

Manuscript Number: CBM-D-10-00319R1

Title: CLASSIFICATION AND RETRIEVAL ON MACROINVERTABRATE IMAGE DATABASES

Article Type: Full Length Article

Keywords: Biomonitoring, classification, radial basis function networks, multilayer perceptrons, Bayesian Networks, Support Vector Machines, Benthic macroinvertebrate

Corresponding Author: Dr. serkan kiranyaz,

Corresponding Author's Institution: Tampere University of Technology

First Author: serkan kiranyaz

Order of Authors: serkan kiranyaz; Turker Ince; Jenni Pulkkinen; Moncef Gabbouj; Johanna Arje; Salme Karkkainen; Ville Tirronen ; Martti Juhola; Tuomas Turpeinen ; Kristian Meissner

Abstract: Aquatic ecosystems are continuously threatened by a growing number of human induced changes. Macroinvertebrate biomonitoring is particularly efficient in pinpointing the cause-effect structure between slow and subtle changes and their detrimental consequences in aquatic ecosystems. The greatest obstacle to implementing efficient biomonitoring is currently the cost-intensive human expert taxonomic identification of samples. While there is evidence that automated recognition techniques can match human taxa identification accuracy at greatly reduced costs, so far the development of automated identification techniques for aquatic organisms has been minimal. In this paper, we focus on advancing classification and data retrieval that are instrumental when processing large macroinvertebrate image datasets. To accomplish this for routine biomonitoring, in this paper we shall investigate the feasibility of automated river macroinvertebrate classification and retrieval with high precision. Besides the state-of-the-art classifiers such as Support Vector Machines (SVMs) and Bayesian Classifiers (BCs), the focus is particularly drawn on feed-forward artificial neural networks (ANNs), namely multilayer perceptrons (MLPs) and radial basis function networks (RBFNs). Since both ANN types have been proclaimed superior by different investigations even for the same benchmark problems, we shall first show that the main reason for this ambiguity lies in the static and rather poor comparison methodologies applied in most earlier works. Especially the most common drawback occurs due to the limited evaluation of the ANN performances over just one or few network architecture(s). Therefore, in this study, an extensive evaluation of each classifier performance over an ANN architecture space is performed. The best classifier among all, which is trained over a dataset of river macroinvertebrate specimens, is then used in the MUVIS framework for the efficient search and retrieval of particular macroinvertebrate peculiars. Classification and retrieval results present such a high accuracy that can match experts' ability for taxonomic identification.

CLASSIFICATION AND RETRIEVAL ON MACROINVERTABRATE IMAGE DATABASES

Serkan Kiranyaz, Turker Ince, Jenni Pulkkinen and Moncef Gabbouj¹,

Serkan.kiranyaz@tut.fi, turker.ince@ieu.edu.tr, jenni.pulkkinen@tut.fi, moncef.gabbouj@tut.fi,

Johanna Arje and Salme Karkkainen,

johanna.arje@jyu.fi, salme.karkkainen@jyu.fi,

Ville Tirronen and Martti Juhola,

ville.e.tirronen@jyu.fi, Martti.Juhola@cs.uta.fi,

Tuomas Turpeinen and Kristian Meissner

tuomas.turpeinen@jyu.fi, Kristian.Meissner@ymparisto.fi

Abstract-- Aquatic ecosystems are continuously threatened by a growing number of human induced changes. Macroinvertebrate biomonitoring is particularly efficient in pinpointing the cause-effect structure between slow and subtle changes and their detrimental consequences in aquatic ecosystems. The greatest obstacle to implementing efficient biomonitoring is currently the cost-intensive human expert taxonomic identification of samples. While there is evidence that automated recognition techniques can match human taxa identification accuracy at greatly reduced costs, so far the development of automated identification techniques for aquatic organisms has been minimal. In this paper, we focus on advancing classification and data retrieval that are instrumental when processing large macroinvertebrate image datasets. To accomplish this for routine biomonitoring, in this paper we shall investigate the feasibility of automated river macroinvertebrate classification and retrieval with high precision. Besides the state-of-the-art classifiers such as Support Vector Machines (SVMs) and Bayesian Classifiers (BCs), the focus is particularly drawn on feed-forward artificial neural networks (ANNs), namely multilayer perceptrons (MLPs) and radial basis function networks (RBFNs). Since both ANN types have been proclaimed superior by different investigations even for the same benchmark problems, we shall first show that the main reason for this ambiguity lies in the static and rather poor comparison methodologies applied in most earlier works. Especially the most common drawback occurs due to the limited evaluation of the ANN performances over just one or few network architecture(s). Therefore, in this study, an extensive evaluation of each classifier performance over an ANN architecture space is performed. The best classifier among all, which is trained over a dataset of river macroinvertebrate specimens, is then used in the MUVIS framework for the efficient search and retrieval of particular macroinvertebrate peculiars. Classification and retrieval results present such a high accuracy that can match experts' ability for taxonomic identification.

Index Terms-- Biomonitoring, classification, radial basis function networks, multilayer perceptrons, Bayesian Networks, Support Vector Machines, Benthic macroinvertebrate.

I. INTRODUCTION

IT is an unfortunate fact that aquatic ecosystems are facing a growing number of anthropogenic pressures operating at several temporal and spatial scales (e.g. eutrophication, global warming).

1
2
3 Well planned biomonitoring is essential to detect the cause-effect structure between the often
4
5 subtle pressures and their ecosystem consequences. The resulting growing global need to
6
7 implement more biomonitoring is apparent but due to the cost-intensive human expert taxonomic
8
9 identification of samples, this need cannot currently be met. Automatic and semi-automatic
10
11 signal and image processing techniques have been successfully applied in similar fields of
12
13 application to solve such challenges. For instance, automatic image recognition techniques of
14
15 aquatic phytoplankton have been shown to match human taxa identification accuracy at a greatly
16
17 reduced cost [1]. Despite their obvious potential, the development of automated taxa
18
19 identification techniques has long been hampered by the reluctance of taxonomic experts to
20
21 embrace alternative methods of taxa identification. A detailed review on advances in automated
22
23 taxa identification [2] deemed misconceptions, the lack of vision and the lack of enterprise more
24
25 limiting to the development of automated taxa identification than actual practical constraints.
26
27 Research on automated recognition of aquatic organisms has mainly concentrated on plankton
28
29 [3], while automated classification and particularly retrieval of freshwater macroinvertebrates has
30
31 received little attention [4]. In a recent work based on SVMs [5] on a set of river
32
33 macroinvertebrates, mean correct classification rates of 88.2% and 75.3% have been achieved in
34
35 training and test sets, which match the levels of human accuracy for other aquatic taxonomic
36
37 groups [6].
38
39
40
41
42
43
44
45
46
47

48 In this paper, the primary goal is to develop a low-cost, automated and accurate benthic
49
50 macroinvertebrate classification and retrieval system for routine biomonitoring. In order to
51
52 accomplish this we shall investigate the state-of-the-art classifiers such as Support Vector
53
54 Machines (SVMs), Bayesian classifiers (BCs) and two most common feed-forward artificial
55
56
57
58
59

60 ¹ This work was supported by the Academy of Finland, project No. 213462 (Finnish Centre of Excellence Program (2006 - 2011))
61
62
63
64
65

1
2
3 neural networks (ANNs): multilayer perceptrons (MLPs) and radial basis function networks
4
5 (RBFNs). SVMs are known to be efficient classifiers, i.e. they can classify test cases with a high
6
7 accuracy. BCs are simple to implement, and they are often efficient classifiers, i.e. they classify
8
9 test cases almost as accurately as more sophisticated and complicated classifiers, see e.g. [7].
10
11 Particular focus will be drawn on feed-forward ANNs since earlier work on classification of
12
13 aquatic organisms has shown that neural networks usually outperform decision trees [8] and
14
15 other classical statistical techniques [1]. ANNs have proven to perform complex classification
16
17 tasks, provided that a proper structure for the network is selected and a suitable training
18
19 technique is applied to a sufficiently representative set of data. In order to achieve the highest
20
21 retrieval performance possible, among all classifiers investigated, the best classifier, which
22
23 achieves the minimum classification error (CE) in the test (validation) set, thus showing the
24
25 utmost generalization ability, will then be used for retrieval within a Benthic macroinvertebrate
26
27 image database. Finally, in order to achieve a low-cost application and avoid computationally
28
29 extensive features, we use only a simple and basic feature extraction technique composed of
30
31 mainly geometrical and statistical features. In this way the efficiency of the classifiers will also
32
33 be clearly investigated and tested against the limited discrimination power of those simple
34
35 features.
36
37
38
39
40
41
42
43
44

