# An Ensemble Approach for Data Fusion with Learn++

Michael Lewitt, Robi Polikar

Electrical and Computer Engineering, Rowan University,
136 Rowan Hall, Glassboro, NJ 08028, USA.
mlewitt@ieee.org, polikar@rowan.edu

**Abstract.** We have recently introduced Learn++ as an incremental learning algorithm capable of learning additional data that may later become available. The strength of Learn++ lies with its ability to learn new data without forgetting previously acquired knowledge and without requiring access to any of the previously seen data, even when the new data introduce new classes. Learn++, inspired in part by AdaBoost, achieves incremental learning through generating an ensemble of classifiers for each new dataset that becomes available and then combining them through weighted majority voting with a distribution update rule modified for incremental learning of new classes. We have recently discovered that Learn++ also provides a practical and a general purpose approach for multisensor and/or multimodality data fusion. In this paper, we present Learn++ as an addition to the new breed of classifier fusion algorithms, along with preliminary results obtained on two real-world data fusion applications.

## I. Introduction

### A. Incremental Learning and Data Fusion

A common, and often painful, characteristic of classification algorithms is that they require the availability of an adequate and representative set of training examples for satisfactory generalization performance. Often, acquisition of such data is expensive and time consuming. Consequently, it is not uncommon for the entire data to become available in small batches over a period of time. Furthermore, the datasets acquired in later batches may introduce instances of new classes that were not present in previous datasets. In such settings, it is necessary to update an existing classifier in an incremental fashion to accommodate new data without compromising classification performance on old data. The ability of a classifier to learn under this setting is usually referred to as *incremental* (also called *cumulative* or *lifelong*) *learning*.

Incremental learning however, is conceptually related to data fusion, as new data may be obtained using a different set of sensors, or simply be composed of a different set of features. In such cases, the classifier is expected to learn and integrate the novel information content provided by new features, hence data fusion.

Ensemble or multiple classifier systems (MCS) have attracted a great deal of attention over the last decade due to their reported superiority over single classifier sys-

tems on a variety of applications. MCS combines an ensemble of generally weak classifiers to take advantage of the so-called *instability* of the weak classifier. This instability causes the classifiers to construct sufficiently different decision boundaries for minor modifications in their training datasets (or other parameters), causing each classifier to make different errors on any given instance. A strategic combination of these classifiers then eliminates the individual errors, generating a strong classifier.

A rich collection of algorithms have been developed using multiple classifiers with the general goal of improving the generalization performance of the classification system. Using multiple classifiers for incremental learning, however, has been largely unexplored. Learn++ was developed in response to recognizing the potential feasibility of ensemble of classifiers in solving the incremental learning problem.

In our previous work, we have shown that Learn++ is indeed capable of incrementally learning from new data, without forgetting previously acquired knowledge and without requiring access to previous data even when additional datasets introduce new classes [1]. The general approach in Learn++, much like those in other MCS algorithms, such as AdaBoost [2], is to create an ensemble of classifiers, where each classifier learns a subset of the dataset. The classifiers are then combined using weighted majority voting [3]. Learn++ differs from other techniques, however, in the way the data subsets are chosen to allow incremental learning of new data.

Recognizing that data fusion also involves combining different datasets consisting of new features or modalities, we have evaluated Learn++ on two real world applications requiring data fusion. Learn++ was used to generate additional ensembles of classifiers from datasets comprising of different features/sensors/modalities, which were then combined using weighted majority voting. While the algorithm certainly has much room for improvement when used in data fusion mode, the initial results utilizing the existing version of the algorithm have been very promising. In this paper, we describe the Learn++ algorithm and how it can be used as a general purpose approach for a variety of data fusion applications, along with our preliminary results on two such applications.

