Dynamic Probability Estimator for Machine Learning

Janusz A. Starzyk, *Senior Member, IEEE,* and Feng Wang

***Abstract*—An efficient algorithm for dynamic estimation of probabilities without division on unlimited number of input data is presented. The method estimates probabilities of the sampled data from the raw sample count, while keeping the total count value constant. Accuracy of the estimate depends on the counter size, rather than on the total number of data points. Estimator follows variations of the incoming data probability within a fixed window size, without explicit implementation of the windowing technique. Total design area is very small and all probabilities are estimated concurrently. Dynamic probability estimator was implemented using a programmable gate array from Xilinx. The performance of this implementation is evaluated in terms of the area efficiency and execution time. This method is suitable for the highly integrated design of artificial neural networks where a large number of dynamic probability estimators can work concurrently.**

***Index Terms*—Classification, entropy, machine learning, neural network hardware, probability estimator.**

I. Introduction

HARDWARE implementation of neural networks and learning machines gains popularity as integrated circuits are now being fabricated in deep submicron technologies and access to powerful programmable devices like FPGAs [1]–[3], facilitates design of complex systems. Machines use different measures to estimate the quality of learning and the ability of extracted features to solve a given classification task [4]–[6]. For instance, statistical learning uses information theory to determine the entropy of the training data and apply it to estimate the quality of learning [7]–[9]. This, in turn, requires estimation of probabilities based on the training data [8]–[10]. Probability estimates are critical for fining such useful learning measures as classic Kullback-Leibler information number and log-likelihood [11]. In addition, other machine learning methods like kernel based clustering [12], independent component analysis [13], [14], a-posteriori parameter estimation [15], blind signal separation [16], [17], and activation function neurons [18] benefit from probability estimation of training data.

Proportions may be used to provide such estimates, and confidence intervals, which depend on the size of the training data set, determine the estimation accuracy [5], [19]. Learning machines, like neural networks, use many computing processing units working in parallel [3], [20]. In such machines, proportions must be estimated using simple architecture to save the machine cost and increase the integration level.

In general, calculating proportion requires a counter and a divider. While counters have relatively simple hardware, dividers are complex and may be too slow for real time calculations. For instance a single 8-bit divider requires 61 LUTs and eight clock cycles to divide [21]. Another problem is the limited size of counters, which limits the maximum number of data that can be used for training. On-line machine learning must accept any number of training data, therefore no specific preset counter size can satisfy all possible applications. Finally, if learning is organized within dynamically changing environment, when proportions vary with time, an approach is needed to have a dynamic estimation of probabilities. While this could be accomplished using a walking window approach, its implementation would require dynamical storage of the counter values in the memory with size proportional to the window size. This, in turn, would require additional hardware resources. This paper presents a method and its hardware implementation to estimate probabilities in open learning environment, where no specified limits exist on the amount of the training data. No divider is required and probabilities can be adjusted dynamically.

Manuscript received April 15, 2003; revised October 23, 2003.

The authors are with the School of Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701 USA (e-mail: Starzyk@bobcat.ent.ohiou.edu).

Digital Object Identifier 10.1109/TNN.2004.824254

II. Estimating Probabilities

Let us first consider a stationary case, when the estimated probabilities are constant. Let us assume a fixed number of classes c with probabilities $p_1, p_2, \ldots, p_c$ and assume that

$$\sum_{i=1}^{c} p_i = 1. \tag{1}$$

These probabilities can be estimated by proportions

$$p_i \cong \frac{n_i}{n_{\text{tot}}}, \quad i = 1, 2, \ldots, c \tag{2}$$

where, n_i are counts for individual classes and n_{tot} is the total count. Assuming that the training data are uncorrelated, and that they are presented sequentially one at a time, the proportions will converge to probabilities with the confidence intervals that depend on the number of training data. In simple arithmetic units with a finite size counters (for instance 8 bits), there is a problem of counter overflow.

To address this problem, let us first analyze the case when the total sum reached its maximum count $n_{\max} = n_{\text{tot}}$ and any increase would cause an overflow. If this happens, simply divide the total count n_{tot} and count of all classes n_i by 2 and continue counting. Each time the overflow occurs, n_{tot} is reduced to $n_{\max}/2$ and the count continues until the next overflow condition. We define the observation period between two successive overflows as the **overflow interval**.