45
46 A certain ambiguity arises when comparing the performance of ANNs or to search for the best
47
48 ANN type or configuration among many alternatives. For instance most of the biomedical
49
50 applications of ANNs use the well-known MLPs [9]. Some promising results have also been
51
52 gained with another basic type of feed-forward ANN, the RBFNs [10]. Compared to MLPs, the
53
54 internal operation of RBFNs is easier to comprehend and faster training algorithms are available
55
56 for them [11]. However, the use of RBFNs for practical purposes is still quite limited. One of the
57
58
59
60
61
62
63
64
65

1
2
3 reasons is certainly that the comparison results between the two ANN types are varying and
4
5 sometimes even contradictory. Several articles report better results with RBFNs than MLPs with
6
7 applications in function approximation [12], image segmentation [13], speaker recognition [14],
8
9 modeling of rating curves in hydrology [15], etc. Yet some articles have found MLPs to perform
10
11 better than RBFNs when applied to, for example, remote sensing classification [16] and
12
13 automatic speech recognition [17]. Several comparisons of MLPs and RBFNs have also been
14
15 conducted within the biomedical domain, but the results are equally varying.
16
17
18
19
20

21 To address this problem, we shall first show that the major cause for such inconsistencies is
22
23 due to the fact that the evaluation methodologies used are limited or rather deficient and often
24
25 biased. For instance, in most comparative evaluations of MLPs and RBFNs, only one or few
26
27 network architecture(s) are considered. We shall demonstrate that there are significant variations
28
29 between performances of different architectures especially with MLPs. If this simple fact is
30
31 omitted, it can lead to either of the ANN types to be better on nearly any problem. Moreover, the
32
33 training method for both network types, the well-known back-propagation (BP) yields
34
35 significantly varying network parameters (weights and biases) after each training session and thus
36
37 a limited number of training runs cannot yield statistically significant performance measures and
38
39 accurate evaluations. In order to clarify such varying or even contradictory results of the previous
40
41 studies, we first propose an accurate and in-depth performance assessment approach for
42
43 comparing these two common network types. In the current work, we evaluate the classification
44
45 error (CE) over the training and the test sets and as performance criteria we use the *mean* and the
46
47 *minimum* errors that an ANN classifier of a particular type can achieve. We also analyze the
48
49 variations of these errors among different architectures for each network type in order to
50
51 investigate the dependence of the classification performance on the variations of the network
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3 configurations. In the proposed assessment scheme, to avoid the bias of the network architecture
4
5
6 used, rather than focusing on only one or few architectures, we propose that the evaluations are
7
8
9 performed over an architecture space containing a large variety of ANNs, e.g. from the simplest
10
11 single-layer perceptrons (SLPs) to the MLPs with several hidden layers and several neurons; and
12
13
14 similarly, from the simplest RBFNs with few hidden neurons to networks with many hidden
15
16 neurons.

17
18 Following performance evaluation of ANN classifiers using the proposed assessment
19
20 methodology and comparison with the other two common classifier types (SVMs and BCs), the
21
22
23 best classifier, which yields the best generalization capability (the lowest CE in the
24
25
26 test/validation set) is then integrated into the search engine of the MUVIS [18] and used for the
27
28
29 (dis-) similarity distance computation within the similarity-based queries. In this way we shall
30
31 demonstrate that the retrieval performance can be significantly improved, particularly as
32
33
34 compared to the traditional query methods.

35
36 The rest of the paper is organized as follows. Section II surveys the related work on all
37
38
39 classifiers used in the present work and discusses briefly their training methods. The proposed
40
41
42 approach for feature extraction, classification and retrieval in macroinvertebrate databases with
43
44
45 the detailed discussion over the proposed assessment system for evaluating ANN classifiers are
46
47
48 presented in Section III. Section IV provides the classification and retrieval experiments
49
50
51 conducted over Benthic macroinvertebrate image databases and discusses the results. Finally,
52
53
54
55
56
57
58
59
60
61
62
63
64
65 Section V concludes the paper and proposes topics for future research.

II. RELATED WORK

A. Support Vector Machines

SVMs are a modern computational data analysis method to classify data mainly to two classes [19] - [21]. The idea is to find a hyperplane that optimally separates a given dataset, i.e., has the maximal margin between the data points and the hyperplane. In the linear case this is illustrated in Figure 1. To handle non-linearly separable cases the so called “*kernel trick*” is used. Instead of finding the hyperplane directly in the given feature space, a nonlinear mapping (transformation) is applied to the data. In this usually higher dimensional space it is possible to build a linear hyperplane to separate two classes optimally instead of possibly poorer classification opportunities in the original dimension (e.g. see conceptual illustration in Figure 2). This results in a non-linear classifier in the original feature space.

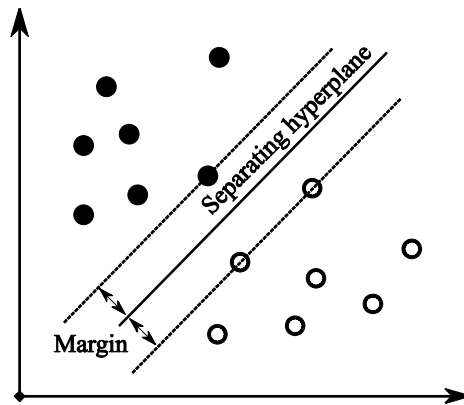


Figure 1: Formation of the linear optimally separating hyperplane.

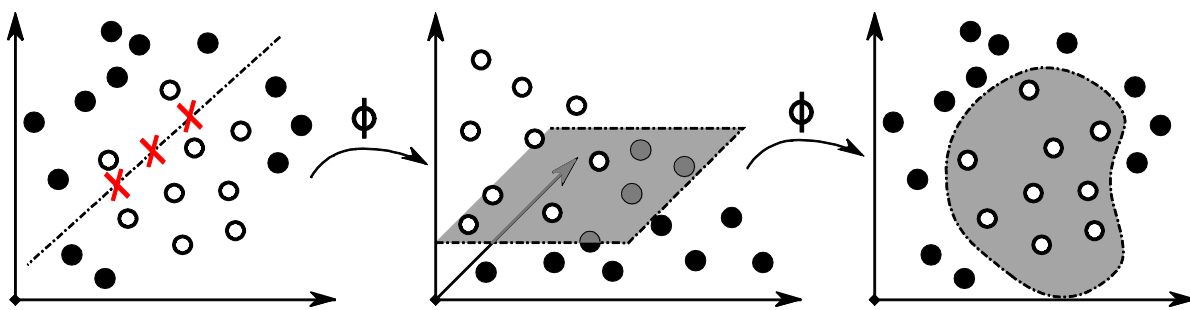


Figure 2: Non-separable sets and transformation to a non-linear classifier.

1
2
3
4 Classes refer to different macroinvertebrate species in the current context. Since this is a
5
6 multiclass problem, (i.e., there are more than two classes) SVMs have to be extended to handle
7
8 such problems. Up to date multiclass classification with SVMs is an ongoing topic in the
9
10 research community. There are two main approaches for multiclass extensions. One approach is
11
12 based on training several binary classifiers and combining their results. It is possible to divide the
13
14 training data of c classes to c sub-problems so that the training data of each class is separated
15
16 from the union of the other $c-1$ classes and this dichotomy is alternately repeated for all c
17
18 divisions. A separate SVM is then trained for each of c divisions. The other approach considers
19
20 all the data at once in a single optimization task (e.g. see [22] and [23]). In [24] the former
21
22 approach was found slightly more practical.
23
24
25
26
27

28
29 In this work we apply SVM implementation as described in [25] that uses the following “*one-*
30
31 *against-one*” methodology (also described in [26]) for multiclass problems. The training
32
33 problem is divided into $c(c-1)/2$ sub-problems so that each pair of classes is used to train one
34
35 classifier that separates those two classes. In classification a voting strategy is used. Each of the
36
37 $c(c-1)/2$ classifiers is applied to the data (feature) point x which is then assigned to the class with
38
39 the majority of the resulting predictions.
40
41
42
43

44 *B. Bayesian Classifiers*

45
46
47 Bayesian classifier (BC) is a traditional method for classifying data of several classes using
48
49 features of class individuals and prior knowledge of proportions of each class. The rules on
50
51 which classification decisions are based are expressed in terms of probabilities [19].
52
53 Mathematically speaking, we consider the class w of an individual to be a random variable
54
55 having values w_1, \dots, w_k and corresponding probabilities $P(w_1), \dots, P(w_k)$, which are called *prior*
56
57 probabilities. Furthermore, features of each class, $x = \{x_1, \dots, x_p\}$, are assumed to be generated by
58
59
60
61
62
63
64
65

