# An Ensemble-Based Incremental Learning Approach to Data Fusion

Devi Parikh and Robi Polikar, *Member, IEEE*

***Abstract*—This paper introduces Learn++, an ensemble of classifiers based algorithm originally developed for incremental learning, and now adapted for information/data fusion applications. Recognizing the conceptual similarity between incremental learning and data fusion, Learn++ follows an alternative approach to data fusion, i.e., sequentially generating an ensemble of classifiers that specifically seek the most discriminating information from each data set. It was observed that Learn++ based data fusion consistently outperforms a similarly configured ensemble classifier trained on any of the individual data sources across several applications. Furthermore, even if the classifiers trained on individual data sources are fine tuned for the given problem, Learn++ can still achieve a statistically significant improvement by combining them, if the additional data sets carry complementary information. The algorithm can also identify—albeit indirectly—those data sets that do not carry such additional information. Finally, it was shown that the algorithm can consecutively learn both the supplementary novel information coming from additional data of the same source, and the complementary information coming from new data sources without requiring access to any of the previously seen data.**

***Index Terms*—Data fusion, incremental learning, Learn++, multiple classifier/ensemble systems.**

## I. Introduction

### A. Incremental Learning and Data Fusion

CLASSIFICATION algorithms usually require an adequate and representative set of training data to generate an appropriate decision boundary among different classes. This requirement still holds even for ensemble (of classifiers)-based approaches that resample and reuse the training data. However, acquisition of such data for real-world applications is often expensive and time consuming. Hence, it is not uncommon for the entire data set to gradually become available in small batches over a period of time. In such settings, an existing classifier may need to learn the novel—or supplementary—information content in the new data without forgetting the previously acquired knowledge and without requiring access to previously seen data. The ability of a classifier to learn under these circumstances is commonly referred to as "incremental learning."

Manuscript received November 29, 2005; revised May 9, 2006 and July 24, 2006. This material is based upon work supported by the National Science Foundation under Grant ECS-0239090. This paper was recommended by Associate Editor N. Chawla.

D. Parikh was with the Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028 USA. She is now with the Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

R. Polikar is with the Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028 USA (e-mail: polikar@rowan.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2006.883873

On the other hand, in many applications that call for automated decision making, it is not unusual to receive data obtained from different sources that may provide complementary information. A suitable combination of such information is known as "data" or "information" fusion, and can lead to improved accuracy of the classification decision compared to a decision based on any of the individual data sources alone. Consequently, both incremental learning and data fusion involve learning from different sets of data. If the consecutive data sets that later become available are obtained from different sources and/or consist of different features, the incremental learning problem turns into a data fusion problem. Recognizing this conceptual similarity, we propose an approach based on an ensemble of classifiers—originally developed for incremental learning—as an alternative and surprisingly well-performing approach to data fusion.

### B. Ensemble Approaches and Data Fusion

An ensemble of classifiers based system combines several, and preferably diverse, classifiers. The diversity in the classifiers is typically achieved by using a different training data set for each classifier, which then allows each classifier to generate different decision boundaries. The expectation is that each classifier will make a different error, and strategically combining these classifiers can reduce the total error. Since its humble beginnings with such seminal works including, but not limited to [1]–[7], research in multiple classifier systems has expanded rapidly and has become an important research topic [8]. Over the last decade, ensemble systems appeared in the literature under many creative names, such as combination of multiple classifiers [9]–[12], dynamic classifier selection [12], classifier fusion [13]–[15], mixture of experts [4], [16], committees of neural networks [17], stacked generalization [5], or composite classifier systems [1], among others. These approaches differ from each other in terms of the procedure by which individual classifiers are generated, the procedure by which the classifiers are combined, or both.

There are generally two types of combination, namely, classifier selection and classifier fusion [8], [12], [13]. In classifier selection, each classifier is trained to become an expert in some local area of the entire feature space. The combination of classifiers is then based on the given data instance: the classifier trained with data closest to the vicinity of this instance is given the highest credit. One or more local experts can be nominated to make the decision [4], [12], [18], [19]. In classifier fusion—not to be confused with data fusion—all classifiers are trained over the entire feature space. The classifier combination process then involves merging the individual (weaker) classifiers to obtain a single (stronger) expert of superior performance, such as in bagging [20], or boosting-based approaches [3], [21].

The combination may operate on classification labels, rank ordering of labels, or continuous valued outputs of the individual classifiers [7], [9], [13]. In the latter case, classifier outputs are normalized to the [0, 1] interval, typically using the softmax rule [22], [23], which can then be interpreted as the support given by the classifier to each class, or even as class-conditional posterior probabilities [8], [13], [24].

Several combination rules are available, such as ranking, voting, sum, product, or some combination of posterior probabilities [6], [7], [15], [23], [25], fuzzy integral [26], [27], Dempster–Shafer (DS) based combination [10], [28], and, more recently, decision templates [13], [14]. Comparison and theoretical analyses of these rules are discussed in [7], [15], and [29]–[31]. A sample of the immense literature on ensemble systems can be found in [8] and elsewhere.[1]

We must mention that the word "fusion," which appears often in the aforementioned references, usually refers to the "combination" of classifiers for improving classifier performance using a single training data set and not necessarily to "data fusion" (combining information coming from different data sources). Commonly used methods for "data fusion" are generally based on Bayesian theory [32], [33], state estimation with Kalman or particle filtering [34]–[39], evidence theory (DS) [40], [41] and its variations [10], [42]–[45], information theoretic framework [46], neural networks [47], and evolutionary algorithms [48], [49]. The traditional application area for data fusion has long been target detection and tracking [38], [39], [50]–[54]. However, the aforementioned approaches, with their many variations and combinations, are becoming increasingly popular for emerging areas, such as remote sensing [36], [45], [55], [56], biometrics [57], [58], nondestructive evaluation (NDE) [59], [60], electroencephalographic data analysis [61], speech processing [62], [63], information retrieval [49], [64], and many others. Excellent review articles on various data fusion methods can be found in [50] and [65]–[67].

Using the ensemble approach for data fusion applications (i.e., combining complementary knowledge from different data sources), while addressed in some studies [9], [10], [23], [30], including our preliminary efforts [68], [69], has, in general, been less explored, particularly—if ever—in the context of incremental learning. Hence, our goal is to investigate the feasibility and properties of such an approach for data fusion applications, including such cases that may consecutively call for both incremental learning of supplementary information and data fusion of complementary information.

Specifically, we evaluate our proposed approach, i.e., the modified Learn++ algorithm, under three settings.

1) We first optimize the ensemble system by fine tuning its parameters so that its performance on each individual data source is maximized. We then show that even when the classifiers are individually optimized for each data source, their combination using Learn++ still provides a statistically significant performance improvement. This indicates that Learn++ is in fact able to learn complementary information from different sources.
2) We train the classifiers with nonoptimized (but still meaningful) parameters and combine these classifiers to illustrate that combining such nonoptimized classifiers performs reasonably close to individually optimized ensembles. Hence, Learn++ can be used to avoid the computationally expensive fine tuning step at a modest cost.
3) We evaluate the algorithm's ability to learn both incrementally and in a data fusion setting. The algorithm is first asked to learn complementary information provided by different data sources but then further learn the supplementary information provided by additional data from each data source.

The rest of this paper is organized as follows. In Section II, we describe the Learn++ algorithm in detail. In Section III, we discuss the experimental setup and the five databases used for evaluating the algorithm. In Section IV, we tabulate and interpret the results followed by conclusions and discussions in Section V.

## II. Learn++ for Data Fusion

Learn++ can incrementally learn novel information from new data—including from new classes—without forgetting the previously acquired knowledge and without requiring access to previous data, hence without suffering from catastrophic forgetting [70], [71]. Inspired in part by AdaBoost [21], Learn++ achieves incremental learning by generating an ensemble of classifiers, where each classifier is trained on a strategically updated distribution of the training data that focus on instances previously not seen or learned. Unlike AdaBoost, whose goal is to improve the performance of a classifier on a given data set, Learn++ specifically targets learning from additional data, i.e., Learn++ generates an ensemble for each data set that becomes available, and combines these ensembles to create an ensemble of ensembles or a meta-ensemble of classifiers. More importantly, the procedure through which consecutive classifiers are generated is different in Learn++, and is geared toward incrementally learning the novel and discriminating information provided by each data set that has not yet been learned by the "current ensemble." Therein lies the conceptual similarity between incremental learning and data fusion: the latter also requires learning from additional data, albeit generated from different sources or composed of heterogeneous sets of features. It is this incremental learning ability of the algorithm that we would like to explore within a data fusion setting. The overall approach is then to generate an ensemble of classifiers for each data set coming from a different source, and appropriately combine the classifier outputs to take advantage of the additional information in subsequent data sources.