Division by 2 is very easy in hardware and this operation would correctly estimate proportions with the confidence interval depending on the maximum count $n_{\max}$ of the total counter. This method introduces a small random error if the fractional part is truncated, however, unlike in the random walk,

1045-9227/04$20.00 © 2004 IEEE

this error does not accumulate, since the sum and the error are divided by 2 at each overflow.

To prove this assertion let us consider the effect of a single such division step on the accuracy of the estimated proportion. After division by 2, the count for each class is replaced by $\boldsymbol{n}_{i2}+\boldsymbol{n}_{i1}/2 (i=1,2,\ldots,c)$ and the total count is replaced by $\boldsymbol{n}_{\mathrm{tot}2}+\boldsymbol{n}_{\mathrm{tot}1}/2$ where $\boldsymbol{n}_{i1}$ and $\boldsymbol{n}_{\mathrm{tot}1}$ are the class and total count before the overflow and $\boldsymbol{n}_{i2}$ and $\boldsymbol{n}_{\mathrm{tot}2}$ are the class and total count after the overflow. Using these values the class probabilities are estimated from

$$p_i \cong \frac{\boldsymbol{n}_i}{\boldsymbol{n}_{\mathrm{tot}}} = \frac{\boldsymbol{n}_{i2}+\frac{\boldsymbol{n}_{i1}}{2}}{\boldsymbol{n}_{\mathrm{tot}2}+\frac{\boldsymbol{n}_{\mathrm{tot}1}}{2}} = \frac{\boldsymbol{n}_{i2}+\boldsymbol{n}_{i1\_1}+\frac{\varepsilon_{i1}}{2}}{\boldsymbol{n}_{\mathrm{tot}2}+\boldsymbol{n}_{\mathrm{tot}1\_1}+\frac{\varepsilon_{\mathrm{tot}1}}{2}} \quad i=1,2,\ldots,c \tag{3}$$

where $\boldsymbol{n}_{i1\_1}$ and $\boldsymbol{n}_{\mathrm{tot}1\_1}$ are the truncated values obtained from $\boldsymbol{n}_{i1}$ and $\boldsymbol{n}_{\mathrm{tot}1}$ after division by 2 (in hardware obtained by using right shift by 1 bit). Truncation errors ε_{i1} and $\varepsilon_{\mathrm{tot}1}$ are equal to 0 or 1, depending on the least significant bit of $\boldsymbol{n}_{i1}$ and $n_{\mathrm{tot}1}$, respectively. Notice that $(n_{\mathrm{tot}1\_1}+\varepsilon_{\mathrm{tot}1}/2) = n_{\max}/2$ and $n_{\mathrm{tot}2} \leq n_{\max}/2$.

After a new total value reaches $n_{\max}$ a new overflow interval begins, all previously counted values are divided by 2, and new probabilities are estimated from

$$p_i \cong \frac{n_i}{n_{\mathrm{tot}}} = \frac{n_{i3}+n_{i2\_1}+n_{i1\_2}+\frac{\varepsilon_{i2}}{2}+\frac{\varepsilon_{i1}}{4}}{n_{\mathrm{tot}3}+n_{\mathrm{tot}2\_1}+n_{\mathrm{tot}1\_2}+\frac{\varepsilon_{\mathrm{tot}2}}{2}+\frac{\varepsilon_{\mathrm{tot}1}}{4}} \quad i=1,2,\ldots,c \tag{4}$$

where n_{i1_2} and $n_{\mathrm{tot}1\_2}$ are the truncated values obtained from n_{i1_1} and $n_{\mathrm{tot}1\_1}$ after division by 2. In addition, we have $n_{\mathrm{tot}2\_1}+n_{\mathrm{tot}1\_2}+(\varepsilon_{\mathrm{tot}2}/2)+(\varepsilon_{\mathrm{tot}1}/4) = n_{\max}/2$ and $n_{\mathrm{tot}3} \leq n_{\max}/2$.