### B. Ensemble Approaches for Data Fusion

Several approaches have been developed for data fusion, for which ensemble approaches constitute a relatively new breed of algorithms. Traditional methods are generally based on probability theory, such as the Dempster-Schafer (DS) theory of evidence and its many variations. However, algorithms based on DS require specific knowledge of the underlying probability distribution, which may not be readily available. The majority of these algorithms have been developed in response to the needs of military applications, most notably target detection and tracking [4-6]. Ensemble approaches seek to provide a fresh and a more general solution for a broader spectrum of applications. Such approaches include simpler combination schemes such as majority vote, threshold voting, averaged Bayes classifier, maximum/minimum rules, and linear combinations of posterior probabilities [6-8]. More complex data fusion schemes are also widely used in practice including ensemble based variations of DS, neural network and fuzzy logic classifiers, and stacked generalization [9-14].

A related approach to data fusion and classifier combination schemes is input decimation, the use of feature subsets in multiple classifiers [15, 16]. In addition to the simpler combination methods of majority vote, maximum, minimum, average, and product, slightly more complex combination schemes such as behavior-knowledge space or decision templates can be employed [15, 16]. Input decimation can be useful in allowing different modalities, such as Fourier coefficients and pixel averages, to be naturally grouped together for independent classifiers [15]. Input decimation can also be used to lower the dimensionality of the input space by "weeding out input features that do not carry strong discriminating information" [16]. A useful addition to this list of classifier ensembles is a more general structure capable of using a variety of different basic network architectures and containing the ability to combining their outputs for (a) a stronger overall classifier, (b) a classifier capable of incremental learning, and (c) a classifier capable of easily fusing its outputs with other ensembles.

## II. Learn++

The power of Learn++ as an ensemble of classifiers approach lies in its ability to learn incrementally additional information from new data. Specifically, for each database that becomes available, Learn++ generates an ensemble of relatively weak classifiers, whose outputs are combined through weighted majority voting to obtain the final classification. The weak classifiers are trained based on a dynamically updated distribution over the training data instances, where the distribution is biased towards those novel instances that have not been properly learned or seen by the previous ensemble(s). The pseudocode for the Learn++ algorithm is provided in Figure 1.

For each database $\mathcal{D}_k$, $k=1,...,K$ that is submitted to Learn++, the inputs to the algorithm are (i) $S_k = \{(x_i, y_i) \mid i = 1, \cdots, m_k\}$ , a sequence of $m_k$ training data instances $x_i$ along with their correct labels $y_i$, (ii) a weak classification algorithm **BaseClassifier** to generate weak hypotheses, and (iii) an integer $T_k$ specifying the number of classifiers (hypotheses) to be generated for that database. The only requirement on the **BaseClassifier** algorithm is that it can obtain a 50% correct classification performance on its own training dataset. **BaseClassifier** can be any supervised classifier such as a multilayer perceptron, radial basis function, or a support vector machine, whose weakness can be achieved by reducing their size and increasing their error goal with respect to the complexity of the problem. Using weak classifiers allows generating sufficiently different decision boundaries based on slightly different training datasets. Weak classifiers also have the advantage of rapid training because, unlike stronger classifiers, they only generate a rough approximation of the decision boundary, further helping to prevent overfitting of the training dataset.

Learn++ starts by initializing a set of weights for the training data, $\boldsymbol{w}$, and a distribution $\boldsymbol{D}$ obtained from $\boldsymbol{w}$, according to which a training subset $TR_t$ and a test subset $TE_t$ are drawn at the $t^{th}$ iteration of the algorithm. Unless apriori information indicates otherwise, this distribution is initially set to be uniform, giving equal probability to each instance to be selected into the first training subset.