In the context of data fusion, each source introduces data with a new feature set denoted as $\mathrm{FS}_k$, $k = 1, 2, \ldots, K$, where $K$ is the total number of data sources. For each data set

[1]See also various authors in the Proceedings of International Workshop on Multiple Classifiers Systems (2000–2005, 2007), Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany.

```
Input: For each feature set $FS_k$, $k=1,2,\dots,K$
  • Training data $S_k = [(\mathbf{x}_i, y_i)]$, $i=1,\dots,m_k$ with labels $y_i \in \Omega$
  • Supervised algorithm BaseClassifier.
  • Integer $T_k$, specifying the number of classifiers.
Do for each $k=1,2,\dots,K$:
    Initialize $w_1(i) = \boldsymbol{D_1}(i) = 1/m_k,\ \forall i,\ i=1,2,\cdots,m_k$
    Do for $t = 1,2,\dots,T_k$:
        1. Set $\boldsymbol{D_t} = \boldsymbol{w_t} \Big/ \sum_{i=1}^{m_k} w_t(i)$ so that $\boldsymbol{D_t}$ is a distribution.
        2. Draw training data subset $\boldsymbol{TR_t}$ from $\boldsymbol{S_k}$ according to $\boldsymbol{D_t}$.
        3. Obtain $h_t^k$ by training BaseClassifier with data $\boldsymbol{TR_t}$
        4. Calculate the error of $h_t^k$ on $S_k$: $\varepsilon_t^k = \sum_{i:h_t^k(\mathbf{x}_i)\neq y_i} \boldsymbol{D_t}(i)$
           If $\varepsilon_t^k > 1/2$, discard $h_t^k$ and → Go to Step 2.
        5. Set $\beta_t^k = \varepsilon_t^k / (1-\varepsilon_t^k)$. Obtain the composite hypothesis $H_t^k$
           $H_t^k = \arg\max_{y\in\Omega} \sum_{z:h_z^k(\mathbf{x})=y} \log\left(1/\beta_z^k\right),\quad z=\{1,\cdots,t\}$
        6. Compute the error of $H_t^k$ on $S_k$: $E_t^k = \sum_{i:H_t^k(\mathbf{x}_i)\neq y_i} \boldsymbol{D_t}(i)$
        7. Set $B_t^k = E_t^k/(1-E_t^k)$, and update the weights:
           $w_{t+1}(i) = w_t(i) \times \begin{cases} B_t^k, & \text{if } H_t^k(\mathbf{x}_i) = y_i \\ 1, & \text{otherwise} \end{cases}$
    Compute voting weights adjustment factor
           $\alpha_k = \left(\sum_{i=1}^{m_k} \left[\left|H_{T_k}(\mathbf{x}_i) \neq y_i\right|\right]\right) \Big/ m_k$
Output the final hypothesis:
           $H_{final}(\mathbf{x}) = \arg\max_{y\in\Omega} \sum_{k=1}^{K} \frac{1}{\alpha_k}\left(\sum_{t:h_t^k(\mathbf{x})=y} \log\left(1/\beta_t^k\right)\right)$
```

Fig. 1. Learn++ pseudocode for data fusion.

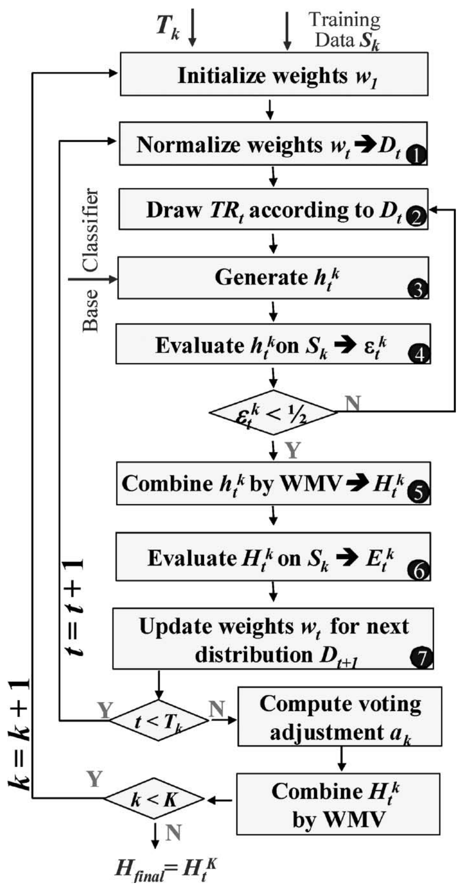
$T_k$
Training Data $S_k$
Initialize weights $w_1$
Normalize weights $w_t \rightarrow D_t$ 1
Draw $TR_t$ according to $D_t$ 2
Base Classifier
Generate $h_t^k$ 3
Evaluate $h_t^k$ on $S_k \rightarrow \varepsilon_t^k$ 4
$\varepsilon_t^k < 1/2$
N
Y
Combine $h_t^k$ by WMV $\rightarrow H_t^k$ 5
Evaluate $H_t^k$ on $S_k \rightarrow E_t^k$ 6
Update weights $w_t$ for next distribution $D_{t+1}$ 7
$t = t+1$
$t < T_k$
Y
N
Compute voting adjustment $a_k$
$k = k+1$
Combine $H_t^k$ by WMV
$k < K$
Y
N
$H_{final} = H_t^K$

Fig. 2. Learn++ block diagram.

FS$_k$ submitted to Learn++, the inputs to the algorithm are: 1) the training data $S_k$ with $m_k$ instances $\mathbf{x}_i$ along with their correct labels $y_i \in \Omega = \{\omega_1, \dots, \omega_C\}$, $i = 1, 2, \dots, m_k$, for $C$ number of classes; 2) a supervised classification algorithm "BaseClassifier" to generate individual classifiers (henceforth, hypotheses); and 3) an integer $T_k$, indicating the number of classifiers (NOCs) to be generated for the $k$th data set. The pseudocode of the algorithm and its block diagram are provided in Figs. 1 and 2, respectively, and described below in detail. The script $k$ is dropped whenever the meaning is unambiguous to avoid multiple scripts.

The BaseClassifier can be any supervised classifier whose "weakness" can be adjusted. This weakness ensures additional diversity, and can be controlled by adjusting training parameters [e.g., error goal (EG) of a neural network] with respect to the complexity of the problem. However, a meaningful minimum performance is enforced: the probability of a classifier to produce the correct labels on any given training data set must be at least 1/2. If classifier outputs are class-conditionally independent, then the overall error monotonically decreases as new classifiers are added. The proof of this argument, originally known as the Condorcet Jury Theorem (1786), can be found in [72] and [73]. This condition is necessary and sufficient for a two-class problem ($C = 2$), and is sufficient, but not necessary, for $C > 2$.

Learn++ generates each classifier of the ensemble sequentially using an iterative process. During the $t$th iteration, Learn++ trains the BaseClassifier on a strategically selected subset of the current training data to generate hypothesis $h_t^k$. The current training subset $\boldsymbol{TR}_t$ is drawn from the training data according to a distribution $\boldsymbol{D}_t$, which is obtained by normalizing a set of weights $\boldsymbol{w}_t$ maintained on the training data. The distribution $\boldsymbol{D}_t$ determines which instances of the training data are more likely to be selected into the training subset $\boldsymbol{TR}_t$. Unless *a priori* information requires otherwise, this distribution is initially set to be uniform, i.e.,

$$w_1(i) = 1/m_k, \qquad \forall i = 1, \dots, m_k \tag{1}$$

giving equal probability for each instance to be selected into $\boldsymbol{TR}_1$. At each subsequent iteration loop $t$, the weights previously adjusted at iteration $t - 1$ are normalized (in step 1 of the inner loop in Figs. 1 and 2)

$$\boldsymbol{D}_t = w_t \Big/ \sum_{i=1}^{m_k} w_t(i) \tag{2}$$

to ensure proper distribution. Training subset $\boldsymbol{TR}_t$ is drawn according to $\boldsymbol{D}_t$ (step 2), and the BaseClassifier is trained on $\boldsymbol{TR}_t$ (step 3). A hypothesis $h_t^k$ is generated by the $t$th classifier, whose error $\varepsilon_t^k$ is computed on the current data set $S_k$ as the sum of the distribution weights of the misclassified instances (step 4), i.e.,

$$\varepsilon_t^k = \sum_{i:h_t^k(\mathbf{x}_i)\neq y_i} \boldsymbol{D}_t(i) = \sum_{i=1}^{m_k} \boldsymbol{D}_t(i)\left[\left|h_t^k(\mathbf{x}_i)\neq y_i\right|\right] \tag{3}$$

where $[|\cdot|]$ evaluates to 1 if the predicate holds true, and 0 otherwise. As mentioned above, we insist that this error be less than 1/2. If that is the case, the hypothesis $h_t^k$ is accepted, and its error is normalized to obtain

$$\beta_t^k = \varepsilon_t^k / \left(1-\varepsilon_t^k\right), \qquad 0 < \beta_t^k < 1. \tag{4}$$

If $\varepsilon_t^k > 1/2$, then the current $h_t^k$ is discarded, and a new training subset is selected by returning to step 2. All hypotheses generated thus far are then combined using "weighted majority voting" to obtain the composite hypothesis $H_t^k$ (step 5), for which each hypothesis $h_t^k$ is assigned a weight inversely proportional to its normalized error. Therefore, those hypotheses with smaller training error are awarded a higher voting weight and thus have more say in the final classification decision. $H_t^k$ then represents the current ensemble decision

$$H_t^k(\mathbf{x}) = \arg\max_{y\in\Omega} \sum_{z:h_z^k(\mathbf{x})=y} \log\left(1/\beta_z^k\right), \qquad z=\{1,\ldots,t\}. \tag{5}$$

It can be shown that the weight selection of $\log(1/\beta_t^k)$ is optimum for weighted majority voting [8]. The error of the composite hypothesis $H_t^k$ is then computed in a similar fashion to that of $h_t^k$ (step 6) as

$$E_t^k = \sum_{i:H_t^k(\mathbf{x}_i)\neq y_i} \boldsymbol{D}_t(i) = \sum_{i=1}^{m_k} \boldsymbol{D}_t(i)\left[\left|H_t^k(\mathbf{x}_i)\neq y_i\right|\right]. \tag{6}$$

Since individual hypotheses that make up the composite hypothesis all have individual errors less than 1/2, so too will the composite error, i.e., $0 \le E_t^k < 1/2$. The normalized composite error $B_t^k$ can then be obtained as

$$B_t^k = E_t^k / \left(1-E_t^k\right), \qquad 0 < B_t^k < 1 \tag{7}$$

and is used for updating the distribution weights assigned to individual instances

$$\begin{aligned} w_{t+1}(i) &= w_t(i) \times B_t^{k\left(1-\left[\left|H_t^k(\mathbf{x}_i)\neq y_i\right|\right]\right)} \\ &= w_t(i) \times \begin{cases} B_t^k, & \text{if } H_t^k(\mathbf{x}_i)=y_i \\ 1, & \text{otherwise.} \end{cases} \end{aligned} \tag{8}$$

Equation (8) indicates that the distribution weights of those instances correctly classified by the composite hypothesis $H_t^k$ are reduced by a factor of $B_t^k$, making them less likely to be selected to the training subset of the next iteration. Readers familiar with AdaBoost have undoubtedly noticed not only the overall similarities but also the key difference between the two algorithms: the weight update rule of Learn++ specifically targets learning novel information from new data, whereas AdaBoost specifically targets improving the generalization performance of a weak learner on a single data set. This is because AdaBoost updates its weight distribution based on the decision of previously generated "hypothesis" $h_t$ [21], whereas Learn++ updates its distribution based on the decision of the current "ensemble" through the composite hypothesis $H_t^k$. This procedure forces Learn++ to focus on instances that have not been properly learned by the "ensemble." The final hypothesis $H_{\text{final}}$ is obtained by combining all hypotheses that have been generated thus far from all $K$ data sources.