In general, after $\boldsymbol{w}$ steps of such process (i.e., in the w-th overflow interval) class probabilities are estimated from the following:

$$p_i \cong \frac{n_i}{n_{\mathrm{tot}}} = \frac{\sum_{k=0}^{w-1} n_{i(w-k)\_k}+\sum_{k=1}^{w-1}\frac{\varepsilon_{i(w-k)}}{2^k}}{\sum_{k=0}^{w-1} n_{\mathrm{tot(w-k)\_k}}+\sum_{k=1}^{w-1}\frac{\varepsilon_{\mathrm{tot(w-k)}}}{2^k}} \quad i=1,2,\ldots,c \tag{5}$$

where $n_{i(w-k)\_k}$ and $n_{\mathrm{tot(w-k)\_k}}$ are the truncated values obtained from $n_{i1(w-k)}$ and $n_{\mathrm{tot(w-k)}}$ after right shift by k bits. Notice, that since each $\varepsilon_{i(w-k)} \leq 1$ and each $\varepsilon_{\mathrm{tot(w-k)}} \leq 1$, then the error series in the numerator and denominator have an upper bound of a geometrical series $\sum_{k=1}^{w-1} 1/2^k \leq 1$, so both error terms are limited by 1.

Lemma 1: Errors in probability estimation due to the rounding of division in the described procedure do not accumulate when counting continues over many overflow intervals. Errors to the estimated values $\boldsymbol{n}_i$ and $\boldsymbol{n}_{\mathrm{tot}}$ in (5) have an upper bound of 1.

Proof: So far, we assumed that r.h.s. expression in (5) indeed approximates the probability of each class data. To prove Lemma 1 let us analyze the case of each class having a specified fixed probability value p_i. Since each $n_{i(w-k)}$ represents its corresponding class count and $n_{\mathrm{tot(w-k)}}$ represents the total count added after the last overflow, then their individual rates also approximate class probabilities $p_i \cong (\boldsymbol{n}_{i(w-k)}/\boldsymbol{n}_{\mathrm{tot(w-k)}})$. Using this we can now estimate the following expression:

$$R = \frac{\sum_{k=0}^{w-1} n_{i(w-k)\_k}+\sum_{k=1}^{w-1}\frac{\varepsilon_{i(w-k)}}{2^k}}{\sum_{k=0}^{w-1} n_{\mathrm{tot(w-k)\_k}}+\sum_{k=1}^{w-1}\frac{\varepsilon_{\mathrm{tot(w-k)}}}{2^k}} \cong \frac{\sum_{k=0}^{w-1} n_{i(w-k)\_k}+\varepsilon_i}{\sum_{k=0}^{w-1} n_{\mathrm{tot(w-k)\_k}}+\varepsilon_{\mathrm{tot}}} \cong p_i \frac{\sum_{k=0}^{w-1} n_{\mathrm{tot(w-k)\_k}}+\frac{\varepsilon_i}{p_i}}{\sum_{k=0}^{w-1} n_{\mathrm{tot(w-k)\_k}}+\varepsilon_{\mathrm{tot}}} \cong p_i(1+\varepsilon) \quad i=1,2,\ldots,c \tag{6}$$

where $\varepsilon_i < 1$ and $\varepsilon_{\mathrm{tot}} < 1$.

Consider the error term in (6)

$$\varepsilon = \frac{\frac{\varepsilon_i}{p_i}-\varepsilon_{\mathrm{tot}}}{\sum_{k=0}^{w-1} n_{\mathrm{tot(w-k)\_k}}+\varepsilon_{\mathrm{tot}}} \leq \frac{\frac{\varepsilon_i}{p_i}}{n_{\mathrm{tot}}}. \tag{7}$$

Since $p_i \cong (n_i/n_{\mathrm{tot}}) \gg (1/n_{\mathrm{tot}})$, then $\varepsilon \ll \varepsilon_i$. The class probability estimation has the largest error when probabilities are close to $(1/n_{\max})$, but these (and lower) probabilities cannot be reliably estimated using a counter with the maximum count $n_{\max}$, since their confidence intervals are large, as discussed in Section II-B.

In addition, the method effectively implements a windowing approach, since the most recent data probability affects the estimated proportions the most, and the historical data loses its importance exponentially with the number of overflows.

Lemma 2: The importance of historical data as expressed by their estimate of a class probability is exponentially reduced with each overflow.