1
2
3 a statistical model, whose distribution, in the case of continuous features, is expressed by the
4 probability density function (or *likelihood*) $f(x|w_i)$. This model can, for example, be a
5
6 multinormal distribution. Our interest is focused on the *posterior* probability, that is the
7 probability of each class given the features x of an individual. According to *Bayes'* rule, this
8
9 posterior is formulated as,
10
11
12
13
14
15

$$16 \quad P(w_i|x) = \frac{f(x|w_i)P(w_i)}{f(x)}, \quad (1)$$

17
18 where $f(x)$ is the probability density function of features x . An integral part of the *posterior* is
19 the product of the likelihood and the prior while $f(x)$ can be interpreted as a scaling factor. When
20 making the decision of the class of an individual with its features x , the natural choice is to
21 choose the class for which the posterior $P(w_i|x)$ is maximum. This is the so-called *Bayesian*
22 decision rule and it can be shown that it minimizes the average error probability. When training
23 the *Bayesian* classifier, the statistical model of features needs to be estimated from the training
24 samples. For instance in the case of *Gaussian* features the *Bayesian* decision rule of classification
25 is simply reached by estimating the mean and covariance matrix for each class that are extracted
26 from the training samples.
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43

44 C. Artificial Neural Networks

45
46 An ANN consists of a set of connected processing units, usually called neurons or nodes.
47 ANNs can be described as directed graphs, where each node performs some activation function
48 to its inputs and then gives the result forward to be the input of some other neurons until the
49 output neurons are reached. ANNs can be divided into feed-forward and recurrent networks
50 according to their connectivity. In a recurrent ANN there can be backward loops in the network
51 structure, while in feed-forward ANNs such loops are not allowed. Furthermore, feed-forward
52
53
54
55
56
57
58
59
60
61
62
63
64
65

ANNs are usually organized into layers of parallel neurons and only connections between adjacent layers are possible. All layers besides the input and output layers are called *hidden* layers. The input layer is just a passive layer, where no computations are carried out and it is not counted to the total number of layers. The active neurons perform an activation function f of the form,

$$y_k^{p,l} = f \left(\sum_{j=1}^{N^{l-1}} w_{jk}^l y_j^{p,l-1} - \theta_k^l \right), \quad (2)$$

where $y_k^{p,l}$ is the output of neuron k of layer l , when pattern p is fed to the ANN, N^{l-1} is the total number of neurons in layer $l-1$, w_{jk}^l is the connection weight between neuron j in layer $l-1$ and neuron k in layer l , θ_k^l is the bias of the neuron k in layer l . For the first processing layer (the layer right after the input layer) $y_j^{p,l-1} = y_j^{p,0}$ is naturally the j^{th} dimension of the input pattern $x^p = \{x_1^p, \dots, x_j^p, \dots, x_{N_i}^p\}$. The number of input neurons N_i and the number of output neurons N_o for ANNs are defined by the problem, while the number of hidden layers and the number of neurons in each hidden layer is somehow decided usually by an expert and with respect to the problem. A sample feed-forward ANN is illustrated in Figure 3. It has three layers (two hidden layers and the output layer). Figure 3 also shows the connection weights w_{j1}^2 and the bias θ_1^2 for the first neuron in layer 2.

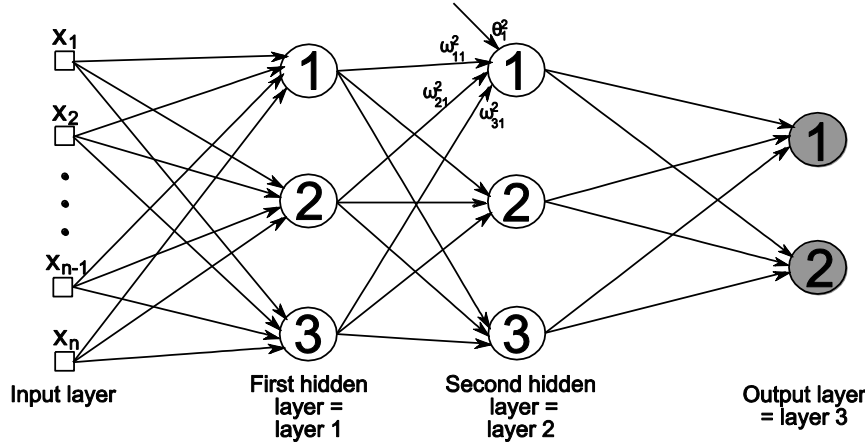


Figure 3: An example of fully-connected feed-forward ANN.

The most common ANN type is the multilayer perceptron (MLP) [9]. It is a feed-forward network, which contains one or more layers of hidden neurons. The degree of neuron connectivity is usually high and the neurons have nonlinear activation functions, but the nonlinearity is smooth (differentiable everywhere). The use of nonlinear activation functions is essential because otherwise the MLP could always be reduced to a single-layer perceptron (SLP) without changing its capabilities. A commonly used activation function is the tangent hyperbolic (tanh) function which is defined as follows:

$$y_k^{p,l} = \tanh(v_k^{p,l}) = \frac{e^{(v_k^{p,l})} - e^{-(v_k^{p,l})}}{e^{(v_k^{p,l})} + e^{-(v_k^{p,l})}} \quad \text{where} \quad v_k^{p,l} = \sum_{j=1}^{N^{l-1}} w_{jk}^l y_j^{p,l-1} - \theta_k^l \quad (3)$$

Another popular type of feed-forward ANN is the radial basis function (RBF) network [10], which has always two layers in addition to the passive input layer: a hidden layer of RBF units and a linear output layer. Only the output layer has connection weights and biases. The activation function of the k^{th} RBF unit is defined as

$$y_k = \varphi\left(\frac{\|X - \mu_k\|}{\sigma_k}\right), \quad (4)$$

where φ is a radial basis function or, in other words, a strictly positive radially symmetric function, which has a unique maximum at N -dimensional center μ_k and whose value drops

1
2
3 rapidly close to zero away from the center. σ_k is the width of the peak around the center μ_k . The
4
5
6 activation function gets noteworthy values only when the distance between the N -dimensional
7
8
9 input X and the center μ_k , $\|X - \mu_k\|$, is smaller than the width σ_k . The most commonly used
10
11
12 activation function in RBFNs is the *Gaussian* basis function defined as,

$$y_k = \exp\left(-\frac{\|X - \mu_k\|^2}{2\sigma_k^2}\right), \quad (5)$$

13
14
15 where μ_k and σ_k are the mean and standard deviation, respectively, and $\| \cdot \|$ denotes the
16
17
18 *Euclidean* norm. More detailed information about MLPs and RBFNs can be obtained from [9].
19
20
21

22 23 24 *D. The Back-propagation Algorithm*

25
26
27 Back-propagation (BP) [27] is the most commonly used training technique for feed-forward
28
29
30 ANNs. It is a powerful supervised training technique which has been used in pattern recognition
31
32
33 and classification problems in many application areas. BP has the advantage of applying directed
34
35
36 search and has strong local search ability. However, BP is just a gradient descent algorithm in the
37
38
39 error space, which can be complex and may contain many deceiving local minima (multi-modal).
40
41
42 Therefore, BP gets most likely trapped into a local minimum, making it entirely dependent on the
43
44
45 initial (weight) settings. There are many BP variants and extensions trying to address this
46
47
48 problem, [9], [28], [29], yet the performance and computational cost of each algorithm varies
49
50
51 with respect to the problem at hand; and the question of which ANN architecture (number of
52
53
54 layers and interconnections, number of nodes, etc.) should be used for a particular problem still
55
56
57 remains unanswered. The BP algorithm can be summarized as follows:

- 58
59
60 1. Initialize the weights w_{jk}^l and biases θ_k^l randomly. For RBF-networks initialize also the peak
61
62
63 centers μ_k and σ_k .
- 64
65 2. Feed pattern p to the network and compute the output $y_k^{p,l}$ of each neuron.

3. Calculate the error between the computed output $y_k^{p,o}$ of each output neuron and the desired output t_k^p as $e_k^{p,o} = t_k^p - y_k^{p,o}$.

4. For each neuron k , calculate the partial derivatives $\frac{\partial E^p}{\partial h_k^l}$, where E^p is the total error energy defined as $E^p = \frac{1}{2} \sum_{k \in o} (e_k^{p,o})^2$ and h_k^l is a uniform symbol for each parameter $w_{jk}^l, \theta_k^l, \mu_k$ and σ_k .

The name of the back-propagation algorithm comes from the fact that one starts to calculate the local gradients from the output layer and then iteratively proceed backwards toward the input layer. The formulas for calculating the local gradients for MLPs and RBFNs can be found in [9] and [30], respectively.

5. Update the parameters as follows:

$$h_k^l(t+1) = h_k^l(t) - \eta \frac{\partial E^p}{\partial h_k^l} \quad (6)$$

where η is the learning rate parameter.