At each iteration $t$, the weights adjusted at iteration $t$-1 are normalized to ensuring a legitimate distribution, $\boldsymbol{D}_t$, is obtained (step 1). Training and test subsets are then

**Input**: For each dataset drawn from $\mathfrak{D}_k$ $k=1,2,...,K$

- Sequence of $m_k$ examples $S_k = \{(x_i, y_i) \mid i = 1, \cdots, m_k\}$
- Weak learning algorithm **BaseClassifier**
- Integer $T_k$, specifying the number of iterations

**Initialize** $w_1(i) = D_1(i)=1/m_k$, $\forall i$, $i=1,2,...,m_k$

**Do** for each $k=1,2,...,K$:

**Do** for $t= 1,2,...,T_k$:

1. Set $D_t = \mathbf{w_t} \Big/ \sum_{i=1}^{m} w_t(i)$ so that $\boldsymbol{D_t}$ is a distribution
2. Draw training $TR_t$ and testing $TE_t$ subsets from $\boldsymbol{D_t}$.
3. Call **BaseClassifier** to be trained with $TR_t$.
4. Obtain a hypothesis $h_t$: $X \rightarrow Y$, and calculate the error of $h_t$: $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$ on $TR_t+TE_t$. If εt> ½ , discard ht and go to step 2. Otherwise, compute normalized error as $\beta_t = \varepsilon_t / (1-\varepsilon_t)$.
5. Call weighed majority voting and obtain the composite hypothesis $H_t = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log(1/\beta_t)$
6. Compute the error of the composite hypothesis $E_t = \sum_{i:H_t(x_i)\neq y_i} D_t(i) = \sum_{i=1}^{m} D_t(i)\left[\left|H_t(x_i) \neq y_i\right|\right]$
7. Set $B_t = E_t/(1-E_t)$, and update the weights: $w_{t+1}(i) = w_t(i) \times \begin{cases} B_t, & if\ H_t(x_i) = y_i \\ 1, & otherwise \end{cases} = w_t(i) \times B_t^{1-\left[\left|H_t(x_i)\neq y_i\right|\right]}$

**Call** Weighted majority voting and **Output** the final hypothesis:

$$H_{final}(x) = \arg\max_{y\in Y} \sum_{k=1}^{K} \sum_{t:h_t(x)=y} \log(1/\beta_t)$$

**Fig. 1.** Learn++ Algorithm

drawn according to $\boldsymbol{D_t}$ (step 2), and the weak classifier is trained with the training subset (step 3). A hypothesis $h_t$ is obtained as the $t^{th}$ classifier, whose error $\varepsilon_t$ is computed on the entire (current) database $S_k=TR_t + TE_t$, simply by adding the distribution weights of the misclassified instances (step 4)

$$\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i) \tag{1}$$

The error, as defined in Equation (1), is required to be less than ½ to ensure that a minimum reasonable performance can be expected from $h_t$. If this is the case, the hypothesis $h_t$ is accepted and the error is normalized to obtain the normalized error

$$\beta_t = \varepsilon_t /(1-\varepsilon_t), \qquad 0 < \beta_t < 1 \tag{2}$$

If $\varepsilon_t \geq ½$, then the current hypothesis is discarded, and a new training subset is selected by returning to step 2. All hypotheses generated thus far are then combined using the weighted majority voting to obtain the composite hypothesis $H_t$ (step 5).

$$H_t = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log(1/\beta_t) \tag{3}$$

The voting scheme used by Learn++ is less than democratic, however, as the algorithm chooses the class receiving the highest vote from all hypotheses, where the voting weight for each hypothesis is inversely proportional to its normalized error. Therefore, those hypotheses with good performances are awarded a higher voting weight. The error of the composite hypothesis is then computed in a similar fashion as the sum of distribution weights of the instances that are misclassified by $H_t$ (step 6)

$$E_t = \sum_{i:H_t(x_i)\neq y_i} D_t(i) = \sum_{i=1}^{m} D_t(i)\left[\left|H_t(x_i) \neq y_i\right|\right] \tag{4}$$

where $[|\cdot|]$ evaluates to 1, if the predicate holds true.

$$B_t = E_t/(1-E_t), \quad 0 < B_t < 1 \tag{5}$$

The normalized composite error $B_t$ is computed for the weight update rule (step 7):

$$\begin{aligned} w_{t+1}(i) &= w_t(i) \times \begin{cases} B_t, & \text{if } H_t(x_i) = y_i \\ 1 & \text{, otherwise} \end{cases} \\ &= w_t(i) \times B_t^{1-\left[\left|H_t(x_i)\neq y_i\right|\right]} \end{aligned} \tag{6}$$