Proof: Consider the estimate of a class probability expressed by (6) and replace each $n_{i(w-k)}$ using $p_{i(w-k)}n_{\mathrm{tot(w-k)}}$, where $p_{i(w-k)}$ is ith class probability estimated using count in $(w-k)$ overflow interval. Assume, without loosing any generality that all $\boldsymbol{n}_{\mathrm{tot(w-k)}} = \boldsymbol{n}_{\max}/2$. Then

$$R \cong \frac{\sum_{k=0}^{w-1} n_{i(w-k)\_k}+\varepsilon_i}{\sum_{k=0}^{w-1} n_{\mathrm{tot(w-k)\_k}}+\varepsilon_{\mathrm{tot}}} \cong \frac{\sum_{k=0}^{w-1} p_{i(w-k)} n_{\max} 2^{-k-1}}{\sum_{k=0}^{w-1} n_{\max} 2^{-k-1}} \cong \frac{\sum_{k=0}^{w-1} p_{i(w-k)} n_{\max} 2^{-k-1}}{n_{\max}(1-2^{-w})} \cong \frac{\sum_{k=0}^{w-1} p_{i(w-k)} 2^{-k-1}}{1-2^{-w}}. \tag{8}$$

Since the historical data correspond to samples collected in earlier overflow intervals, they have larger k values. Their probabilities are weighted lower as they are multiplied by 2^{-k-1} in (8).

The effective discounting of earlier samples allows for adjustment of probability estimates in case the individual class probability changes over time. However, unlike the sliding window

approach, where discounting of historical data is binary, in this algorithm, a group of historical data receives exponentially declining weight, until their weight is truncated to zero after $\boldsymbol{b}$ overflow intervals, where b is the number of bits in the counter used. The window obtained is also a sliding window, although each time the window slides by $\boldsymbol{n}_{\max}/2$ data points and keeps exponentially declining weight of the past data. We define this kind of windowing a **soft window**.

One problem with this approach is a need of divider to determine proportions. However, when the total count is constant and equals to $n_{\max}$ division is not necessary, as each count n_i represents the fractional part of the corresponding proportion. The following algorithm accomplishes the task of keeping the total count near $n_{\max}$, while adjusting individual counts according to the current estimate of their proportions.

A. Basic Dynamic Probability Estimator Algorithm (BDPE)

In the following BDPE algorithm, class probabilities p_i are estimated as $(n_i/n_{\max})$. Therefore, with $n_{\max}$ set as a power of 2, no division is required.

1) Set the class count and the cumulative sums of all the classes to zero, $n_i = s_i = 0$, for $i = 1, 2, \ldots, c$;
2) For a training data d_t determine the class k to which this data belongs;
3) Increase the class count n_k by one;
4) Increase the cumulative sums of all the classes $s_i = s_i + n_i$, $i = 1, 2, \ldots, c$;
5) For all the classes, if $s_i > n_{\max}$ set $s_i = s_i - n_{\max} - 1$ and reduce the corresponding count n_i by one;
6) If no more training data, stop, else go to 2.

Some insight into the working of this algorithm may be gained by understanding the meaning of the cumulative sums of all the classes' s_i and their role in adjusting the sample count for each class. The major idea behind this algorithm is to keep the total sum $n_{\text{tot}} \cong n_{\max}$.

Let us first prove numerical stability of the algorithm.

Lemma 3: BDPE algorithm employs a self correcting mechanism for each class count n_i producing a steady state in which

$$n_i \cong \hat{n}_i \text{ where } \hat{n}_i = p_i n_{\max}. \tag{9}$$

Proof: Suppose that in contradiction to Lemma 3, n_i is consistently greater than $\hat{n}_i$, $n_i = \hat{n}_i \Delta_j$, $\Delta_j > 0$, where j is an iterative step of the algorithm. The algorithm adjusts the sample count of each class by -1 after t_i iterative steps when the cumulative sum for this class reaches $n_{\max}$.

$$s_i = \sum_{j=1}^{t_i} n_i = \sum_{j=1}^{t_i} (\hat{n}_i + \Delta_j) = n_{\max} > t_i \hat{n}_i = t_i p_i n_{\max} \tag{10}$$

from which we have $1 > t_i p_i$.