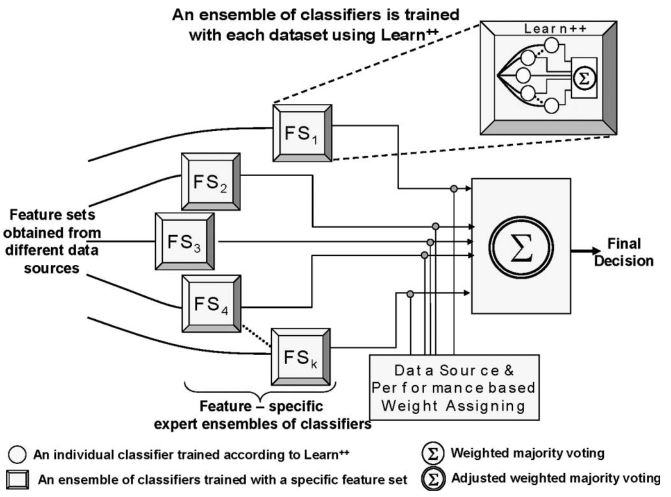
An ensemble of classifiers is trained with each dataset using Learn++
Learn++
FS1
FS2
FS3
FS4
FSk
Feature sets obtained from different data sources
Final Decision
Data Source & Performance based Weight Assigning
Feature – specific expert ensembles of classifiers
An individual classifier trained according to Learn++
An ensemble of classifiers trained with a specific feature set
Weighted majority voting
Adjusted weighted majority voting

Fig. 3. Schematic representation of the Learn++-based data fusion algorithm.

Fig. 3 conceptually illustrates the system-level organization of the overall algorithm as structured for data fusion applications: an ensemble of classifiers is generated as described above for each of the feature sets, which are then combined through weighted majority voting. For data fusion applications, however, performance-based voting weights for each classifier $\log(1/\beta_t^k)$ are further adjusted before final voting based on expected or observed training performance on each data source: if prior information is available about reliability of data obtained from a particular feature set, a higher voting weight can be assigned to classifiers trained with such data. Alternatively, the adjustment can be based on the training performance of the ensemble on its own feature set. If such a strategy is chosen, the performance-based weight of each classifier $\log(1/\beta_t^k)$ is multiplied by the "reliability factor" of the feature set to which it belongs. The adjusted weight is then used to obtain the final hypothesis

$$H_{\text{final}}(\mathbf{x}) = \arg\max_{y\in\Omega} \sum_{k=1}^{K} \frac{1}{\alpha_k} \left( \sum_{t:h_t^k(\mathbf{x})=y} \log\left(\frac{1}{\beta_t^k}\right)\right) \tag{9}$$

where $1/\alpha_k$ is the reliability factor assigned to the ensemble trained on the $k$th feature set. In this paper, $\alpha_k$ was chosen as the empirical error, that is, the misclassification ratio of the final composite hypothesis on $S_k$

$$\alpha_k = \left(\sum_{i=1}^{m_k} \left[\left|H_T^k(\mathbf{x}_i)\neq y_i\right|\right]\right) \Big/ m_k \tag{10}$$

where $H_T^k$ indicates the final composite hypothesis generated from the $k$th training data $S_k$ of feature set $\text{FS}_k$.

To summarize, Learn++ employs two sets of weights and a reliability factor when used for data fusion.

- The distribution weights $w(i)$ assigned to each instance $\mathbf{x}_i$ of the training data, and used to determine which instances are more likely to be drawn into the training subset of the next classifier.
- The voting weights $\log(1/\beta_t^k)$ assigned to "each classifier" based on its training performance. The higher is the training performance of $h_t^k$, the higher is its voting weight for final classification.
- The data source (feature set) based reliability factor $1/\alpha_k$ assigned to "all hypotheses in the $k$th ensemble" to adjust their training votes. $\alpha_k$ is based on prior information (if available and reliable), or on the performance of the $k$th ensemble on its own training data $S_k$.

A couple of implementation issues should also be mentioned to prevent rare but pathological conditions. First, classifiers with infinite voting weight should be avoided: when a classifier perfectly learns the entire training data (potentially overfitting), $\varepsilon_t^k$ will be 0, causing $\beta_t^k = 0$, and $\log(1/\beta_t^k) = \infty$. Then, $h_t^k$ is given the sole power of decision making. This situation can be avoided either by making classifiers weaker (so that the training error exceeds zero) or by adding a small adjustment factor—0.01 usually works well—to $\beta_t^k$.

Second, unless there is prior information to choose otherwise, the NOC $T_k$ generated for each data set should be the same. $T_k$ is usually determined by adding classifiers to the ensemble until the addition of classifiers no longer improves performance on a validation data set. However, the performance may stay mostly constant for a large NOC. A large NOC will then be retained despite the lack of meaningful performance gain. Apart from increased computational burden and potential for overfitting, an unnecessarily large NOC generated with any of the feature sets also causes a spurious imbalance toward the data source with more classifiers. This situation can be avoided by applying regularization to validation, or heuristically picking the NOC to be the same for each data source.

## III. Experimental Setup

### A. *Objectives of the Experimental Setup*

The experimental setup was designed to empirically demonstrate and statistically validate the following.

1) Learn++ is a viable data fusion algorithm, that is, its performance obtained by combining different feature sets will be higher—with a statistical significance—than that obtained by a comparable ensemble classifier trained on any of the feature sets alone.
2) Even if classifier ensembles are optimized to provide the best possible performance on an individual feature set, their combination using Learn++ will still show a performance improvement, over the best performing feature set, provided that the individual classifier decisions are class conditionally independent and each feature set provides at least some complementary information.
3) If classifiers trained on individual feature sets are not optimized, their combination using Learn++ will provide an even larger margin of improvement in fusion performance, possibly comparable to the performance obtained by fusion of "optimized ensemble of classifiers" mentioned above. The approach can therefore provide a meaningful tradeoff for avoiding the expensive fine-tuning steps.
4) The proposed decision-level fusion approach can be favorable to feature-concatenation-based feature-level fusion, even if the latter is used along with another ensemble approach, such as AdaBoost.
5) The incremental learning ability of Learn++ is preserved in the data fusion setting as well: if additional data later become available from new or existing sources (with or without new features), Learn++ can learn that information from the new data without requiring access to previously seen data.

### B. *Experimental Procedure*

Five databases from diverse application areas, whose details are provided in Section III-C, were used. For each of the databases, the following experimental procedure was followed.

1) Three databases were randomly partitioned into subsets, where each partition used only a portion of the features to simulate a data fusion setting. The remaining databases were naturally in this format, allowing a true real-world data fusion setting. Independence of features is assumed.
2) Using multilayer perceptrons (MLPs) as base classifiers, extensive statistical tests were conducted to determine optimum and moderate design parameters for error goal (EG), number of hidden layer nodes (HLNs), and the number of classifiers (NOC) for each feature set. Optimum parameters are those that provide the best statistically significant ensemble validation performance, resulting in stronger base classifiers. Moderate parameters are those that yield average performance for each feature set, resulting in weaker classifiers. Consider an ensemble whose base classifiers are trained with all possible combinations of $\mathrm{EG} = 0.01$ to $0.10$ in steps of $0.01$, $\mathrm{HLN} = 5$ to $25$ in steps of $5$, and $\mathrm{NOC} = 1$ to $10$. If performances among all combinations were in the 60%–90% range, those that result in 90% and 75% (average of 60%–90%) were chosen as optimum and moderate parameters, respectively. Moderate parameters represent typical values a user may choose, based on past experience, if determining optimal parameters were deemed too costly.
3) Discarding all classifiers generated thus far, a new set of ensembles of classifiers was trained using the optimized parameters on each feature. These ensembles were then combined for data fusion using Learn++. The procedure was repeated by randomly drawing new training and test data sets (50–200 times), keeping a constant class distribution. Mean and confidence intervals (CIs) of the generalization performances on individual feature sets and on data fusion were compared. The entire procedure was then repeated using moderate parameters. The