Equation (6) reduces the weights of those instances that are correctly classified by the composite hypothesis $H_t$, lowering their probability of being selected into the next training subset. In effect, the weights of misclassified instances are increased relative to the rest of the dataset. We emphasize that, unlike AdaBoost and its variations, the weight update rule in Learn++ looks at the classification of the composite hypothesis, not of a specific hypothesis. This weight update procedure forces the algorithm to focus more and more on instances that have not been properly learned by the ensemble. When Learn++ is learning incrementally, the instances introduced by the new database (and in particular from new classes, if applicable) are precisely those not learned by the ensemble, and hence the algorithm quickly focuses on these instances. At any point, a final hypothesis $H_{final}$ can be obtained by combining all hypotheses that have been generated thus far using the weighed majority voting rule

$$H_{final}(x) = \arg\max_{y \in Y} \sum_{k=1}^{K} \sum_{t:\, h_t(x)=y} \log\frac{1}{\beta_t}. \tag{7}$$

Simulation results of Learn++ on incremental learning using a variety of datasets as well as comparisons to the AdaBoost algorithm and other methods of incremental learning including ensemble of classifier approaches can be found in [1] and references within.

## III. Simulation Results on Learn++ for Data Fusion

Similar to other classifier fusion algorithms, Learn++ seeks to acquire novel and additional information content by selectively distributing the information to individual elements of the ensemble and then strategically combining them. While Learn++ was originally developed as an incremental learning algorithm, its ensemble structure allows it to be used in data fusion applications as well. This is because, the algorithm does not assume that instances in consecutive databases are composed of the same features as those seen previously. When used in data fusion mode, Learn++ seeks to incrementally learn additional information provided by consequent databases, where the instances of the new data still come from the same application but are composed of different features.

We have tested Learn++ on two real world classification problems which required data fusion. In the first, additional databases provided sensor measurements obtained with different sensors. In the second application, additional databases were constructed by taking different transforms of a time domain signal. These applications, how they relate to data fusion, and Learn++ simulation results are presented next.

### A. Gas Identification Database

The gas identification database used in this study consisted of responses of quartz crystal microbalances to twelve volatile organic compounds (VOC), including acetone, acetonitrile, toluene, xylene, hexane, octane, methanol, ethanol, methyethylketone, tricholoroethylene, tricholoroethane, and dicholoroethane. The task was identification of an unknown VOC from the sensor responses to that VOC. A total of 12 sensors were used, where each sensor was coated with a different polymer to alter its sensitivity and selectivity to different VOCs. The entire data was available in three databases $S_1$, $S_2$, $S_3$, where each database consisted of responses of 4 of the 12 sensors. More information on this database, including the experimental setup, names of the polymers, and sample instances are available on the web [17].

Sensor responses were acquired in response to seven different concentrations for each VOC, producing a total of 84 instances for each database. Thirty instances from each database were used for training and the remaining 54 were used for validation. We note that the validation data was not used by Learn++ during training. At any iteration $t$, the test subset, $TR_t$ mentioned above is a different subset of the 30 instance training data. We have randomly mixed the sensors for four separate trials of the algorithm, where the sensors were combined in mutually exclusive groups of four. For each run, $R_1$, $R_2$, $R_3$, $R_4$, three ensembles $E_1$, $E_2$, $E_3$, were generated by Learn++, one for each database. Table 1 summarizes sensor (feature) selection for each run. For the results shown in Table 2, each ensemble consisted of a multilayer perceptron network with a single 8-node hidden layer and an error goal of 0.005. The individual performances indicate the generalization performance of Learn++ when trained with only four features. Results for binary combination of ensembles indicate the generalization performance when two ensembles were combined by Learn++ to effectively fuse information from two four-feature datasets. Finally, the tertiary combination of ensembles

show how the algorithm performed when three datasets were combined to fuse information from three four-feature datasets.