Let us consider the expected value of an adjustment made to a sample count by this algorithm after t_s iterative steps. Since the algorithm reduces n_i by 1 after t_i iterative steps, then the expected reduction is

$$D = \frac{t_s}{t_i} > t_s p_i \tag{11}$$

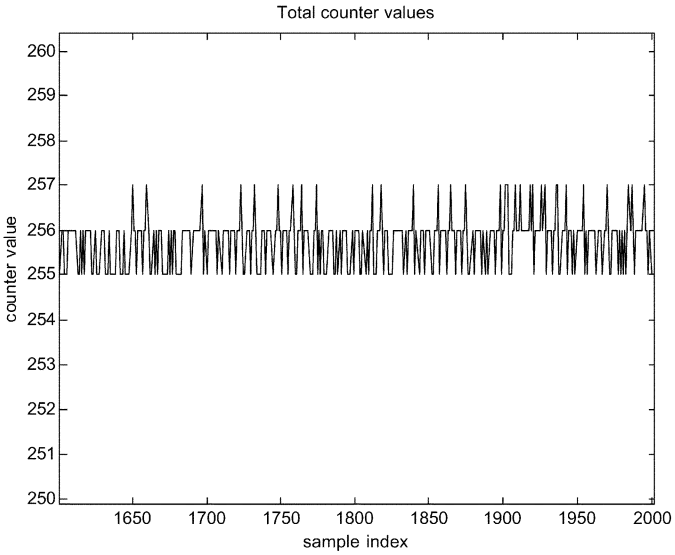

Fig. 1. Total count values in a steady state for 8-bit counters.

while the expected increase in the class count resulting from its frequency of occurrence p_i is $A = t_s p_i$. So, the expected new value of the sample count is in general smaller than the old value as

$$n_{i\,\text{new}} = \hat{n}_i + \Delta_j + A - D < \hat{n}_i + \Delta_j = n_{i\,\text{old}}. \tag{12}$$

In a similar way we can observe that the opposite adjustment happens when n_i is consistently smaller than $\hat{n}_i$, thus the algorithm keeps each sample stable and its value oscillates around $\hat{n}_i = p_i n_{\max}$.

It is easy to show that the total count is kept near $n_{\max}$ as $n_{\text{tot}} = \sum_{i=1}^{c} n_i \cong \sum_{i=1}^{c} p_i n_{\max} = n_{\max}$.

As illustrated in Fig. 1, this algorithm produces stable total sum, which allows us to use the individual class counts as proportions without division. Notice that this total count was never set or checked by the algorithm explicitly.

Total count values presented in Fig. 1 are shown as a function of the sample index when the sample index significantly exceeds the maximum count value $n_{\max}$ set to 256 in this example (for 8-bit counters). The following example illustrates the use of the BDPE algorithm.

Example 1: Fig. 2 shows the result of an example simulation of the BDPE algorithm to estimate class probabilities in a four-class problem. In this example the maximum count $n_{\max}$ was set to 256 which corresponds to an 8 bit counter. The training sample index was set to 2000, well beyond the maximum counter size. In this example data points from four classes were generated with class probabilities set to 0.1, 0.2, 0.5, and 0.2 respectively. Fig. 3 shows the corresponding counts of each class counter as a function of training sample index.

The jitter in the total count values observed in Fig. 1 is related to the mechanism of adjusting the individual classes count n_i in the BDPE algorithm. In general, a class probability in any overflow interval is not a rational number and can be expressed by

$$p_i \cong \frac{\hat{n}_i}{n_{\max}} = \frac{n_i \pm \varepsilon_i}{n_{\max}}, \quad 0 < \varepsilon_i < 1 \tag{13}$$

where n_i is integer.