TABLE I
PROPERTIES OF DATABASES

| | Feature sets | Datasets | Parameters |
|---|---|---|---|
| **Optical Character Recognition (OCR)**[1] **Database** 10 classes: handwritten numerals '0' – '9' | **Features (649)**: 6 sets (Pixel averages, Profile correlations, Fourier coef., KL coefficients. etc.) $FS_1(216)$, $FS_2(76)$ $FS_3(64)$, $FS_4(240)$ $FS_5(47)$, $FS_6(6)$ | Total # of instances: 2000 200 per class Training set size: 300 Test set size: 1700 | **EG:** 6 error goals between 0.01 and 0.06 **HLN:** 10:25:5 **NOC:** 1:35:1 **Repetitions:** 2 Total # of experiments: 1,680 x 6FS = 10,080 |
| **Wine Recognition Database** 3 classes: Identification of 3 types of wine from 13 quantitative features | **Features (13)**: Quantities of 13 constituents of wine (Alcohol, Ash, Hue, etc) Divided into 3 sets $FS_1(4)$: 5, 7, 10, 4 $FS_2(4)$: 8, 1, 11, 12 $FS_3(5)$: 3, 9, 13, 2, 6 | Total #of instances: 178 Class 1: 59 Class 2: 71 Class 3: 48 Training set size: 30 Test set size: 148 | **EG:** 11 error goals between 0.005 and 0.35 **HLN:** 15:30:5 **NOC:** 1:15:1 **Repetitions:** 3 Total # of experiments: 1,980 x 3FS = 5,940 |
| **Water Database**[2] 4 classes: Operational state of urban waste water treatment plant | **Features (38):** Daily measures of sensors (water flow, zinc content, pH levels, etc.) Divided into 3 sets $FS_1(12)$, $FS_2(12)$, $FS_3(14)$ | # Total instances: 374 Class 1: 191 Class 2: 88 Class 3: 51 Class 4: 44 Training set size: 80 Test set size: 294 | **EG:** 11 error goals between 0.01 and 0.21. **HLN:** 15:30:5 **NOC:** 1:15:1 **Repetitions:** 3 Total # of experiments: 1,980 x 3FS = 5,940 |
| **VOC Database** Identification of 12 volatile organic compounds (Acetone, Hexane, Octane, Toluene, Xylene, Methanol, Ethanol, etc.) | **Features (12):** Responses of twelve microbalance gas sensors to 12 VOCs Divided into 3 sets $FS_1(4)$: 1,9,2,8 $FS_2(4)$: 10,12,7,4 $FS_3(4)$: 11,5,3,6 | # Total instances: 84 (7 per class) Training set size: 48 Test set size: 36 | **EG:** 11 error goals between 0.0005 and 0.05 **HLN:** 15:45:10 **NOC:** 1:15:1 **Repetitions:** 3 Total # of experiments: 1,980 x 3FS = 5,940 |
| **Nondestructive Evaluation (NDE) Database** 5 classes: four types of defects (pitting, crack, mechanical damage, weld) in pipelines + no defect | **Features (87):** DCT coefficients of images from $FS_1(15)$: Ultrasonic scan $FS_2(72)$: Magnetic flux leakage scan | # Total instances: 22 (about 4 per class) Training set size: 10 Test set size: 11 | **EG:** 13 error goals between 0.01 and 0.12 **HLN:** 5:45:5 **NOC:** 10:60:10 **Repetitions:** 2 Total # of experiments: 910 x 2FS = 1,820 |

[1]This database appears as Multiple Features Database in [74]. The name is changed here to better reflect the application. [2]The original database consisted of 527 instances. Instances with missing features, and classes with fewer than 5 instances were removed.

effect of the order of addition of feature sets was also tested.

4) Learn++ was then compared to a single strong classifier and to an AdaBoost-based ensemble of classifiers, for both of which the feature-level data fusion was obtained through feature concatenation.
5) Learn++ performance was also tested in a combined incremental learning and data fusion setting, that is, learning from "additional" data of "new" feature sets without access to previously seen data.

## C. Databases

Table I provides detailed information on the experimental setup for the three databases obtained from the UCI repository [74] and the two real-world applications. The first column provides the database name and the number of classes. The total number of features available, the physical nature of the feature sets, and the number of features in each set are given in the second column. For instance, the optical character recognition (OCR) database has a total of 649 features, which were already grouped into six heterogeneous feature sets, such as pixel averages, Fourier coefficients, etc., and feature set 1 ($FS_1$) has 216 features. The volatile organic compound (VOC) database, however, had 12 features, representing outputs of different polymer-coated chemical sensors. Sensors were randomly partitioned into three feature sets; $FS_1$, for example, receiving sensors 1, 9, 2, and 8. The total number of instances in the database, the class distribution, and the size of training and test data are provided in the third column. The last column indicates the training parameters that were evaluated. For instance, the OCR database was evaluated on all possible combinations of six EGs (0.01–0.06 in steps of 0.01), 10–25 HLNs in steps of 5, and 1–35 classifiers in steps of 1: a total of 840 possible combinations of parameters. The experiments were repeated using a different partitioning of training and test data, again separated randomly but maintaining the same class distribution. Thus, a total of 1680 ensembles were generated for each of the

TABLE II
OPTIMUM AND MODERATE TRAINING PARAMETERS FOR EACH DATABASE AND FEATURE SET

| | Error Goal | | Hidden Layer Nodes | | Number of Classifiers | |
|---|---|---|---|---|---|---|
| | **Optimum** | **Moderate** | **Optimum** | **Moderate** | **Optimum** | **Moderate** |
| **OCR Database** | 0.01 | 0.05 | $FS_1$: 20<br>$FS_2$, $FS_4$: 10<br>$FS_3$: 15<br>$FS_5$, $FS_6$: 25 | 15 | 10 | 3 |
| **Wine Database** | $FS_1$: 0.005<br>$FS_2$, $FS_3$: 0.01 | 0.175 | $FS_1$: 30<br>$FS_2$, $FS_3$: 25 | $FS_1$, $FS_3$: 30<br>$FS_2$: 25 | 5 | 4 |
| **Water Database** | $FS_1$: 0.05<br>$FS_2$: 0.03<br>$FS_3$: 0.07 | $FS_1$: 0.15<br>$FS_2$: 0.13<br>$FS_3$: 0.11 | 30 | 30 | 6 | 11 |
| **VOC Database** | 0.0005 | 0.03 | $FS_1$, $FS_2$: 15<br>$FS_3$: 45 | $FS_1$, $FS_2$: 25<br>$FS_3$: 35 | 5 | 5 |
| **NDE Database** | 0.05 | 0.08 | 30 | 30 | 30 | 20 |

six feature sets of OCR data, resulting in 10 080 simulations on which statistical analysis was performed to pick the optimum and moderate parameters for the OCR data.

The values of the chosen parameters are provided in Table II for each data set and feature set. For parameters not spelled out separately for each feature set, including the NOC, the given common value was used for all feature sets in a given database. The EG had the most, and the number of HLNs (in the tested range) had the least influence on the performance of the system.

Classifier ensembles using the parameters in Table II were generated and combined using Learn++. In order to obtain CIs, all fusion experiments were repeated 50–200 times (depending on the size of the database) using a different random partitioning of training and test subsets while observing the data distributions in column 3 of Table I. Learn++ performances for fusing different feature sets are provided first followed by results on combined incremental learning and data fusion experiments.

## IV. RESULTS

### A. Results on OCR Database

Fig. 4 summarizes the simulation results on the OCR database using optimum parameters. The values next to the bars are the percent generalization performances. The first six are performances obtained by individual feature sets: Feature set 2 performed the worst, followed by $FS_4$, $FS_5$, $FS_6$, and $FS_3$, with $FS_1$ providing the best performance. The remaining bars are the performances of combined feature sets, fused by Learn++, where additional feature sets are added in ascending (left) and descending (right) order of their individual performances. FS245, for example, denotes fusion of features sets $FS_2$, $FS_4$, and $FS_5$ in this order. The CIs are obtained from 50 repetitions of resampling training and test data sets.

The following observations can be made from Fig. 4. The generalization performance obtained by every combination of the feature sets is better than any of the individual feature sets fused, and the improvement is statistically significant at 95%, as indicated by nonoverlapping CIs. While fusing the feature sets in ascending order, we note that the performance levels off after fusing the first four sets and does not increase significantly

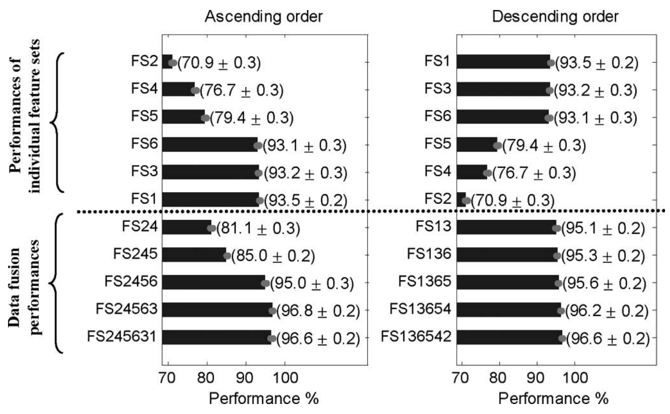

Fig. 4. Data fusion results on the OCR database using optimum parameters.

with the addition of the last two (highest performing) sets. That is, despite their higher performance, feature sets 3 and 1 do not provide any new information beyond what is already provided by feature sets 2, 4, 5, and 6. Hence, the need to find the best feature set(s) may be eliminated simply by combining available feature sets. On the other hand, while fusing the feature sets in descending order, most of the increase is realized when the first two (best) feature sets are combined (as expected), with subsequent sets adding slight but consistent improvements. Furthermore, the final performance combining all six sets remains the same irrespective of the order in which they are combined. This is expected as Learn++ uses a linear combination scheme, and thus is independent of the order in which the classifiers are combined.

Similar analysis was performed using a moderate set of parameters, the results of which are shown in Fig. 5. When fused in ascending order of performance, there is now a steady and a more significant increase in performance with each added feature set. Performance does not seem to level off as fast as it did with the optimum parameters. This observation also makes sense: with weaker classifiers using nonoptimized parameters, there is more novel information to be acquired from each feature set.

Finally, one additional observation: the average individual performance on six data sets is 84% using optimum parameters and 66% using moderate parameters, a difference of 18%.

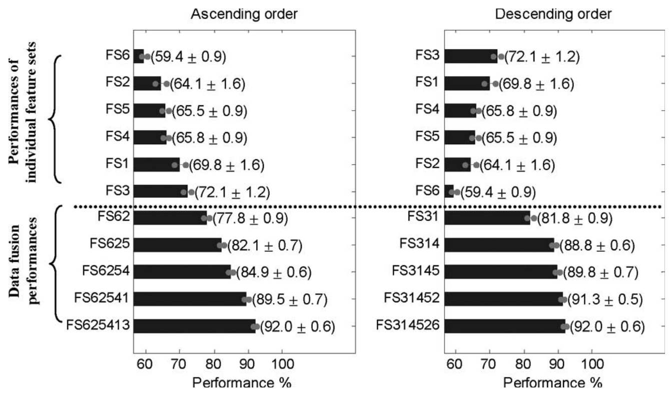

Fig. 5. Data fusion results on the OCR database using moderate parameters.