**Table 1.** Ensemble features for each run were determined randomly to form three ensembles with four mutually exculsive sets of features.

| Dataset features | $E_1$ | $E_2$ | $E_3$ |
|---|---|---|---|
| $R_1$ | 12, 7, 6, 3 | 10, 4, 5, 9 | 8, 1, 2, 11 |
| $R_2$ | 2, 9, 4, 3 | 10, 11, 5, 1 | 6, 8, 7, 12 |
| $R_3$ | 3, 2, 10, 12 | 7, 5, 9, 8 | 1, 11, 4, 6 |
| $R_4$ | 3, 7, 12, 11 | 1, 8, 9, 10 | 4, 5, 6, 2 |

**Table 2.** Learn++ generalization performance on the validation dataset.

| | Individual Performance | | | Combined Performance | | | |
|---|---|---|---|---|---|---|---|
| | $E_1$ | $E_2$ | $E_3$ | $E_1$ $E_2$ | $E_1$ $E_3$ | $E_2$ $E_3$ | $E_1$ $E_2$ $E_3$ |
| $R_1$ | 69.0% | 79.8% | 66.7% | 82.1% | **83.3%** | 82.1% | 82.1% |
| $R_2$ | 79.8% | 79.8% | 75.0% | 82.1% | 91.7% | 83.3% | **95.2%** |
| $R_3$ | 78.6% | 77.4% | 76.2% | 76.2% | 84.5% | 81.0% | **84.5%** |
| $R_4$ | 81.0% | 82.1% | 77.4% | 86.9% | 84.5% | 90.5% | **92.9%** |

The results indicate a general improvement in the generalization performance when the ensembles are combined. The best performance is indicated in bold for each run. It is interesting to note that in $R_1$, the best performance was achieved when combining ensembles $E_1$ and $E_3$. In all other cases, combining all three ensembles produced a performance increase over performances of any individual ensemble or any binary combination of other ensembles. Furthermore, no combination ever performed worse than an individual ensemble, demonstrating the ability of Learn++ in fusing information from three different datasets and acquiring additional information from each consecutive database.

### B. Ultrasonic Weld Inspection (UWI) Database

The UWI database consists of ultrasonic scans of nuclear power plant pipes for the purpose of distinguishing between three types of pipe wall defects. The three defects of interest in this database are intergranular stress corrosion cracking (IGSCC), counterbores, and weld roots. IGSCCs are usually found in an area immediately neighboring the welding region, known as the heat affected zone, and form a discontinuity in the pipe that can be detected by using ultrasonic (or eddy current) techniques. The counterbore and weld roots also appear as discontinuities in the pipe wall, but they do not degrade the structural integrity of the pipe. These two *geometric reflectors* often generate signals that are very similar to those generated by cracks, making the defect identification a very challenging task. The cross section in Figure 2 helps illustrate the ultrasonic testing procedure, employing 1 MHz ultrasonic transducers, used to generate the first database analyzed in this study.

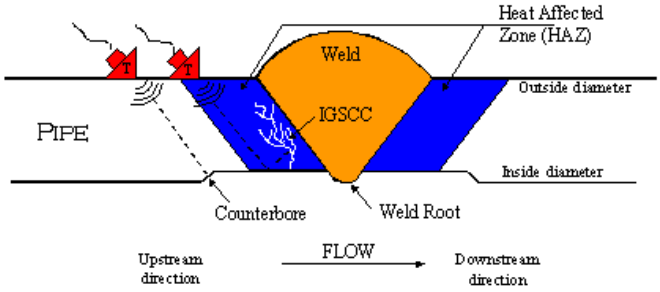


**Fig. 2.** Ultrasonic testing of nuclear power plant pipes for identification of hazardous *crack* vs non-hazardous *weld root* and *counterbore*.