6. Repeat steps 2-5 until some stopping criteria is reached.

One complete presentation of the training set is called an *epoch*. Usually many *epochs* are required to obtain the best training results, but, on the other hand, too many training *epochs* can lead to over-fitting. In the above realization of the BP algorithm the network parameters are updated after every training sample. This is called the *online* or *sequential* mode. The other possibility is the *batch* mode, where all the training samples are first presented to the network

and then the parameters are adjusted so that the total training error $E = \frac{1}{2} \sum_{p=1}^P \sum_{k \in o} (e_k^{p,o})^2$, where P

is the number of training samples, is minimized instead of E^p . The *sequential* mode is often favored over the *batch* mode as it requires less storage space. Moreover, the *sequential* mode is less likely to get trapped in a local minimum as updates at every training sample make the search

1
2
3 stochastic in nature. However, the stochastic aspects also make it difficult to derive theoretical
4
5
6 conditions for the convergence of the *sequential* BP mode, while for the *batch* mode the
7
8
9 convergence to a (local) minimum can be guaranteed under simple conditions. Also most of the
10
11 existing improvement strategies are based on the *batch* mode.
12

13
14 In this study we use *sequential* BP mode for MLP training and SuperSAB enhancement [31] of
15
16 the BP algorithm when training RBFNs. The main difference in SuperSAB is that it modifies the
17
18 update-values for each parameter according to the sequence of signs of the partial derivatives.
19
20 This only leads to a faster convergence, while the problems of a hill-climbing algorithm are not
21
22 solved. Further details about BP and SuperSAB can be found in [31] and [32], respectively.
23
24
25
26
27
28
29

30 III. CLASSIFICATION AND RETRIEVAL IN MACROINVERTEBRATE DATABASES

31 32 33 A. Dataset Creation and Feature Extraction

34
35
36 The Benthic macroinvertebrate image dataset used in this work consists of 1350 images
37
38 representing 8 different taxonomical groups: *Baetis rhodani*, *Diura nanseni*, *Heptagenia*
39
40 *sulphurea*, *Hydropsyche pellucidula*, *Hydropsyche siltalai*, *Isoperla sp.*, *Rhyacophila nubila* and
41
42 *Taeniopteryx nebulosa*. Members from the same taxonomical group were imaged by a flatbed
43
44 scanner, digitized, normalized and eventually each macroinvertebrate in each scan was saved as
45
46 an individual image. Three individuals from four taxonomical classes are shown in Figure 4 and
47
48 demonstrate some crucial properties of the data: specimens are semi-rigid so that the actual shape
49
50 may vary from one sample to another. Furthermore, there can be overlapping, repetitions,
51
52 rotations, scaling and variations in the intensity levels, all of which make the classification
53
54 problem even more challenging and the need for a powerful classifier is imminent to match the
55
56 accuracy of expert-level human classification.
57
58
59
60
61
62
63
64
65

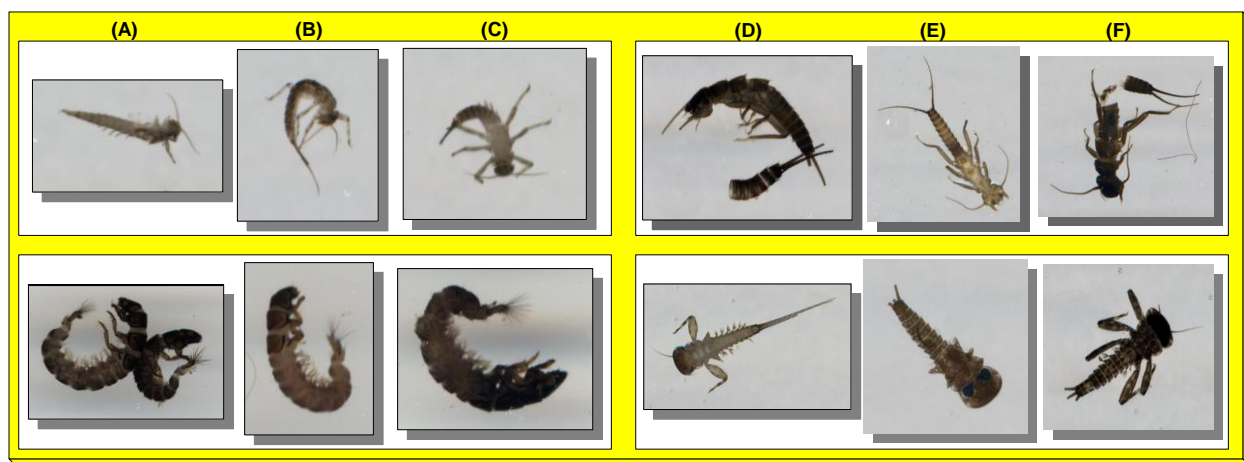


Figure 4: Three samples from *Baetis rhodani* (top: A, B, C), *Diura nanseni* (top: D, E, F), *Hydropsyche pellucidula* (bottom: A, B, C) and *Heptagenia sulphurea* (bottom: D, E, F) classes.

Feature extraction and selection depend on the data and the classifier used. As such, features can be seen as a part of the classifier system itself. However, such feature selection in case of images has remained empirical science as there are many possible features and an enormous amount of their combinations. Examples of classical features are various edge, curve, ridge, blob, and corner based features, shape descriptors [33] such as various moments and Fourier descriptors, simple textural features such as histograms of intensity, gradient [5] and gray-scale co-occurrences [34]. Current state-of-the-art feature extraction methods include Local Binary Patterns [35], Gabor packet based methods [36], Co-occurrence matrices [37], scale invariant features of SIFT algorithm [38] and various other orientation based features such as in [39].

Among all these possibilities, in order to achieve a low-cost solution and to demonstrate the efficiency of the proposed classifier for similarity-based retrieval, we have applied a simple and basic feature extraction technique composed of mainly geometrical and statistical features. 15-D features of each macroinvertebrate image are extracted by using *ImageJ* [40], which is a public domain, Java-based image processing program. The following set of 15 features are selected by using *ImageJ*'s built in measurement and analysis functions: pixel value (grayscale) statistics $\{\mu, \sigma, Mode, Median, IntDen, Kurtosis, Skewness\}$ and geometric features $\{Area, Perimeter,$

1
2
3 *Width, Height, Ferret, Major, Minor, Circularity*}, for which *ImageJ* performs a simple
4
5
6 thresholding operation to extract the binary mask of each macroinvertebrate sample. The detailed
7
8 description of these features can be found in [40]. In a pre-processing step, each feature vector is
9
10 then normalized to have a zero mean and linearly scaled into [-1, 1] interval before being
11
12 presented to the classifier.
13
14

15 16 17 *B. The Assessment Methodology for Feed-forward ANNs*

18
19 In order to objectively assess the performance of MLPs and RBFNs, we apply exhaustive BP
20
21 training over a wide variety of network architectures whilst keeping in mind that too complicated
22
23 configurations may not be applicable in practice. In this way we can avoid the bias or possible
24
25 effect of a particular network on the performance, while many of the aforementioned studies
26
27 failed to do so as they were mostly performed using only one or few fixed network
28
29 architecture(s). For training, the tangent hyperbolic activation function given in Eq. (3) was used
30
31 with MLPs while the *Gaussian* basis function given in Eq. (5) was used with RBFNs.
32
33
34
35
36

37 The architecture space for MLPs may be defined over a wide range of configurations, i.e. say
38
39 from a SLP to complex MLPs with many hidden layers and neurons. Suppose that a range is
40
41 defined for the number of layers, $\{L_{\min}, L_{\max}\}$ and another for the number of neurons for each
42
43 layer l , $\{N_{\min}^l, N_{\max}^l\}$. Consequently, the architecture space can now be defined with only two
44
45 arrays, $R_{\min} = \{N_I, N_{\min}^1, \dots, N_{\min}^{L_{\max}-1}, N_O\}$ and $R_{\max} = \{N_I, N_{\max}^1, \dots, N_{\max}^{L_{\max}-1}, N_O\}$, one for minimum
46
47 and the other for maximum number of neurons allowed for each layer l of a MLP. The size of
48
49 both arrays is naturally $L_{\max} + 1$, where the corresponding entries define the range of neurons
50
51 possible on the l^{th} hidden layer for all those MLP configurations, which have an l^{th} hidden layer.
52
53
54
55
56
57
58
59 $L_{\min} \geq 1$ and $L_{\max} \geq L_{\min}$ can be set to any value meaningful for the problem at hand. The size of
60
61
62
63
64
65

the input and output layers, $\{N_i, N_o\}$, are defined by the problem (e.g. $\{N_i, N_o\} = \{15, 8\}$ for the current work) and remains the same for all configurations in the architecture space, while the geometric pyramid rule (GPR) [41] may for instance be used as a rough guideline for setting the range for the number of neurons in each hidden layer. All network configurations in the architecture space are then enumerated into a hash table with a proper hash function, which basically ranks the configurations with respect to their complexity, i.e. it associates higher hash indices to architectures with higher complexity. The hash indices start from the simplest SLP and proceed to the most complex network configuration with $L_{\max} - 1$ hidden layers, each of which has the maximum number of neurons given in R_{\max} .

Table I. Sample MLP architecture space containing 49 MLP configurations.