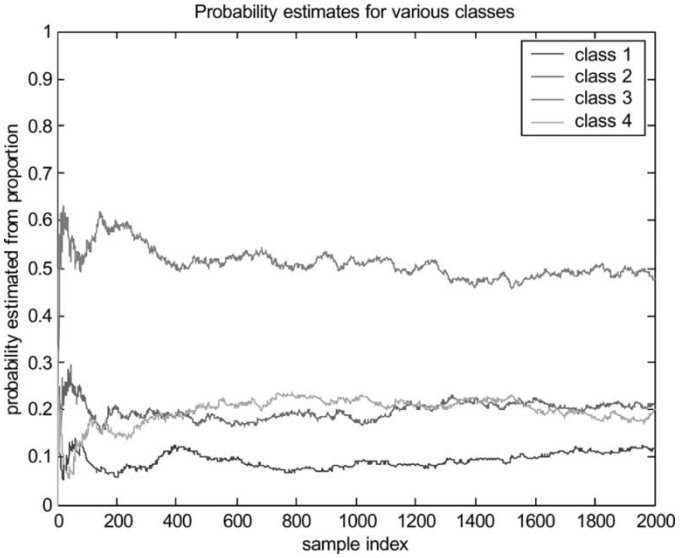

Fig. 2. Probabilities estimated from proportions.

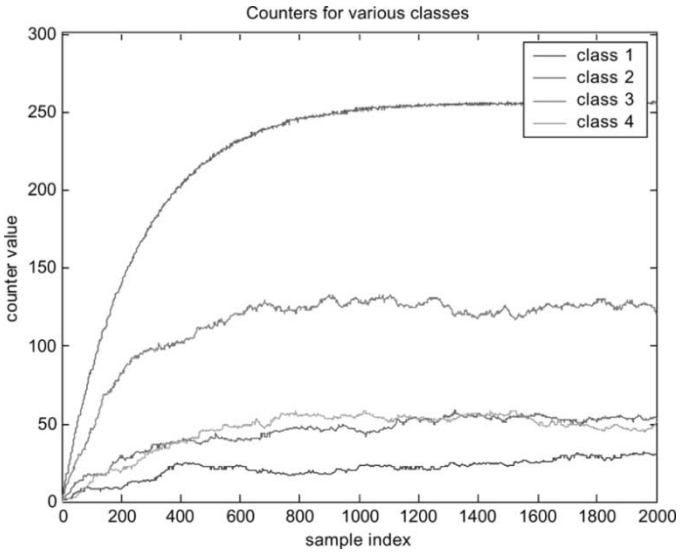

Fig. 3. Counter values n_i for all classes and the total count $n_{\rm tot}$.

For instance, if n_i is used to calculate the cumulative sum instead of $n_i - \varepsilon_i$, the cumulative sum is overestimated, and according to Lemma 3, it will be adjusted downwards. These adjustments cause a small integer noise in the total count value. This noise increases slightly when the number of classes under observation is larger, due to the increased probability that two or more classes will have they count simultaneously overestimated or underestimated. This is illustrated in Fig. 4, where total count is displayed as a function of the sample index for eight and 12 classes respectively. These results were obtained with the counter size of 10 bits.

Under the assumption that individual class errors are statistically independent, we can estimate an error of total count by