The goal of the classification algorithm is to distinguish between the three different types of pipe wall defects from the ultrasonic scans. The database consists of 1845 instances, specifically 553 crack, 615 counterbore, and 677 weld root instances. A subset of these instances was given to the first ensemble of each run as the time domain information. The Fourier Transform of each instance was obtained and the same subset as before was given to the second ensemble of each run as frequency domain information. Finally, the discrete wavelet transform coefficients from a Daubechies-4 mother wavelet were computed to form the third ensemble of each run as a time-scale representation of the data. Training three ensembles of Learn++ on the time, frequency, and time-scale data provided three modalities of information to be fused for final classification. Table 3 shows the base classifier architecture and number of classifiers generated for five different runs, whereas Table 4 presents the results obtained by Learn++ on the UWI database when used in data fusion mode.

As before, combinations of ensembles performed better than individual ensembles. In general, the combination of all three ensembles (that is combining time, frequency and time-scale information) performed better than any individual or other binary combination, with the exceptions of $R_3$ and $R_5$, where the frequency and time-scale combination performed only slightly better then the combination of all. The results given in Table 4 provides further promising results indicating that the algorithm is able to combine information of different features from different datasets.

**Table 3.** UWI Database: Ensemble parameters for individual runs

| Run # | Hidden Layer nodes | Error goal | Training size | # of classifiers/ensemble | | |
|---|---|---|---|---|---|---|
| | | | | $E_1$ | $E_2$ | $E_3$ |
| $R_1$ | 8 | 0.1 | 500 | 15 | 20 | 23 |
| $R_2$ | 20 | 0.1 | 500 | 23 | 19 | 25 |
| $R_3$ | 30 | 0.1 | 500 | 8 | 8 | 9 |
| $R_4$ | 10 | 0.1 | 500 | 6 | 10 | 7 |
| $R_5$ | 20 | 0.1 | 500 | 10 | 8 | 9 |

**Table 4.** Learn++ Datafusion performance on the test dataset

| | Individual Performance | | | Combined Performance | | | |
|---|---|---|---|---|---|---|---|
| | $E_1$ | $E_2$ | $E_3$ | $E_1 E_2$ | $E_1 E_3$ | $E_2 E_3$ | $E_1 E_2 E_3$ |
| $R_1$ | 87.36% | 81.41% | 87.36% | 89.29% | 88.84% | 89.77% | **91.52%** |
| $R_2$ | 86.84% | 80.52% | 86.47% | 88.69% | 87.36% | 90.93% | **91.30%** |
| $R_3$ | 82.97% | 79.03% | 81.04% | 85.80% | 82.16% | **85.80%** | 85.72% |
| $R_4$ | 82.01% | 78.74% | 82.01% | 84.24% | 81.19% | 84.01% | **85.58%** |
| $R_5$ | 80.74% | 80.30% | 80.30% | 86.10% | 80.59% | **86.62%** | 85.13% |

## IV. Discussion & Conclusions

Learn++ has been evaluated as a potential data fusion algorithm capable of combining data from ensembles trained on separate uncorrelated features, as well as ensembles trained on different correlated modalities such as time, frequency, and wavelet domain data. The algorithm relies on the weighted voting scheme inherent to Learn++ to take advantage of the synergistic knowledge acquisition property of an ensemble of classifiers. In essence, the incremental learning algorithm is used in a data fusion setting, where the consecutive databases use instances of different features.

The two datasets presented here show improved performance when the ensemble's information are combined, forming a joint classification. However, the datasets also show that not all combinations are necessarily better than a single ensemble's performance. This information could be used to help select the features and modalities for future ensembles. A related advantage of Learn++ used in a data fusion setting is the ability of the algorithm to add or remove modalities or feature groups from the overall system without having to retrain the complete system. The algorithm has already been shown to be capable of incremental learning [1], so the combination of incremental learning of new data with the flexibility of adding or removing features and/or modalities makes for an extremely versatile algorithm.

Further testing on the algorithm is currently underway, along with the following future directions:

- Determine whether the algorithm can be used to obtain the optimum subset of a large number of features
- Since the algorithm is independent of the base classifiers, determine whether using different classifier structures particularly suited to a specific set of features may also be combined for improved data fusion performance (Learn++ has already demonstrated that it can work with a variety of base classifiers when running in the incremental learning mode [18]).