Ind.	Configuration	Ind.	Configuration	Ind.	Configuration
0	15 x 8	16	15 x 15 x 4 x 8	32	15 x 15 x 6 x 8
1	15 x 8 x 8	17	15 x 8 x 5 x 8	33	15 x 8 x 7 x 8
2	15 x 9 x 8	18	15 x 9 x 5 x 8	34	15 x 9 x 7 x 8
3	15 x 10 x 8	19	15 x 10 x 5 x 8	35	15 x 10 x 7 x 8
4	15 x 11 x 8	20	15 x 11 x 5 x 8	36	15 x 11 x 7 x 8
5	15 x 12 x 8	21	15 x 12 x 5 x 8	37	15 x 12 x 7 x 8
6	15 x 13 x 8	22	15 x 13 x 5 x 8	38	15 x 13 x 7 x 8
7	15 x 14 x 8	23	15 x 14 x 5 x 8	39	15 x 14 x 7 x 8
8	15 x 15 x 8	24	15 x 15 x 5 x 8	40	15 x 15 x 7 x 8
9	15 x 8 x 4 x 8	25	15 x 8 x 6 x 8	41	15 x 8 x 8 x 8
10	15 x 9 x 4 x 8	26	15 x 9 x 6 x 8	42	15 x 9 x 8 x 8
11	15 x 10 x 4 x 8	27	15 x 10 x 6 x 8	43	15 x 10 x 8 x 8
12	15 x 11 x 4 x 8	28	15 x 11 x 6 x 8	44	15 x 11 x 8 x 8
13	15 x 12 x 4 x 8	29	15 x 12 x 6 x 8	45	15 x 12 x 8 x 8
14	15 x 13 x 4 x 8	30	15 x 13 x 6 x 8	46	15 x 13 x 8 x 8
15	15 x 14 x 4 x 8	31	15 x 14 x 6 x 8	47	15 x 14 x 8 x 8
				48	15 x 15 x 8 x 8

Take, for instance, the following range arrays, $R_{\min} = \{N_i, 8, 4, N_o\}$ and $R_{\max} = \{N_i, 15, 8, N_o\}$, which indicate that $L_{\max} = 3$. If $L_{\min} = 1$ then the hash function enumerates all MLP configurations in the architecture space as shown in Table I for $\{N_i, N_o\} = \{15, 8\}$. The hash

1
2
3 function associates index 0 with the simplest network configuration, which is the SLP with no
4
5 hidden layers. From indices 1 to 8, all configurations belong to 2-layer MLPs with 8 to 15
6
7 neurons in the single hidden layer (as specified in the 2nd entries of arrays R_{\min} and R_{\max}).
8
9 Similarly for indices 9 and up, 3-layer MLPs are enumerated in which the number of neurons in
10
11 the 1st and 2nd hidden layers is varied according to the corresponding entries in R_{\min} and R_{\max} .
12
13 Finally, the most complex MLP with the largest possible number of layers and the highest
14
15 number of neurons in each layer is associated with the highest index, $d=48$. Therefore, all 49
16
17 entries in the hash table span the architecture space with respect to the configuration complexity.
18
19
20
21
22

23
24 Since there is only one hidden layer for RBFNs, the number of hidden neurons can be directly
25
26 used for the architecture space indexing. In this work, we used an architecture space consisting of
27
28 26 RBFNs containing 10-35 hidden neurons.
29
30
31

32 *C. Training of the Classifiers*

33
34 Recall that in the Benthic macroinvertebrate image dataset used in this work, 15-D features are
35
36 extracted per image and there are 8 classes, which make the input and output layer sizes as
37
38 $\{N_I, N_O\} = \{15, 8\}$. In order to evaluate the effect of the data partitioning, we created 10 different
39
40 training and test partitions over the entire dataset, each with randomly chosen 650 and 700
41
42 samples, respectively. Therefore, in each partition the training and test sets contain a random
43
44 shuffle of images whilst the test set has the majority (700) of the dataset. Figure 5 presents an
45
46 overview of classifier training for a particular training set. In order to accomplish an exhaustive
47
48 evaluation of ANN classifiers, 10 independent runs are performed with random parameter
49
50 initialization to compute the error statistics for each dataset partition and furthermore, the
51
52 training process is performed for each configuration in the architecture spaces so as to
53
54 accomplish the proposed assessment methodology as explained earlier. We consider the training
55
56
57
58
59
60
61
62
63
64
65

CE and particularly the test CE, which is the primary objective of the classifier as it shows the classification accuracy level achieved as well as the generalization capability of each network type. Both training and test CEs shall then be evaluated by considering the *mean* and the *minimum* errors achieved by each classifier.

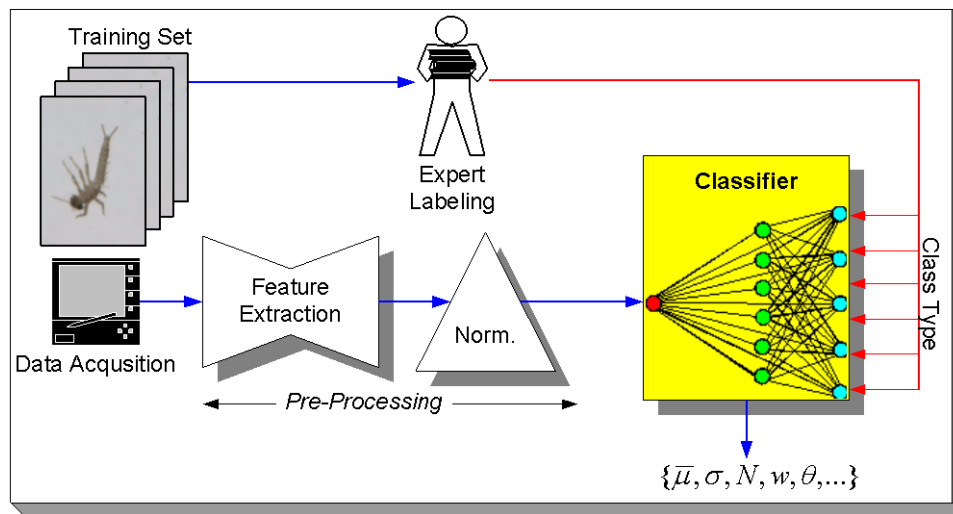


Figure 5: Overview of a classifier training.

For BP training, we have applied 10000 epochs for the training of both types of ANNs. The increase and decrease factors of BP (SuperSAB) operations over RBFNs are set to 1.05 and 0.25, respectively. The learning rate for MLP training is set to 0.002. Such parameters for BP (high iteration number and quite low learning rate / increase factor) are purposefully set to prevent oscillations and to ensure convergence to a local optimum.

The support vector machines are applied with RBF kernel and automatic parameter selection. The critical parameters for this kernel, C and γ , are selected using cross validation and parallel grid search. In cross validation the data is split into several folds. Then each fold is in turn considered as a validation set for the rest of the data. The parameter selection is done by testing the parameter space $C \times \gamma$ using a grid of, $C = 2^{-5}, 2^{-3}, \dots, 2^{15}$ and $\gamma = 2^{-15}, 2^{-13}, \dots, 2^3$. Parameters resulting in highest cross-validation accuracy are used for the training. This approach is similar to

what is described in [42].

Finally, BCs are trained in such a way that the priors are set to be equal for each class. As the statistical model of features the multinormal distribution is used when the classification rule is reached by estimating the mean and covariance matrix of each class from the training sample.

IV. EXPERIMENTAL RESULTS

A. Classification Results

The classification performance evaluation of the SVMs and BCs are relatively straightforward since both techniques have *deterministic* training methods, which do not depend on (random) parameter initialization. So the training and test CEs per partition, as enlisted in Table 2, can directly be used for comparative evaluations. It is clear that both classifiers show comparable performance over the test sets. Yet minimum training CEs are achieved only by SVMs performing consistently better classification results than BCs within the training sets.

Table 2: Train and test classification error statistics for SVMs and BCs per dataset partition. The best (minimum) error statistics are highlighted.

Partitions	Training CE		Test CE	
	SVM	BC	SVM	BC
Par-1	0.0108	0.0308	0.0729	0.0571
Par-2	0.0169	0.0338	0.0486	0.0714
Par-3	0.0123	0.0369	0.0571	0.0714
Par-4	0.0123	0.0492	0.0786	0.0657
Par-5	0.0092	0.0338	0.07	0.0786
Par-6	0.0108	0.0338	0.0614	0.0728
Par-7	0.0077	0.0276	0.0643	0.0786
Par-8	0.0077	0.0323	0.06	0.06
Par-9	0.0108	0.0338	0.0529	0.0728
Par-10	0.0123	0.04	0.0514	0.0728

In order to perform a comprehensive and systematic assessment of the performance of ANN classifiers, we plot the error statistics (both training and test CEs) *versus* hash indices for each

configuration in the architecture space(s). Note that a hash index simply corresponds to the number of hidden (*Gaussian*) neurons for RBFNs and see Table I for the architecture space used for MLPs. Due to space limitations we present results only on two partitions (1 and 3).