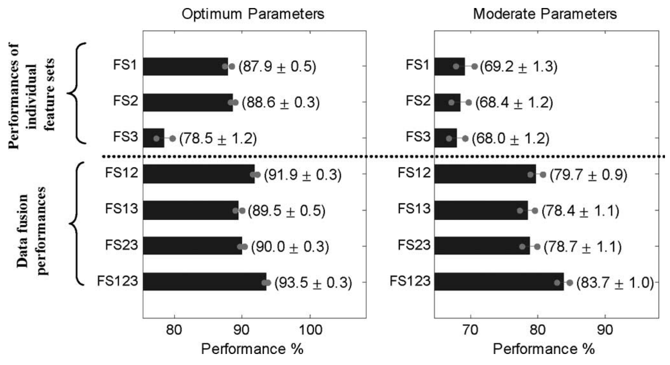

Fig. 6. Data fusion results on the wine recognition database.

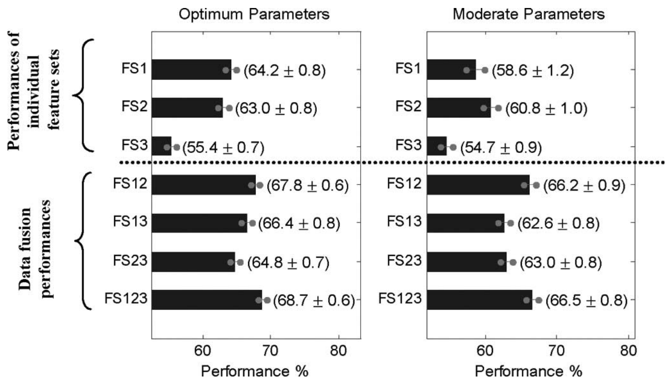

Fig. 7. Data fusion results on the water treatment plant database.

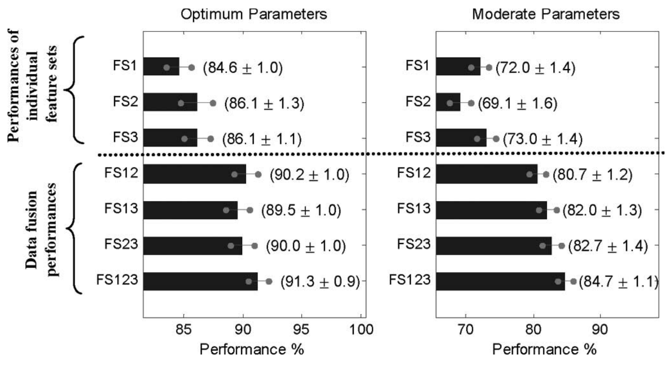

Fig. 8. Data fusion results on the VOC database.

After data fusion, this difference is only 4.6% (96.6% versus 92.0%). Hence, the performance difference between optimized and nonoptimized classifiers is significantly reduced by the fusion since the fusion of nonoptimized classifiers performed reasonably close to that of optimized classifiers. Then, at a relatively modest cost (4.6% in this case), the expensive step (10 080 evaluations) of fine tuning can be avoided by fusing nonoptimized classifiers.

## *B. Results on Wine Recognition Database*

Fig. 6 summarizes the simulation results on the Wine database using optimum (left) and moderate parameters (right). With fewer number of feature sets (three), all possible combinations of feature sets were fused ($FS_1$ and $FS_2$, $FS_1$ and $FS_3$, $FS_2$ and $FS_3$, and $FS_1$ and $FS_2$ and $FS_3$). The CIs are determined through 200 repetitions of random resampling of training and test data sets. Once again, we observe that each of the performances involving fusion of any two feature sets is better than the performance on the individual feature sets. Also, the fusion of all three sets (FS123) is better yet than any of the individual three, or any two feature set combination, all with statistical significance. Similar trends of increase in average performance, bigger margin of improvement with moderate parameters, etc., are also observed with this database.

## *C. Results on Water Treatment Plant Database*

Fig. 7 summarizes the simulation results on the somewhat more challenging water treatment plant database using optimum (left) and moderate parameters (right). CIs are determined through 200 random resampling of training and test data sets, as before. Previously mentioned trends can also be seen in the performances on this database, such as data fusion outperforming any of the ensembles trained on individual data sets, larger improvements in fusing classifiers with nonoptimized parameters, etc. We note that while FS123 provides significantly better performance than FS13 or FS23, the addition of $FS_3$ to FS12 provides only a small increase in performance, not statistically significant (CI overlap), neither with optimized nor with moderate parameters. This may indicate that $FS_3$ is not as discriminating and adds very little, if any, new information in addition to that provided by $FS_1$ and $FS_2$ combined. Such knowledge can be used to identify features that contain no discriminatory information.

## *D. Results on VOC Database*

The VOC database is obtained from a real-world problem, where the task is to recognize one of 12 VOCs from the responses of 12 gas sensors. Fig. 8 summarizes the simulation results using optimum parameters (left) and moderate parameters (right). CIs are determined through 100 repetitions of random resampling of training and test data sets. Similar general trends observed earlier are also observed here. However, while the fusion performance of any two feature sets was always higher (with significance) than that of the individual feature sets, the performance improvement on the fusion of all three sets was not statistically significant compared to a combination of two feature sets. This finding again indicates that the third

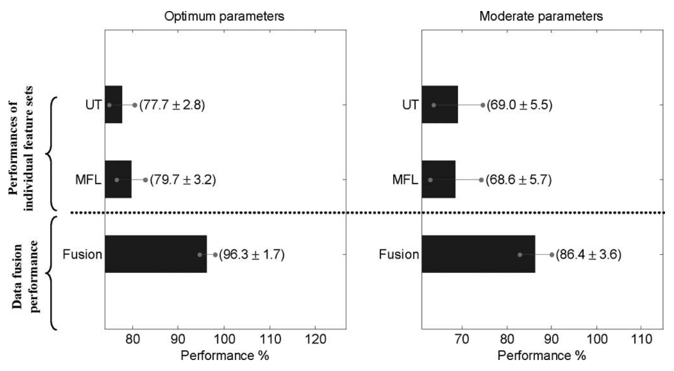

Fig. 9. Data fusion results on the NDE database.

set offers no additional information once the system is trained on the first two sets.

## *E. Results on NDE of Pipelines Database*

Fig. 9 illustrates the results on a real-world NDE database that naturally fits into a data fusion setting. The data sources are two different imaging modalities that use different physical phenomena for locating and identifying various types of defects in materials: ultrasonic scans that use mechanical waves, and magnetic flux leakage that use electromagnetic waves [75], [76]. Hence, similar to the OCR database, no artificial splitting of feature sets was required. Since there are only two feature sets, combining the feature sets in different orders was also not relevant. As with other databases, data fusion was performed using Learn++ by training the base classifiers with optimum and moderate parameters, as indicated in Table II. The results demonstrate the true effectiveness of the algorithm, perhaps more so than it did with databases where the feature sets were randomly split to simulate a data fusion environment. This is probably because the two sets of features do in fact carry considerable complementary information due to different physical nature of the data sources. In either case of using optimum or moderate parameters, Learn++ provided a substantial (and statistically very significant) improvement by fusing the information from two data sets.

## *F. Summary and Analysis of Data Fusion Performances*

Table III summarizes all results, comparing the mean performance and CI widths of the following:

- the average of all feature sets, indicating the expected average ensemble performance that can be obtained with a single data set, had we randomly picked one;
- the ensemble with the best individual feature set, indicating the expected average performance of the best feature set, if we had the luxury of evaluating each of them to find the best feature set;
- Learn++ based fusion, indicating the expected performance of simply fusing all feature sets with Learn++ without individually evaluating them.

We observe that the data fusion performance is consistently better than the performance of classifiers trained with even the best of the individual feature sets, and as expected, the performance improvement in ensembles trained with moderate parameters is more prominent. The difference between the mean data fusion performance and the performance of best ensembles trained on individual feature sets is also provided. We note that the CI widths are always lower for data fusion, a natural benefit of combining a large number of classifiers.

Finally, we also compare the performances obtained through Learn++ based data fusion with those obtained by simple concatenation of features. Concatenation of features is a commonly used pragmatic approach to data fusion, particularly when the individual feature sets are independent of each other. Such a comparison is provided for all databases in Table IV, except for the NDE database, where the concatenation of features does not make any physical sense. The comparison is specifically made among: 1) a single MLP trained on the concatenation of all feature sets; 2) the performance of an ensemble trained on the concatenation of all features (AdaBoost-based ensemble using similar training parameters, such as NOC, HLN, etc., for fair comparison); and 3) the data fusion performance obtained through Learn++. We observe from the relatively poor performance of a single MLP that such a concatenation scheme, while useful for certain applications (such as VOC[2]), is not necessarily an optimal approach. In fact, it may cause deterioration of performance if additional feature sets are irrelevant, or if the feature sets are completely heterogeneous, such as the OCR (or NDE) database. The addition of irrelevant features would have less of an impact on Learn++ due to performances of larger number of classifiers being averaged.

Regular (AdaBoost) ensembles trained with concatenated features do outperform the single MLP, empirical evidence of previously cited works that combining classifiers can be effectively used for data fusion. However, regular ensemble performances fall short those of Learn++, leading us to believe that improved performances are due not only to using an ensemble but also to the incremental learning strategy of Learn++.

## *G. Data Fusion From Additional Data in an Incremental Learning Setting*

We also look at the ability of the algorithm to handle both data fusion and incremental learning consecutively, a distinct property of Learn++ that distinguishes it from other data fusion algorithms. To test this property, the following setup was used. The algorithm is first asked to learn "complementary information" that become available from different sources in a data fusion setting as described above. The algorithm is then asked to further learn any "supplementary information" that may be present in additional data obtained from each of the data sources seen earlier. At no point in time was the algorithm allowed to revisit any of the data sets that it had seen before once training on that data set was complete. Once again, we test the algorithm separately on optimum and moderate parameters, as discussed above. For brevity, we present the results on two databases,

[2]We note that feature concatenation on a VOC data set outperforms Learn++ based classifier fusion. This could be due to the random splitting of the 12 features to simulate the data fusion setting for Learn++: the random split might have separated features that provide a better discriminative power when used together. Such a scenario should not occur in real-world applications where no random and artificial splitting of feature sets take place.