In summary, the unique characteristics of the Learn++ algorithm make it a potentially powerful and versatile system that can not only incrementally learn additional knowledge, but also can combine ensembles generated by training with different features for a diverse set of data fusion applications.

**Acknowledgement**: This material is based upon work supported by the National Science Foundation under Grant No. ECS-0239090.

## V. References

[1] R. Polikar, L. Udpa, S. Udpa, V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans Systems, Man and Cybernetics*, vol.31, no.4, pp.497-508, 2001.

[2] Y. Freund and R. Schapire, "A decision theoretic generalization of online learning and an application to boosting," *Computer and System Sciences*, vol. 57, no. 1, pp. 119-139, 1997.

[3] N. Littlestone and M. Warmuth, "Weighted majority algorithm," *Information and Computation*, vol. 108, pp. 212-261, 1994.

[4] D. Hall and J. Llinas, "An intrododuction to multisensor data fusion", *IEEE Proceedings*, vol. 85, no. 1, 1997.

[5] D. Hall and J. Llinas (editors), Handbook of Multisensor Data Fusion, CRC Press: Boca Raton, FL, 2001.

[6] L. A. Klein, Sensor and Data Fusion Concepts and Applications, SPIE Press, vol. TT35: Belingham, WA, 1999.

[7] J. Grim, J. Kittler, P. Pudil, and P. Somol, "Information analysis of multiple classifier fusion," *2nd Intl Workshop on Multiple Classifier Systems*, MCS 2001, pp. 168-177.

[8] J. Kittler, M. Hatef, R.P. Duin, J. Matas, "On combining classifiers," *IEEE Trans on Pattern Analysis and Machine Intelligence,* vol. 20, no.3, pp. 226-239, 1998.

[9] L.O. Jimenez, A.M. Morales, A. Creus, "Classification of hyperdimensional data based on feature and decision fusion approaches using projection pursuit, majority voting and neural networks, *IEEE Trans Geoscience and Remote Sensors*, vol. 37, no. 3, pp 1360-1366, 1999.

[10] G.J. Briem, J.A. Benediktsson, and J.R. Sveinsson, "Use of multiple classifiers in classification of data from multiple data sources," *Proc. of IEEE Geoscience and Remote Sensor Symposium*, vol. 2, pp. 882-884, Sydney, Australia, 2001.

[11] A. Krzyzak, C.Y. Suen, L. Xu. "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Trans Systems, Man, and Cybernetics*, vol.22, no.3, pp. 418-435, 1992.

[12] F.M. Alkoot.; J. Kittler. "Multiple expert system design by combined feature selection and probability level fusion," *Proc of the 3rd Intl Conf on FUSION 2000,* vol. 2, pp. 9-16, 2000.

[13] D. Wolpert, "Stacked Generalization," *Neural Networks*, vol. 2, pp 241-259, 1992.

[14] B.V. Dasarathy, "Adaptive fusion processor paradigms for fusion of information acquired at different levels of detail," Optical Engineering, vol 35, no 3 pp 634-649, 1996.

[15] L. Kuncheva and C. Whitaker, "Feature subsets for classifier combination: an enumerative experiment," *2nd Intl Workshop on Multiple Classifier Systems*, MCS 2001, pp. 228-237.

[16] N. Oza and K. Tumer, "Input decimation ensembles: decorrelation through dimensionality reduction," *2nd Intl Workshop on Multiple Classifier Systems*, MCS 2001, pp. 238-247.

[17] R. Polikar, VOC Identification database available at http://engineering.eng.rowan.edu/~polikar/RESEARCH/voc_database.doc

[18] R. Polikar, J. Byorick, S. Krause, A. Marino, M. Moreton, "Learn++: a classifier independent incremental learning algorithm for supervised Neural Networks," *Proc. of Intl Joint Conf on Neural Networks*, vol.2, pp. 1742-1747, 2002.