$$\varepsilon_{\rm tot} = \sqrt{\sum_{i=1}^{c} \varepsilon_i^2}. \tag{14}$$

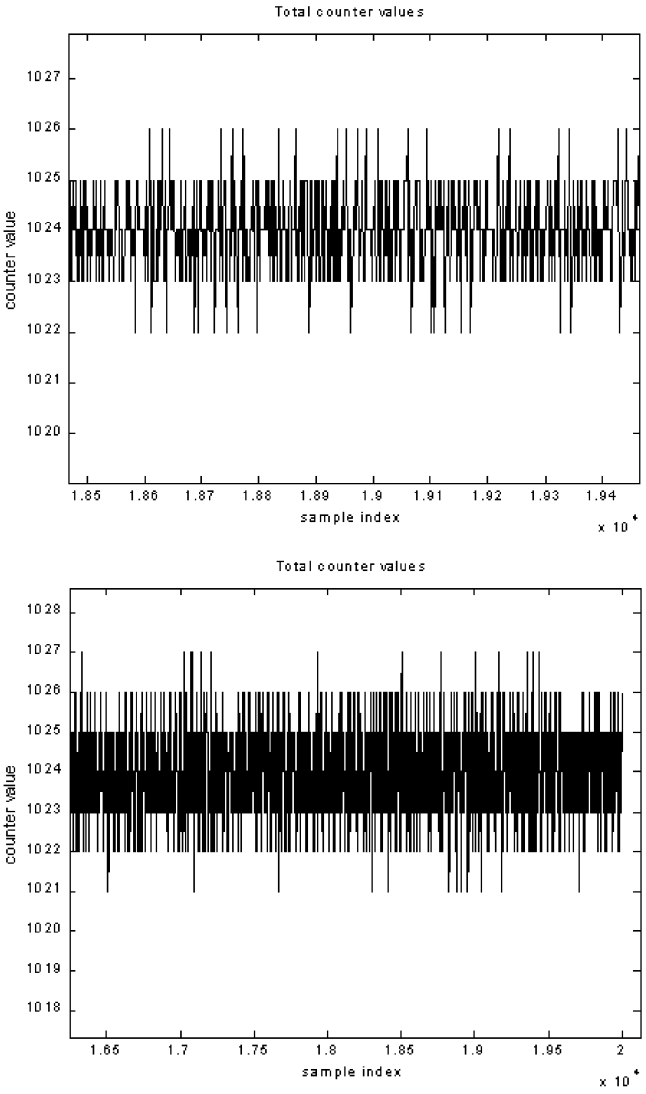

Fig. 4. Total count values for different number of classes.

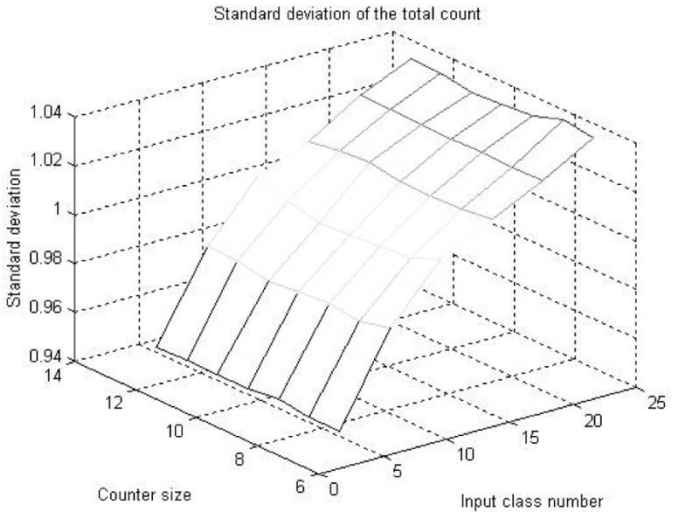

Fig. 5. Standard deviation of the total count for various counter size and number of classes.

This is illustrated in Fig. 5, where standard deviation of the total count is shown as a function of the class number and the

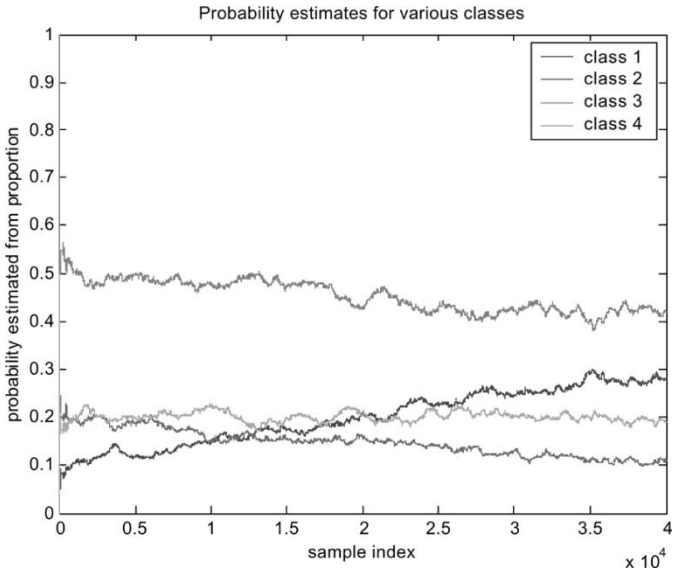

Fig. 6. Class probabilities estimated by 10-bit counters.