TABLE III
SUMMARY OF RESULTS ON ALL FOUR DATABASES

| Database | Optimum Parameter Performances | | | Moderate Parameter Performances | | |
|---|---|---|---|---|---|---|
| | | Mean % | CI Width% | | Mean % | CI Width% |
| OCR Database | Ave. feature set | 84.5 | 0.3 | Ave. feature set | 66.1 | 1.2 |
| | Best feature set | 93.5 | 0.2 | Best feature set | 72.1 | 1.2 |
| | Fusion | 96.6 | 0.2 | Fusion | 92.0 | 0.6 |
| | **Difference** | **3.10** | **0.0** | **Difference** | **19.0** | **- 0.6** |
| Wine Database | Ave. feature set | 85.0 | 0.7 | Ave. feature set | 68.5 | 1.2 |
| | Best feature set | 88.6 | 0.5 | Best feature set | 69.2 | 1.3 |
| | Fusion | 93.5 | 0.3 | Fusion | 83.7 | 1.0 |
| | **Difference** | **4.90** | **- 0.2** | **Difference** | **14.5** | **- 0.3** |
| Water Database | Ave. feature set | 60.9 | 0.8 | Ave. feature set | 58.0 | 1.0 |
| | Best feature set | 64.2 | 0.8 | Best feature set | 60.8 | 1.0 |
| | Fusion | 68.7 | 0.6 | Fusion | 66.5 | 0.8 |
| | **Difference** | **4.50** | **- 0.20** | **Difference** | **5.7** | **- 0.2** |
| VOC Database | Ave. feature set | 85.6 | 1.1 | Ave. feature set | 71.4 | 1.5 |
| | Best feature set | 86.1 | 1.1 | Best feature set | 73.0 | 1.4 |
| | Fusion | 91.3 | 0.9 | Fusion | 84.7 | 1.1 |
| | Difference | **5.20** | **- 0.20** | **Difference** | **11.7** | **- 0.30** |
| NDE Database | Ave. feature set | 78.7 | 3.0 | Ave. feature set | 68.8 | 5.6 |
| | Best feature set | 79.7 | 3.2 | Best feature set | 69.0 | 5.5 |
| | Fusion | 96.3 | 1.7 | Fusion | 86.4 | 3.6 |
| | Difference | **17.1** | **- 1.5** | **Difference** | **17.6** | **-1.9** |

TABLE IV
LEARN++ BASED DATA FUSION VERSUS FEATURE CONCATENATION-BASED DATA FUSION

| | | Optimum | | Moderate | |
|---|---|---|---|---|---|
| Database | Fusion Strategy | Mean% | CI Width% | Mean % | CI Width% |
| OCR Database | Strong MLP with feature concatenation | 91.2 | 0.6 | 62.0 | 0.9 |
| | Ensemble with feature concatenation | 95.8 | 0.2 | 72.4 | 1.5 |
| | Learn++ Fusion | 96.6 | 0.2 | 92.0 | 0.6 |
| VOC Database | Strong MLP with feature concatenation | 95.3 | 1.0 | 70.8 | 1.2 |
| | Ensemble with feature concatenation | 97.6 | 0.6 | 83.8 | 1.3 |
| | Learn++ Fusion | 91.3 | 0.9 | 84.7 | 1.1 |
| Water Database | Strong MLP with feature concatenation | 63.8 | 0.8 | 45.3 | 1.3 |
| | Ensemble with feature concatenation | 56.7 | 1.3 | 60.5 | 0.9 |
| | Learn++ Fusion | 68.7 | 0.4 | 66.5 | 0.8 |
| Wine Database | Strong MLP with feature concatenation | 53.1 | 3.6 | 44.8 | 2.1 |
| | Ensemble with feature concatenation | 89.8 | 0.8 | 72.8 | 1.3 |
| | Learn++ Fusion | 93.5 | 0.3 | 83.7 | 1.0 |

namely, the OCR and the VOC databases. The results for all other databases exhibited similar trends to those presented for OCR and VOC.

Fig. 10 summarizes the results on the OCR data using optimum (left) and moderate (right) parameters. All results are averages of 100 independent trials. The 300 training instances (see Table I for details), 30 per class, were partitioned into two subsets to be presented to the algorithm in separate sessions. That is, the algorithm was first presented with the first 150 training instances, 15 per class, constituting data set 1 ($\mathrm{DS}_1$). With optimum parameters, ten classifiers were generated per feature set, five with each data subset $\mathrm{DS}_1$ and $\mathrm{DS}_2$. Three classifiers were generated with the moderate parameters, two with the first data set and one with the second, per feature set. The performance on each feature set using the first half of the training data is shown as the first six bars in Fig. 10. The fusion of these feature sets is indicated as FS123456_DS1. Similar to previous results, the fusion performance is better than any of the individual sets. Note that the data fusion performance using half of the data (96.2%) is just a bit under that of the

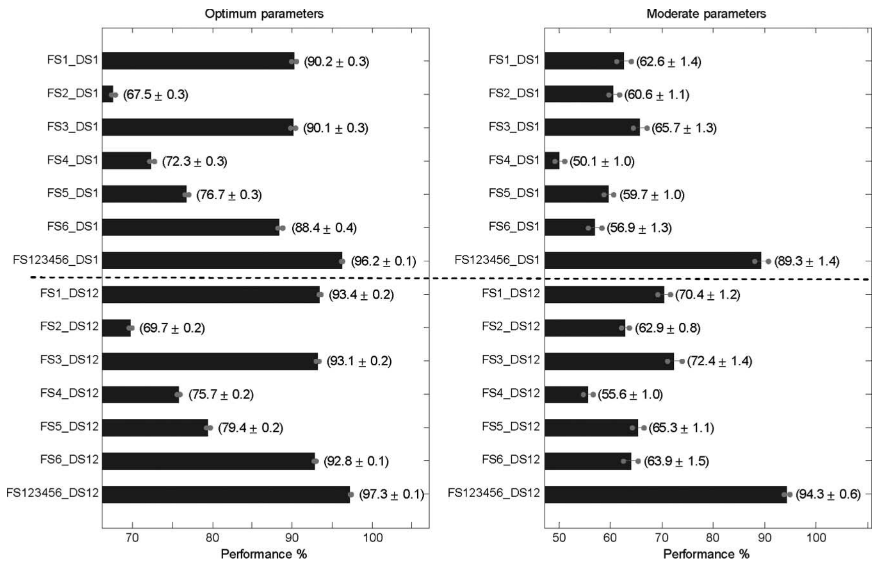

Fig. 10. Incremental learning and data fusion performance on OCR data.

earlier trial that used all training data with optimum parameters (96.6%). The large number of repetitions do indicate that this difference is statistically significant—albeit barely. A plausible interpretation for this observation is that virtually all of the discriminatory information that is available in the entire training data is in fact also available in the subset data that use only half the number of instances. As expected for moderate parameters, the difference is larger, i.e., 89.3% when trained on half of the data versus 92% with the entire training data.

Learn++ was then asked to generate additional ensembles using the remaining half of the training data, again one feature set at a time. FS$i$_DS12 indicates the performance of the algorithm on feature set $i$, $i = 1, \ldots, 6$, when the ensembles created during training sessions 1 and 2 are combined. We note in each case that the performance on that feature set increases after training with the additional data. Hence, the algorithm does extract additional novel information from new data for each feature set. FS123456_DS12 then indicates combining all classifiers for all feature sets (data fusion) and in both training sessions (incremental learning). This performance is 97.3%, which is better than all feature sets performing alone even after the incremental learning from the second half of the training data. With moderate parameters, the final data fusion/incremental learning performance is 94.3%, a much larger gain compared to the best feature set, as expected.

Furthermore, in both cases (optimum and moderate parameters), the performance is actually higher after the incremental learning of two halves of the data than the performance when all the training data were shown to the algorithm at once (97.3% versus 96.6 with optimum, and 94.3% versus 92% with moderate parameters). While these are statistically significant gains, the improvement of incremental learning over seeing all the data at once (per feature set) is application and data dependent and, hence, is not a general characteristic of the algorithm. Furthermore, while it may seem natural to compare incremental and nonincremental learning results (they are both presented in this paper), this comparison is in fact an unfair one.

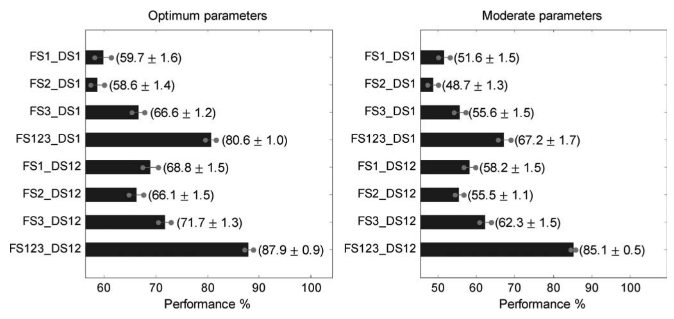

Fig. 11. Incremental learning and data fusion performance on VOC data.

This is because, in practice, incremental learning is only used as a result of a necessity presented by the availability of additional data. We merely point to: 1) the ability of the algorithm to learn from additional data of the same source in an incremental learning setting; 2) its ability to learn from additional data sources in a data fusion setting; and finally 3) its ability to combine both, if and when such data become available. Again, we emphasize that no previously used data set is later made available to the algorithm once the algorithm is trained with that data set. This illustrates the ability of the algorithm to incrementally learn "novel information" from new data and "complementary information" from additional data sources.