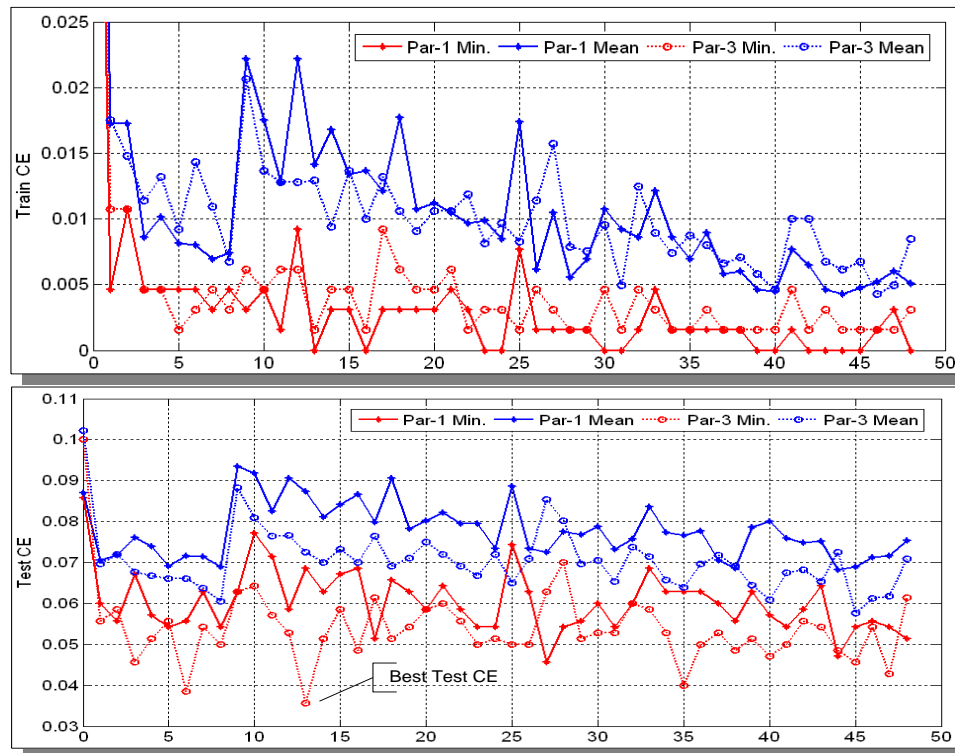


Figure 6: MLP Train (top) and test (bottom) CE statistics vs. hash index plots for partitions 1 and 3.

Figure 6 presents the training and test CE statistic plots obtained from MLP training over the dataset partitions 1 and 3. Both plots clearly demonstrate that the MLP classification performance significantly depends on the network configuration used since particularly the mean training CEs vary among different MLPs more than 15%. Note that several MLP configurations achieved 0% CE in the training set and less than 5% CE in the test set. Among all partitions, the best MLP configuration, which achieves the minimum test CE ($\sim 3.57\%$ corresponding to the train CE $\sim 0.77\%$) is obtained from the training of the partition 3, as indicated in the bottom plot of Figure 6 (with index 13, equivalent to 3-layer MLP configuration $15 \times 12 \times 4 \times 8$).

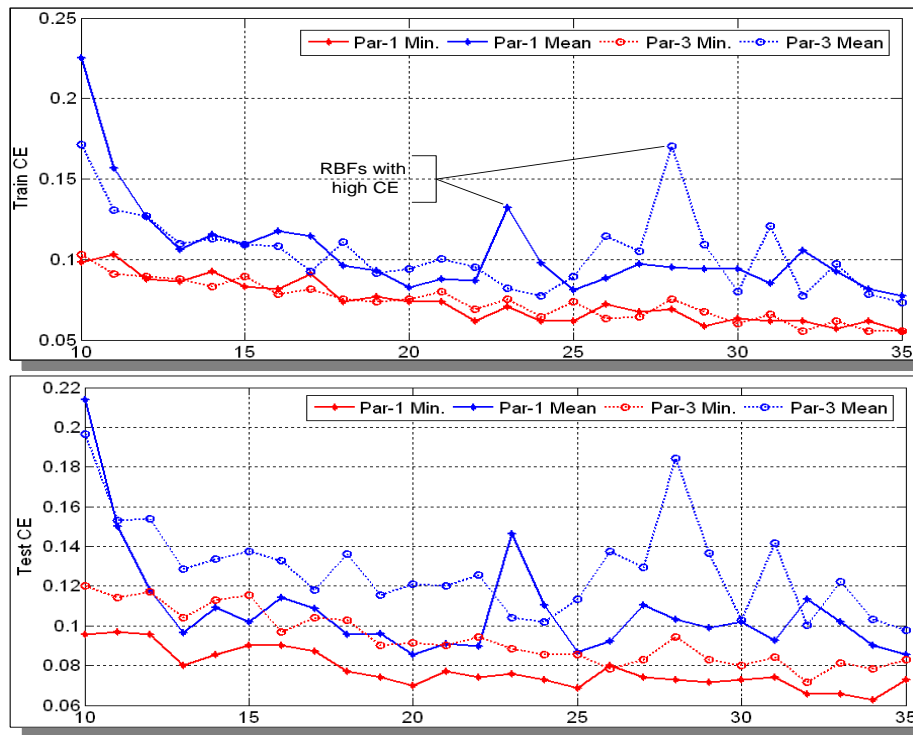


Figure 7: RBF train (top) and test (bottom) CE statistics vs. hash index plots for partitions 1 and 3.

Figure 7 presents the training and test CE statistics plots obtained from RBF training over the same partitions (1 and 3). Note that the effect of different dataset partitioning is quite visible here, i.e. different RBFN configurations yield the poorest classification performance (i.e. the two peaks in the CE plot), as shown in the top plot of Figure 7. It is also worth mentioning that compared to MLPs, RBFNs usually exhibit a monotonous and stable performance level, that is somewhat independent from the network configuration even though (both training and test) CEs get slightly lower as the network complexity (number of *Gaussian* neurons) rises. Yet RBFNs exhibit the *poorest* classification performance level among all classifiers. This is in fact an expected outcome since BP is just a gradient descent algorithm on the error space, and besides weights and biases (as these are the *only* parameters computed for MLPs), it furthermore computes 15-D centroids and variance *per Gaussian* neuron for RBFNs. In this case, the error surface obviously becomes extremely complex and contains massive amount of deceiving local minima. Therefore, BP in RBFN training most likely gets trapped into a local minimum *earlier*,

1
2
3 making its classification performance entirely dependent on the *initial* settings.
4

5
6 The overall classification results, first of all, show that training and test performances of both
7
8 ANN types depend on the network architecture used and the dataset partitioning applied, each of
9
10 which has *varying* effects on the two performance criteria employed. This justifies the use of the
11
12 proposed assessment technique for both ANN types, MLPs and RBFNs, over Benthic
13
14 macroinvertebrate image dataset. Among all classifiers, the overall best classification
15
16 performance is obtained with the MLP configuration mentioned earlier and it is therefore, used to
17
18 improve the retrieval results, as detailed next.
19
20
21
22
23

24 *B. Retrieval Results*

25
26 The retrieval process in MUVIS [43] is based on the traditional query by example (QBE)
27
28 operation. The features of the query item are used for (dis-) similarity measurement among all the
29
30 features of the visual items in the database. Ranking the database items according to their
31
32 similarity distances yields the retrieval result. The traditional (dis-) similarity measurement in
33
34 MUVIS is computed by applying a distance metric such as L2 (Euclidean) between the feature
35
36 vectors of the query and the (next) database item. So in Benthic macroinvertebrate image
37
38 database, this corresponds to computing the Euclidean distance between two 15-D feature
39
40 vectors. In order to obtain the highest retrieval performance, we have chosen the MLP classifier
41
42 with the best generalization ability (i.e. the 3-layer MLP, which achieved the overall minimum
43
44 test CE for partition-3, with the following configuration: $15 \times 12 \times 4 \times 8$). When the classifier is used,
45
46 the same (L2) distance metric is now applied to the class vectors at the output layer to compute
47
48 the (dis-) similarity distance between the query and the (next) database image. Since the classifier
49
50 has an elegant discrimination capability among different class members, in this way the retrieval
51
52 performance can further be improved.
53
54
55
56
57
58
59
60
61
62
63
64
65

