equation 22.

$$car = ||train|| * 2 * b / c = 60000 * 2 * 0.2 / 10 = 2400 \quad (22)$$

It means that each binary classifier is trained on 2400 data points with 2 class labels.

8.3 Experimental Results and Discussion

Table 2 shows the experimental results comparatively. As it is inferred the two frameworks outperform the previous works and the simple classifiers in the case of employing decision tree as the base classifier.

Methods	Base Classifier		
	DT	MLP	KNN
A simple multiclass classifier	95.57	95.7	96.66
Method Proposed in [8]	-	98.89	-
Method Proposed in [7]	-	98.27	-
Method Proposed in [15]	97.20	96.70	96.86
Unweighted Full Ensemble in [14]	98.22	98.11	-
Unweighted Static Classifier Selection in [14]	98.13	98.15	-
Weighted Static Classifier Selection in [14]	98.34	98.21	-
First Proposed Method	**99.01**	98.46	96.89
Second Proposed Method	98.99	**99.04**	**97.14**

Table 2. The accuracies of different settings of both proposed frameworks

It is inferred from Table 2 that both proposed frameworks cause a significant improvement in the classification precision specially when employing DT as base classifier. Taking a look at Table 2 shows that using DT as base classifier in ensemble almost always produces a better performing classification. It may be due to inherent instability of DT. It means that because a DT is unstable classifier, so it is better to use it as a base classifier in an ensemble. A stable classifier is the one converge to an identical classifier apart from its training initialization. It means the 2 consecutive trainings of the classifier with identical initializations, results in two classifiers with the same performance. This is not valid for the MLP and DT classifiers. Although MLP is not a stable classifier, it is more stable than DT. So it is also expected that using DT classifier as base classifier has the most impact in improving the recognition ratio.

As another point to be mentioned, reader can infer that using the framework can outperforms Unweighted Full Ensemble, Unweighted Static Classifier Selection and Unweighted Static Classifier Selection methods explained in [14]. This can be in consequence of employing binary classifiers instead of multiclass classifiers.

It is inferred from the Table 2 that both proposed frameworks affect significantly in improving the classification precision specially when employing DT and MLP as base classifier. It is also obvious that using DT classifier as base classifier has the most impact in improving the recognition ratio. It is may be due to its inherent instability.

As it is expected using a stable classifier like k-NN in an ensemble is not a good option and unstable classifiers like DT and MLP are better options.

9. Conclusion

Although the more accurate classifier leads to a better performance, there is another option to use many inaccurate classifiers while each one is specialized for a few data in the problem space and using their consensus vote as the classifier. So this paper proposes a heuristic classifier ensemble to improve the performance of learning in multiclass problems. The main idea behind both proposed method is to focus classifier in the erroneous spaces of the problem. The two new proposed methods try to improve the performance of multiclass classification system. We also propose a framework based on that a set of classifier ensembles are produced that its size order is not important. It means that we propose a new pairwise classifier ensemble with a very lower order than usage of all possible pairwise classifiers. Indeed paper proposes an ensemble of binary classifier ensembles that has the order of c, where c is number of classes. So first an arbitrary number of binary classifier ensembles are added to main classifier. Then results of all these binary classifier ensembles are given to a set of a heuristic based ensemble. The results of these binary ensembles indeed are combined to decide the final vote in a weighted manner. The two proposed frameworks are evaluated on a very large scale Persian digit handwritten dataset and the experimental results show the effectiveness of the

algorithm. Usage of confusion matrix makes two proposed methods flexible ones. The number of all possible pairwise classifiers is $c * (c - 1) / 2$ that it is $O(c \wedge 2)$. Using this method without giving up a considerable accuracy, we decrease its order to $O(1)$. This feature of our proposed methods makes them applicable for problems with a large number of classes. The experiments show the effectiveness of this method. Also we reached to very good results in Persian handwritten digit recognition which is a very large dataset.

10. Acknowledgement

This research is supported by Islamic Azad University, Nourabad Mamasani Branch, Nourabad Mamasani, Iran.

References

[1] Breiman, L. (1996). Bagging Predictors. *Journal of Machine Learning*, 24 (2) 123-140.

[2] Gunter, S., Bunke, H. (2002). Creation of classifier ensembles for handwritten word recognition using feature selection algorithms. *IWFHR.*

[3] Haykin, S. (1999). Neural Networks, a comprehensive foundation. Prentice Hall International.

[4] Khosravi, H., Kabir, E. (2007). Introducing a very large dataset of handwritten Farsi digits and a study on the variety of handwriting styles. *Pattern Recognition Letters*, 28 (10) 1133-1141.

[5] Kuncheva, L. I. (2005). Combining Pattern Classifiers, Methods and Algorithms. New York: Wiley.

[6] Minaei-Bidgoli, B., Punch, W.F (2003). Using Genetic Algorithms for Data Mining Optimization in an Educational Web-based System. *GECCO*, p. 2252-2263.

[7] Parvin H., Alizadeh H., Minaei-Bidgoli B (2008). A New Approach to Improve the Vote-Based Classifier Selection. *International Conference on Networked Computing and advanced Information Management*, p. 91-95.

[8] Parvin, H., Alizadeh, H., Fathi, M., Minaei-Bidgoli, B. (2008). Improved Face Detection Using Spatial Histogram Features. *Int. Conf. on Image Processing, Computer Vision, and Pattern Recognition*, p. 381-386.

[9] Parvin, H., Alizadeh, H., Minaei-Bidgoli, B., Analoui, M. (2008). An Scalable Method for Improving the Performance of Classifiers in Multiclass Applications by Pairwise Classifiers and GA. *International Conference on Networked Computing and advanced Information Management*, p. 137-142.

[10] Saberi, A., Vahidi, M., Minaei-Bidgoli, B. (2007). Learn to Detect Phishing Scams Using Learning and Ensemble Methods. *IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Workshops*, p. 311-314.

[11] Yang, T. (2006). *International Journal of Computational Cognition*, 4 (4) 34–46.

[12] Parvin, H., Alizadeh, H., Minaei-Bidgoli, B. (2009). A New Method for Constructing Classifier Ensembles. *JDCTA* 3 (2) 62-66.

[13] Parvin, H., Alizadeh, H., Minaei-Bidgoli, B. (2009). Using Clustering for Generating Diversity in Classifier Ensemble. *JDCTA* 3 (1) 51-57.

[14] Parvin H., Alizadeh, H. (2011). Classifier Ensemble Based Class Weighting. *American Journal of Scientific Research*, 11 (19) 84-90.

[15] Parvin, H., Alizadeh, H., Moshki, M., Minaei-Bidgoli, B., Mozayani, N. (2011). Divide & Conquer Classification and Optimization by Genetic Algorithm. *International Conference on Convergence and hybrid Information Technology*, p. 858-863.

[16] Masulli, F., Valentini, G. (2000). Comparing decomposition methods for classification. *In: Proc. International Conference on Knowledge-Based Intelligent Engineering Systems and Applied Technologies*, p. 788-792.

[17] Cutzu, F. (2003). Polychotomous classification with pairwise classifiers: A new voting principle. *In: Proc. 4th International Workshop on Multiple Classifier Systems*, p. 115-124.

[18] Jozwik A., Vernazza G. (1987). Recognition of leucocytes by a parallel k-nn classifier. *In: Proc. International Conference on Computer-Aided Medical Diagnosis*, p. 138-153.

[19] Parvin, H., Minaei-Bidgoli, B. and Alizadeh, H. (2011). A Heuristic Classifier Ensemble for Huge Data. International Conferences on. *Active Media Technology*, p. 29-38.