Finally, Fig. 11 illustrates the data fusion and incremental learning performance of the algorithm on the VOC database. A similar setup is used for the VOC database as the OCR database, that is, half the data for each feature set were shown to the algorithm during the first training session, whose data fusion performance is indicated as FS123_DS1. We observe, for each feature set, that the incremental learning performance after training with the second half of the training data is significantly higher than the performance after training with only the first half of the data. Furthermore, the data fusion plus incremental learning performance is higher than the performance of any other individual data source before or after the second training

TABLE V
LEARN++ UNDER UNBALANCED CONDITIONS OF NUMBER OF INSTANCES VERSUS NUMBER OF FEATURES

| | | Optimum | | Moderate | |
|---|---|---|---|---|---|
| **Scenario** | **Classification strategy** | **Mean%** | **CI Width%** | **Mean %** | **CI Width%** |
| **More Instances** | $FS_{3\_1}$ | 90.5 | 0.3 | 78.1 | 0.8 |
| | $FS_{3\_2}$ | 78.1 | 0.5 | 70.2 | 0.6 |
| | $FS_{3\_3}$ | 83.2 | 0.5 | 74.5 | 0.7 |
| | Ensemble with feature concatenation | 81.7 | 0.4 | 72.0 | 0.6 |
| | **Learn++ Fusion** | **94.7** | **0.3** | **91.2** | **0.4** |
| **More Features** | $FS_1$ | 89.5 | 0.4 | 67.6 | 1.4 |
| | $FS_2$ | 70.8 | 0.5 | 49.0 | 1.1 |
| | $FS_3$ | 89.1 | 0.4 | 61.6 | 1.2 |
| | $FS_4$ | 88.4 | 0.4 | 57.3 | 1.3 |
| | $FS_5$ | 73.7 | 0.5 | 57.1 | 1.2 |
| | $FS_6$ | 68.5 | 0.4 | 63.1 | 1.1 |
| | Ensemble with feature concatenation | 68.3 | 1.3 | 62.2 | 1.4 |
| | **Learn++ Fusion** | **96.0** | **0.2** | **90.9** | **0.9** |

session. Similar results can be observed for both optimum and moderate parameters.

### H. Number of Instances Versus Number of Features

Whether the behavior of the algorithm changes as the number of instances grows with respect to available number of features, or vice versa, may also be of interest for certain applications. To get a sense of the algorithm's behavior under such cases, the following two experiments were designed using the OCR database (as this data set has more features and instances than others).

In the first scenario, we chose $FS_3$, one of the best performing sets in the OCR data, and randomly split its 64 features into three feature subsets containing 20, 20, and 24 features each. 150 instances per class were used for training. Compared to the experiments reported earlier on the same database, the number of features (as compared to the number of instances) is now significantly smaller. Data fusion was performed using Learn++ and was compared to an AdaBoost-based ensemble approach on the concatenated features. In the second scenario, all six feature sets were combined; however, only ten instances were used for training from each class, while 190 were used for testing. With a total of 649 features (see Table I), the number of features is now significantly larger with respect to the number of training data points (only 10). Again, Learn++ based data fusion was performed, and the results were compared to an AdaBoost-based ensemble strategy on the concatenated features. Results for both scenarios are summarized in Table V.

In spite of a significant imbalance between the number of examples and features, Learn++ provides efficient data fusion. In both cases, data fusion performance is higher than any of the individual feature sets with statistical significance, and Learn++ performs better than the standard ensemble approach on concatenated features. We should note that the emphasis in these experiments is not the specific performance numbers (as favorable as they may be) but rather the algorithm's behavioral trends under each scenario.

## V. DISCUSSIONS AND CONCLUSION

Recognizing the conceptual similarities between incremental learning and data fusion, the Learn++ algorithm—originally developed for incremental learning—has been adapted for and evaluated in a data fusion setting. The algorithm incrementally and sequentially learns from data that consist of heterogeneous features by generating an ensemble of classifiers for each data set and then combining them through modified weighted majority voting. Individual classifier outputs and the feature sets are assumed to be class conditionally independent, a common assumption in most ensemble and fusion algorithms. These assumptions may not always be strictly met. In fact, it is likely that the feature sets assumed independent in our experiments were not strictly so. Yet, the algorithm seems to be tolerant to possible mild violations. Of course, time series data would severely violate this assumption and hence should not be used with this algorithm.

Simulation results indicate that Learn++ outperforms, with statistical significance, similarly configured ensemble classifiers trained with data from any of the individual sources. Several other desirable properties of the algorithm are also observed: 1) Learn++ can identify (albeit indirectly) feature sets that are redundant and carry no additional information; 2) the feature sets can be combined in any order without affecting the final performance; and 3) Learn++ based data fusion often performs significantly better than a classification system, whether an ensemble or a single classifier, that uses the pragmatic feature concatenation approach to data fusion. Furthermore, the incremental learning ability of the algorithm does carry over to the data fusion setting, that is, the algorithm can learn both the supplementary novel information coming from additional data of the same source, and the complementary information coming from new data sources without requiring repeated access to any of the previously seen data.

Learn++ can be beneficial regardless whether the ensembles trained on data from individual sources are optimized. Combining classifiers that are trained with optimized parameters allows further fine tuning and a performance gain above and beyond that of the best individual feature set. Combining classifiers trained with nonoptimized (moderate) parameters, however, helps avoid the expensive optimization step, and still shows a very strong data fusion performance—often close to that obtained by fusing optimized classifiers.

In this paper, we have used the MLP due to its wide popularity. However, any supervised classifier can be used as the base classifier. In fact, if there is reason to believe that a certain classifier will perform better on a particular feature set, different classifiers can also be fused.

In conclusion, the ability of the algorithm to learn incrementally, as well as to combine complementary information coming from different data sources, makes Learn++ a versatile technique. We believe that it is a useful addition to other successful ensemble-based algorithms due to its general structure and emphasis on sequential generation of individual classifiers. Evaluation of the algorithm on more challenging problems (such as web mining) and theoretical analysis of the algorithm constitute future work.

## Acknowledgment

The authors would like to thank the anonymous reviewers' most thorough review of this paper. Their valuable comments have greatly benefited this paper.

## References

[1] B. V. Dasarathy and B. V. Sheela, "Composite classifier system design: Concepts and methodology," *Proc. IEEE*, vol. 67, no. 5, pp. 708–713, May 1979.
[2] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990.
[3] R. E. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, no. 2, pp. 197–227, 1990.
[4] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, 1991.
[5] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.
[6] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 1, pp. 66–75, Jan. 1994.
[7] J. Kittler, M. Hatef, R. P. W. Duin, and J. Mates, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, Mar. 1998.
[8] L. I. Kuncheva, *Combining Pattern Classifiers, Methods and Algorithms*. New York: Wiley, 2005.
[9] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 3, pp. 418–435, May/Jun. 1992.
[10] G. Rogova, "Combining the results of several neural network classifiers," *Neural Netw.*, vol. 7, no. 5, pp. 777–781, 1994.
[11] L. Lam and C. Y. Suen, "Optimal combinations of pattern classifiers," *Pattern Recognit. Lett.*, vol. 16, no. 9, pp. 945–954, 1995.
[12] K. Woods, W. P. J. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 4, pp. 405–410, Apr. 1997.
[13] L. I. Kuncheva, J. C. Bezdek, and R. P. W. Duin, "Decision templates for multiple classifier fusion: An experimental comparison," *Pattern Recognit.*, vol. 34, no. 2, pp. 299–314, 2001.
[14] L. I. Kuncheva, "Switching between selection and fusion in combining classifiers: An experiment," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 2, pp. 146–156, Apr. 2002.
[15] ——, "A theoretical study on six classifier fusion strategies," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 281–286, Feb. 2002.
[16] M. J. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Comput.*, vol. 6, no. 2, pp. 181–214, 1994.
[17] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik, "Boosting and other ensemble methods," *Neural Comput.*, vol. 6, no. 6 pp. 1289–1301, 1994.
[18] E. Alpaydin and M. I. Jordan, "Local linear perceptrons for classification," *IEEE Trans. Neural Netw.*, vol. 7, no. 3, pp. 788–792, May 1996.
[19] G. Giacinto and F. Roli, "Approach to the automatic design of multiple classifier systems," *Pattern Recognit. Lett.*, vol. 22, no. 1, pp. 25–33, 2001.
[20] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[21] Y. Freund and R. E. Schapire, "Decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.
[22] R. O. Duda, P. E. Hart, and D. Stork, "Algorithm independent techniques," in *Pattern Classification*, 2nd ed. New York: Wiley, 2001, pp. 453–516.
[23] D. M. J. Tax, M. van Breukelen, R. P. W. Duin, and J. Kittler, "Combining multiple classifiers by averaging or by multiplying?" *Pattern Recognit.*, vol. 33, no. 9, pp. 1475–1485, 2000.
[24] M. Muhlbaier, A. Topalis, and R. Polikar, "Ensemble confidence estimates posterior probability," *Proc. 6th Int. Workshop Multiple Classifier Syst., Lecture Notes in Computer Science*, N. Oza *et al.*, Eds., Monterey, CA, 2005, vol. 3541, pp. 326–335.
[25] J. Grim, J. Kittler, P. Pudil, and P. Somol, "Multiple classifier fusion in probabilistic neural networks," *Pattern Anal. Appl.*, vol. 5, no. 2, pp. 221–233, Jun. 2002.
[26] S. B. Cho and J. H. Kim, "Multiple network fusion using fuzzy logic," *IEEE Trans. Neural Netw.*, vol. 6, no. 2, pp. 497–501, Mar. 1995.
[27] M. Grabisch and J. M. Nicolas, "Classification by fuzzy integral: Performance and tests," *Fuzzy Sets Syst.*, vol. 65, no. 2/3, pp. 255–271, 1994.
[28] Y. Lu, "Knowledge integration in a multiple classifier system," *Appl. Intell.*, vol. 6, no. 2, pp. 75–86, 1996.
[29] K. Tumer and J. Ghosh, "Analysis of decision boundaries in linearly combined neural classifiers," *Pattern Recognit.*, vol. 29, no. 2, pp. 341–348, 1996.
[30] N. S. V. Rao, "On fusers that perform better than best sensor," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 8, pp. 904–909, Aug. 2001.
[31] G. Fumera and F. Roli, "A theoretical and experimental analysis of linear combiners for multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 942–956, Jun. 2005.
[32] C. Biao and P. K. Varshney, "A Bayesian sampling approach to decision fusion using hierarchical models," *IEEE Trans. Signal Process.*, vol. 50, no. 8, pp. 1809–1818, Aug. 2002.
[33] Y. Zhang and Q. Ji, "Active and dynamic information fusion for multisensor systems with dynamic Bayesian networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 2, pp. 467–472, Apr. 2006.
[34] R. Lobbia and M. Kent, "Data fusion of decentralized local tracker outputs," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 30, no. 3, pp. 787–799, Jul. 1994.
[35] K. C. Chou, A. S. Willsky, and A. Benveniste, "Multiscale recursive estimation, data fusion, and regularization," *IEEE Trans. Autom. Control*, vol. 39, no. 3, pp. 464–478, Mar. 1994.
[36] Q. Honghui and J. B. Moore, "Direct Kalman filtering approach for GPS/INS integration," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 38, no. 2, pp. 687–693, Apr. 2002.
[37] Z. Lei, W. Xiaolin, P. Quan, and Z. Hongcai, "Multiresolution modeling and estimation of multisensor data," *IEEE Trans. Signal Process.*, vol. 52, no. 11, pp. 3170–3182, Nov. 2004.
[38] S. R. Maskell, R. G. Everitt, R. Wright, and M. Briers, "Multi-target out-of-sequence data association: Tracking using graphical models," *Inf. Fusion*, vol. 7, no. 4, pp. 434–447, Dec. 2006.
[39] P. Perez, J. Vermaak, and A. Blake, , "Data fusion for visual tracking with particles," *Proc. IEEE*, vol. 92, no. 3, pp. 495–513, Mar. 2004.
[40] A. P. Dempster, "Upper and lower probabilities induced by multivalued mappings," *Ann. Math. Stat.*, vol. 38, no. 2, pp. 325–339, 1967.
[41] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press, 1976.
[42] D. Fixsen and R. P. S. Mahler, "The modified Dempster–Shafer approach to classification," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 27, no. 1, pp. 96–104, Jan. 1997.