The gold standard (or ground truth) data is available for the entire dataset we used. Basically a retrieved image is relevant if it has the same class type (macroinvertebrate type) as with the query image. For each query we used a search window, which is equal to the number of images belonging to the query's class. In order to measure the retrieval performance, we used an unbiased and limited formulation of *Normalized Modified Retrieval Rank* ($NMRR(q)$), which is defined in MPEG-7 as the retrieval performance criteria per query (q). It combines both the traditional hit-miss counters; *Precision* and *Recall*, and further takes the ranking information into account as given in the following expression:

$$AVR(q) = \frac{\sum_{k=1}^{N(q)} R(k)}{N(q)} \text{ and } W = 2N(q)$$

$$NMRR(q) = \frac{2AVR(q) - N(q) - 1}{2W - N(q) + 1} \leq 1 \quad (7)$$

$$ANMRR = \frac{\sum_{q=1}^Q NMRR(q)}{Q} \leq 1$$

where $N(q)$ is the minimum number of relevant (via *ground-truth* obtained by an expert in the field) images in a set of Q retrieval experiments, $R(k)$ is the rank of the k^{th} relevant retrieval within a window of W retrievals, which are taken into consideration during each query, q . If there are less than $N(q)$ relevant retrievals among W then a rank of $W+1$ is assigned for the remaining (missing) ones. $AVR(q)$ is the average rank obtained from the query, q . Note that if the first $N(q)$ retrievals are all relevant, then $NMRR(q)=0$, and the best retrieval performance is thus achieved. On the other hand, if no relevant item is retrieved among W then $NMRR(q)=1$, as the worst case. Therefore, the lower $NMRR(q)$ is, the better (more relevant) the retrieval for the query, q . $ANMRR$ is just the average of $NMRR$ scores for all queries. Along with $ANMRR$, we also used average precision (AP) measure and both measures are computed querying *all* (1350) images in the database within a retrieval window equal to the number of ground truth images, $N(q)$ for each

query q . This henceforth makes the AP identical to average $Recall$ and average $F1$ measures, too.

With the traditional approach (without classifier), we obtain $ANMRR = 0.4757$ and $AP = 0.4912$, indicating in fact a quite poor retrieval performance due to the limited discrimination power of the simple features used. With the use of the classifier, the retrieval performance has been improved to the level of $ANMRR = 0.0397$ and $AP = 0.9558$. This eventually presents such a delicate solution that matches and even may surpass the accuracy of human experts when identifying macroinvertebrate specimens.

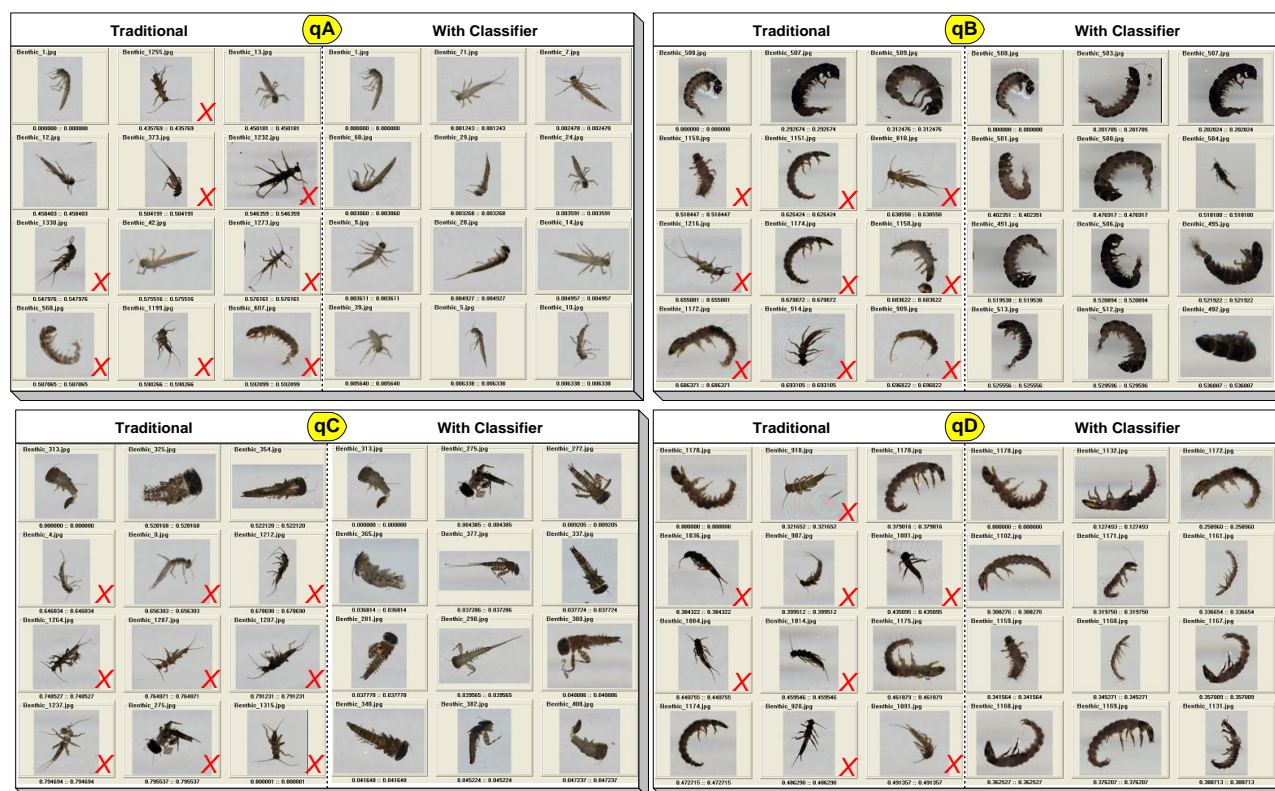


Figure 8: Four sample queries of *Baetis rhodani* (qA), *Hydropsyche pellucidula* (qB), *Heptagenia sulphurea* (qC) and *Rhyacophila nubila* (qD) with (right) and without (left) using classifier. Top-left is the query image as well as the first retrieved image. Each irrelevant retrieval is marked with a red 'X'

For visual evaluation, Figure 8 presents four typical retrieval results (from the snapshot of the first page of MUVIS-MBrowser application) with and without using the proposed classifier. Note that the retrieval results with the traditional approach are highly erroneous since many members of irrelevant classes are retrieved even within the first 12 ranks. With the use of classifier, all

1
2
3 retrievals in the first page (as well as in the second and third pages that are not shown therein due
4
5
6 to space limitations) are relevant.
7
8
9

10 11 V. CONCLUSIONS 12 13

14 In this paper, we addressed the problem of cost-intensive manual taxonomic classification and
15
16 retrieval of macroinvertebrate specimens by investigating the best classifier among powerful
17
18 state-of-the-art automatic classifiers. Among many alternatives, in order to demonstrate the
19
20 efficiency of the classifier and to propose a low cost solution, we have intentionally extracted
21
22 basic and simple features from the Benthic macroinvertebrate images. With the proper
23
24 normalization, the deterministic classifiers, SVMs and BCs achieved training and test CEs much
25
26 lower than 10%. Particularly SVMs performed quite well in the training sets, i.e. ~1% CE.
27
28
29
30
31

32 The performance of both types of ANN classifiers was evaluated by using a novel assessment
33
34 methodology, which can evaluate training (error minimization) and test (generalization)
35
36 performances of both ANN types, MLPs and RBFNs, with respect to different network
37
38 configurations whilst considering two statistical criteria, the *mean* and the *minimum* errors within
39
40 a certain number of training runs. The experimental results clearly demonstrate that both training
41
42 and generalization performance depend on the network configuration and the data partitioning
43
44 between training and test sets. The fact that most of the earlier comparative studies evaluate
45
46 network performances over only one or few network configurations, may lead to the erroneous
47
48 selection of either of the ANN types as the winner for any given problem. The proposed method
49
50 for performance assessment of ANN classifiers can also be used for other types of ANNs with
51
52 different training techniques and in other application areas. This is subject to our future work,
53
54 which will focus on comparison not only the ANN types but also different training methods such
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4 as BP versus evolutionary algorithms, particularly Genetic Algorithms and Particle Swarm
5
6 Optimization [44].
7

8
9 With the proposed in-depth assessment, the best classifier in terms of test (validation)
10
11 classification performance is obtained among MLPs and then used for the purpose of improving
12
13 similarity-based retrievals within the MUVIS framework. The retrieval results from the extensive
14
15 query experiments show that a high performance in retrieval accuracy is achieved, particularly
16
17 when compared to the traditional (without classifier) retrieval methodology.
18
19
20
21
22
23
24

25 REFERENCES

- 26 [1] P. F. Culverhouse, R. G. Simpson, R. Ellis, J. A. Lindley, R. Williams, T. Parisini, B. Reguera, I. Bravo, R.
27 Zoppoli, G. Earnshaw, H. McCall and G. Smith, "Automatic classification of field-collected dinoflagellates by
28 artificial neural network," *Marine Ecology Progress Series*, vol 139, pp.281-287, 1996.
29 [2] K.J. Gaston, and M.A. O'Neill, "Automated species identification: why not?," *Philosophical Transactions of the Royal*
30 *Society of London Series B*, vol. 359, pp. 655-667, 2004.
31 [3] M. Benfield, (and 14 others) "RAPID: Research on Automated Plankton Identification," *Oceanography*, vol. 20, no. 2, pp.
32 172-187, 2007.
33 [4] N. Larios, H. Deng and W. Zhang, "Automated insect identification through concatenated histograms of local appearance
34 features," *Machine Vision and Applications*, vol. 19, pp. 105-123, 2008.
35 [5] V. Tirronen, A. Caponio, T. Haanpää and K. Meissner, "Multiple order gradient feature for macroinvertebrate identification
36 using support vector machines," *Lecture Notes in Computer Science*, in Press.
37 [6] P.F. Culverhouse, R. Williams, B. Reguera, V. Herry and S. Gonzales-Gil, "Do experts make mistakes? A comparison of
38 human and machine identification of dinoflagellates," *Marine Ecology progress Series*, vol. 247, pp. 17-25, 2003.
39 [7] L. Bertelli, R. Cucchiara, G. Paternostro and A. Prati, "A semi-automatic system for segmentation of cardiac M-mode
40 images", *Pattern Analysis and Applications*, vol. 9, pp. 293-306, 2006.
41 [8] Y. P. Ginoris, A. L. Amaral, A. Nicolau, M. A. Z. Coelho and E. C. Ferreira "Recognition of protozoa and metazoa using
42 image analysis tools, discriminant analysis, neural networks and decision trees," *Analytica Chimica Acta*, vol. 595, pp. 160-
43 169, 2007.
44 [9] S. Haykin, "Neural Networks: a Comprehensive Foundation," Prentice hall, USA, June 1998.
45 [10] T. Poggio and F. Girosi, "A theory of networks for approximation and learning," A.I. Memo No. 1140, M.I.T. A.I Lab, 1989.
46 [11] Lu Yingwei, N. Sundararajan and P. Saratchandran, "Performance evaluation of a sequential minimal radial basis function
47 (RBF) neural network learning algorithm," *IEEE Transactions on Neural Networks*, vol. 9, no. 2, pp. 308-318, March 1998.
48 [12] L. Özyilmaz, T. Yildirim and K. Koklu, "Comparison of Neural Networks for function Approximation," *Pakistan Journal of*
49 *Applied Sciences*, vol. 2, no. 3, pp. 288-294, 2002.
50 [13] D. Kovacevic and S. Loncaric, "Radial Basis Function-based Image Segmentation using a Receptive Field," in Proc. of the
51 Tenth IEEE Symposium on Computer-Based Medical Systems, pp. 126-130, Slovenia, June 1997.
52 [14] R.A. Finan, A.T. Sapeluk and R.I. Dampier, "Comparison of Multilayer and Radial Basis Function Neural Networks for
53 Text-Dependent Speaker Recognition," in Proc. of IEEE Int. Conf. on Neural Networks, vol. 4, pp. 1992-1997, USA, June
54 1996.
55 [15] K.P. Sudheer and S. K. Jain, "Radial Basis Function Neural Network for Modeling Rating Curves," *Journal of Hydrologic*
56 *Engineering*, Vol. 8, No. 3, pp. 161-164 May/June 2003.
57 [16] S. Gopal and M. Fischer, "A Comparison of Three Neural Network Classifiers for Remote Sensing Classification," in Proc.
58 of International Geoscience and Remote Sensing Symposium IGARSS'96, vol. 1, pp. 787-789, USA, May 1996.
59 [17] B. A. Hawickhorst, S. A. Zahorian, and R. Rajagopal, "A Comparison of Three Neural Network Architectures for Automatic
60 Speech Recognition," in Proc. of the Artificial Neural Networks in Engineering Conference, Intelligent Engineering Systems
61 Through Artificial Neural Networks, vol. 5, pp. 221-226, USA, November 1995.
62
63
64
65

- 1
2
3
4 [18] MUVIS. [Online] <http://muvis.cs.tut.fi/>
5 [19] R. O. Duda, P. E., Hart, and, D. G. Stork, *Pattern Classification*, 2nd edition, John Wiley & Sons, New York, 2001
6 [20] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 3rd edition, Elsevier, USA, 2006.
7 [21] M. Bicego, M. D. R. Martinez and V. Murino, "A supervised data-driven approach for microarray spot quality
8 classification", *Pattern Analysis and Applications*, vol. 8, pp. 181-187, 2005.
9 [22] V. Vapnik, *Statistical learning theory*, Wiley, New York, 1998.
10 [23] K. Crammer and Y. Singer, "On the learnability and design of output codes for multiclass problems", *Machine Learning*,
11 vol. 47, no. 2, pp. 201-233, 2002.
12 [24] C.W. Hsu, and C.J. Lin, "A comparison of methods for multiclass support vector machines", *IEEE Trans. on Neural
13 Networks*, vol. 13, no. 2, pp. 415-425, 2002.
14 [25] C.C. Chang, C.J. Lin, "LIBSVM: a library for support vector machines," Technical Report, National Taiwan University,
15 Taiwan, 2004.
16 [26] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: A stepwise procedure for building and training a
17 neural network," *Neurocomputing: Algorithms, Architectures and Applications*, vol. F68 of NATO ASI Series, pp. 41-50.
18 Springer-Verlag, 1990.
19 [27] D. Rumelhart, G. Hinton, and R. Williams, "Learning Internal Representation by Error Propagation," *Parallel Distributed
20 Processing: Explorations in the Microstructure of Cognition*, MIT Press, vol. 1, pp. 318-362, USA, 1986.
21 [28] S.E. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Networks," Technical Report, CMU-CS-88-
22 162, Carnegie-Mellon University, Pittsburgh, USA, 1988.
23 [29] R. S. Sutton, "Two problems with backpropagation and other steepest-descent learning procedures for networks," in *Proc. of
24 8th Annual Conf. Cognitive Science Society*, pp. 823-831, May, 1986.
25 [30] R. Neruda and P. Kudova, "Learning methods for radial basis function networks," *Future Generation Computer Systems*,
26 vol. 21, no. 7, pp. 1131-1142, July 2005.
27 [31] T. Tollenaere, "SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties," *Neural Networks*, vol. 3, no.
28 5 pp. 561-573, 1990.
29 [32] Y. Chauvin and D. E. Rumelhart, "Back Propagation: Theory, Architectures, and Applications," Lawrence Erlbaum
30 Associates Publishers, UK, 1995.
31 [33] A. Korzynska, W. Stronjny, A. Hoppe, D. Wertheim and P. Hoser, "Segmentation of microscope images of living cells",
32 *Pattern Analysis and Applications*, vol. 10, pp. 301-319, 2007.
33 [34] R. M. Haralick, K. Shanmugam, and I. Dinstein (1973). "Textural Features for Image Classification". *IEEE Transactions on
34 Systems, Man, and Cybernetics SMC-3*, vol. 6, pp. 610-621, 1973.
35 [35] T. Ojala M. Pietikainen, T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local
36 binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.24, no.7, pp.971-987, Jul 2002.
37 [36] S.E. Grigorescu,; N. Petkov, P. Kruizinga, "Comparison of texture features based on Gabor filters," *IEEE Transactions on
38 Image Processing*, vol.11, no.10, pp. 1160-1167, Oct 2002.
39 [37] M. Partio, B. Cramariuc, and M. Gabbouj, "An Ordinal Co-occurrence Matrix Framework for Texture Retrieval," *EURASIP
40 Journal on Image and Video Processing*, vol. 2007, Article ID 17358, 15 pages, 2007.
41 [38] D.G., Lowe, "Object recognition from local scale-invariant features," *In Proceedings of the Seventh IEEE International
42 Conference on Computer Vision*, vol.2, pp.1150-1157, 1999.
43 [39] N. Dalal,; B. Triggs, , "Histograms of oriented gradients for human detection," *Computer Vision and Pattern Recognition,
44 2005. CVPR 2005. IEEE Computer Society Conference on*, vol.1, pp.886-893, 25 June 2005.
45 [40] ImageJ: public domain Java-based image processing program, [Online]. Available: <http://rsbweb.nih.gov/ij/docs/index.html>
46 [41] T. Masters, "Practical Neural Network Recipes in C++," Morgan Kaufmann, 1993.
47 [42] C. W. Hsu, C. C. Chang, and C. J. Lin, "A practical guide to support vector classification," Taipei, Tech. Rep., 2003.
48 [Online]. Available: <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>
49 [43] S. Kiranyaz, M. Gabbouj, "Hierarchical Cellular Tree: An Efficient Indexing Scheme for Content-based Retrieval on
50 Multimedia Databases", *IEEE Transactions on Multimedia*, vol. 9, Issue 1, pp. 102-119, January 2007.
51 [44] S. Kiranyaz, T. Ince, A. Yildirim and M. Gabbouj, "Evolutionary Artificial Neural Networks by Multi-Dimensional Particle
52 Swarm Optimization", *Neural Networks*, in Press, doi:10.1016/j.neunet.2009.05.013, 2009.
53
54
55
56
57
58
59
60
61
62
63
64
65