[43] S. Le Hegarat-Mascle, I. Bloch, and D. Vidal-Madjar, "Application of Dempster–Shafer evidence theory to unsupervised classification in multisource remote sensing," *IEEE Trans. Geosci. Remote Sens.*, vol. 35, no. 4, pp. 1018–1031, Jul. 1997.
[44] R. R. Murphy, "Dempster–Shafer theory for sensor fusion in autonomous mobile robots," *IEEE Trans. Robot. Autom.*, vol. 14, no. 2, pp. 197–206, Apr. 1998.
[45] F. Rottensteiner, J. Trinder, S. Clode, and K. Kubik, "Using the Dempster–Shafer method for the fusion of LIDAR data and multi-spectral images for building detection," *Inf. Fusion*, vol. 6, no. 4, pp. 283–300, Dec. 2005.
[46] M. B. Hurley, "An extension of statistical decision theory with information theoretic cost functions to decision fusion: Part II," *Inf. Fusion*, vol. 6, no. 2, pp. 165–174, Jun. 2005.
[47] G. A. Carpenter, S. Martens, and O. J. Ogas, "Self-organizing information fusion and hierarchical knowledge discovery: A new framework using ARTMAP neural networks," *Neural Netw.*, vol. 18, no. 3, pp. 287–295, Apr. 2005.
[48] I. V. Maslov and I. Gertner, "Multi-sensor fusion: An evolutionary algorithm approach," *Inf. Fusion*, vol. 7, no. 3, pp. 304–330, 2006.
[49] W. Fan, M. Gordon, and P. Pathak, "On linear mixture of expert approaches to information retrieval," *Decis. Support Syst.*, vol. 42, no. 2, pp. 975–987, Nov. 2006.
[50] *Handbook of Multisensor Data Fusion*, D. L. Hall, Ed. Boca Raton, FL: CRC, 2001.
[51] T. Kirubarajan and Y. Bar-Shalom, "Probabilistic data association techniques for target tracking in clutter," *Proc. IEEE*, vol. 92, no. 3, pp. 536–557, Mar. 2004.
[52] Y. Bar-Shalom and C. Huimin, "IMM estimator with out-of-sequence measurements," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 41, no. 1, pp. 90–98, Jan. 2005.
[53] B. Ristic, "Target identification using belief functions and implication rules," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 41, no. 3, pp. 1097–1103, Jul. 2005.
[54] E. Bosse, P. Valin, A. C. Boury-Brisset, and D. Grenier, "Exploitation of a priori knowledge for information fusion," *Inf. Fusion*, vol. 7, no. 2, pp. 161–175, 2006.
[55] M. Weis, S. Muller, C. E. Liedtke, and M. Pahl, "A framework for GIS and imagery data fusion in support of cartographic updating," *Inf. Fusion*, vol. 6, no. 4, pp. 311–317, Dec. 2005.
[56] P. Gamba, F. Dell'Acqua, and B. V. Dasarathy, "Urban remote sensing using multiple data sets: Past, present, and future," *Inf. Fusion*, vol. 6, no. 4, pp. 319–326, Dec. 2005.
[57] M. Faundez-Zanuy, "Data fusion in biometrics," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 20, no. 1, pp. 34–38, Jan. 2005.
[58] A. Ross and A. Jain, "Information fusion in biometrics," *Pattern Recognit. Lett.*, vol. 24, no. 13, pp. 2115–2125, 2003.
[59] D. Horn and W. R. Mayo, "NDE reliability gains from combining eddy-current and ultrasonic testing," *NDT E Int.*, vol. 33, no. 6, pp. 351–362, Sep. 2000.
[60] V. Kaftandjian, M. Z. Yue, O. Dupuis, and D. Babot, "The combined use of the evidence theory and fuzzy logic for improving multimodal nondestructive testing systems," *IEEE Trans. Instrum. Meas.*, vol. 54, no. 5, pp. 1968–1977, Oct. 2005.
[61] L. Gupta, C. Beomsu, M. D. Srinath, D. L. Molfese, and K. Hyunseok, "Multichannel fusion models for the parametric classification of differential brain activity," *IEEE Trans. Biomed. Eng.*, vol. 52, no. 11, pp. 1869–1881, Nov. 2005.
[62] R. P. Ramachandran, K. Farrell, R. Ramachandran, and R. Mammone, "Speaker recognition-general classifier approaches and data fusion methods," *Pattern Recognit.*, vol. 35, no. 12, pp. 2801–2821, 2002.
[63] H. Altincay and M. Demirekler, "Speaker identification by combining multiple classifiers using Dempster–Shafer theory of evidence," *Speech Commun.*, vol. 41, no. 4, pp. 531–547, 2003.
[64] S. Wu and S. McClean, "Performance prediction of data fusion for information retrieval," *Inf. Process. Manage.*, vol. 42, no. 4, pp. 899–915, Jul. 2006.
[65] M. Kam, Z. Xiaoxun, and P. Kalata, "Sensor fusion for mobile robot navigation," *Proc. IEEE*, vol. 85, no. 1, pp. 108–119, Jan. 1997.
[66] J. Z. Sasiadek, "Sensor fusion," *Annu. Rev. Control*, vol. 26, no. 2, pp. 203–228, 2002.
[67] R. P. S. Mahler, "'Statistics 101' for multisensor, multitarget data fusion," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 19, no. 1, pp. 53–64, Jan. 2004.
[68] M. Lewitt and R. Polikar, "An ensemble approach for data fusion with Learn++," in *Proc. 4th Int. Workshop Multiple Classifier Syst.*, Surrey, U.K., 2003, *Lecture Notes in Computer Science*, vol. 2709, pp. 176–185.
[69] D. Parikh, M. T. Kim, J. Oagaro, S. Mandayam, and R. Polikar, "Combining classifiers for multisensor data fusion," in *Proc. Int. Conf. Syst., Man and Cybern.*, The Hague, The Netherlands, 2004, vol. 2, pp. 1232–1237.
[70] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 4, pp. 497–508, Nov. 2001.
[71] R. Polikar, L. Udpa, S. Udpa, and V. Honavar, "An incremental learning algorithm with confidence estimation for automated identification of NDE signals," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 51, no. 8, pp. 990–1001, Aug. 2004.
[72] P. J. Boland, "Majority system and the Condorcet jury theorem," *Statistician*, vol. 38, no. 3, pp. 181–189, 1989.
[73] D. Berend, and J. Paroush, "When is Condorcet's jury theorem valid?" *Soc. Choice Welfare*, vol. 15, no. 4, pp. 481–488, 1998.
[74] C. L. Blake and C. J. Merz, *UCI Repository of Machine Learning Database at Irvine CA*, 2005.
[75] X. E. Gros, *Applications of NDT Data Fusion*. New York: Springer-Verlag, 2001.
[76] P. E. Mix, *Introduction to Nondestructive Testing: A Training Guide*, 2nd ed. New York: Wiley, 2005.

**Devi Parikh** received the B.Sc. degree in electrical and computer engineering from Rowan University, Glassboro, NJ, in 2005. She is currently working toward the Ph.D. degree at Carnegie Mellon University, Pittsburgh, PA.

Her research interests include pattern recognition, machine learning, and computer vision. Apart from using an ensemble of classifiers for data fusion, she has worked on using pattern recognition techniques for intrusion detection and on feature-based retrieval for 3-D reassembly.

Ms. Parikh is the recipient of a National Science Foundation Graduate Research Fellowship.

**Robi Polikar** (S'93–M'00) received the B.Sc. degree in electronics and communications engineering from Istanbul Technical University, Istanbul, Turkey, in 1993, and the M.Sc. and Ph.D. degrees both in electrical engineering and biomedical engineering from Iowa State University, Ames, in 1995 and 2000, respectively.

He is currently an Associate Professor of electrical and computer engineering at Rowan University, Glassboro, NJ. His current research interests are pattern recognition, ensemble systems, computational models of learning, incremental learning, and their applications in biomedical engineering. His current work is funded primarily through NSF's CAREER program and NIH's Collaborative Research in Computational Neuroscience program.

Dr. Polikar is a member of the American Society for Engineering Education, Tau Beta Pi, and Eta Kappa Nu.