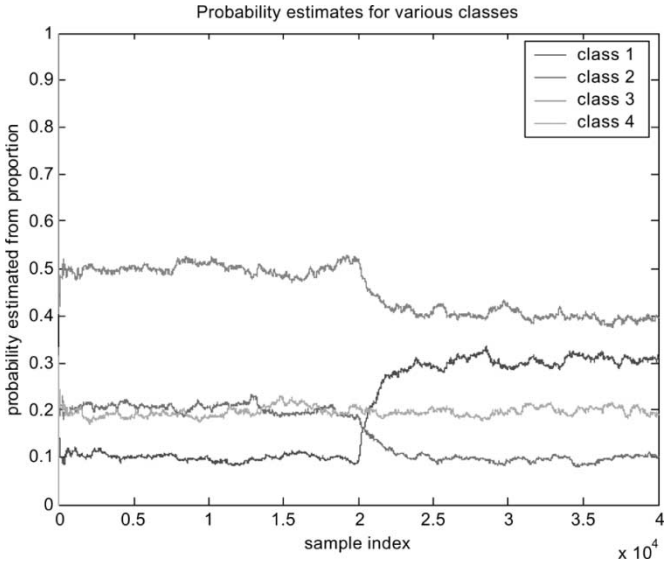

Fig. 7. Class probabilities with an abrupt change estimated by 10 bit counters.

counter size. Notice that the absolute value of the total count error does not depend on the counter size. This square root dependence of the standard deviation of the total count from $n_{\max}$ is a confirmation that individual class errors ε_i are statistically independent.

In all cases the increase in the counter size improves the resolution of the process, as the total count remains stable and the confidence intervals are smaller as discussed in Section II-C.

B. Probability Adjustments in BDPE

The developed method of estimating proportions satisfies adaptability requirement by discounting the historical data with the soft window approach. This means that the probability estimation follows the changes in a probability value. This is illustrated in the following example:

Example 2: Fig. 6 shows simulation results in which probabilities of classes 1,2, and 3 were changing linearly from 0.1–0.3, from 0.2–0.1, and from 0.5–0.4, respectively and class 4 probability was set to 0.2. As we can see, probability estimates follows the changes in the probability values. These results were obtained with 40 000 training data and 10-bit counters.

However, the changes of probability values are fully implemented only after the soft window moves to the new sample location. This means that in the transition period, probabilities reflect walking average values between old and new probability estimates.

This is illustrated in Fig. 7, where after 20 000 samples, probabilities of classes 1,2, and 3 changed abruptly from 0.1, 0.2, and 0.5 to 0.3, 0.1, and 0.4, respectively.

As we can see, the probabilities were fully adjusted to the new values after about 3000 samples that correspond to three times $n_{\max}$ for this counter size.

C. Confidence Interval

An important practical problem is the estimation of the confidence interval that sets limits for a probability calculated from proportion with a set degree of statistical confidence in the result. Many methods to estimate this interval exists in the literature from a classical Wilson interval [22], through Clopper-Pearson equal tail binomial test [23], arcsine interval introduced by Bickel [24], logit transform described by Stone [25], with modifications by Cox [26] and Sunter [27], the likelelihood ratio interval by Rao [28], to a simple and stable Agresti-Coull interval [29]. A recent paper by Brown *et al.* [30] clarifies some of the misunderstandings in the confidence interval estimation, pointing out a chaotic coverage properties of the standard and widely used Wald confidence interval [29].

We compare accuracy of the results obtained in our work against the Agresti-Coull interval, which is recommended in [30] for the number of samples larger than 40, as relatively simple to compute, and preferable over the standard Wald interval, even for the small sample size (see also [31]). For $100(1-\alpha)\%$ statistical confidence the upper and lower limits of the confidence intervals using the method of Agresti and Coull are given by

$$p_{\text{upper}} = \frac{p + \frac{z^2_{\alpha/2}}{2n} + z_{\alpha/2}\sqrt{\frac{p(1-p)}{n} + \frac{z^2_{\alpha/2}}{4n^2}}}{1 + \frac{z^2_{\alpha/2}}{n}} \tag{15}$$

$$p_{\text{Lower}} = \frac{p + \frac{z^2_{\alpha/2}}{2n} - z_{\alpha/2}\sqrt{\frac{p(1-p)}{n} + \frac{z^2_{\alpha/2}}{4n^2}}}{1 + \frac{z^2_{\alpha/2}}{n}} \tag{16}$$

where p is the probability of an individual class estimated from proportion, n is the number of samples, and $z_{\alpha/2}$ denotes the variant value from the standard normal distribution such that the area to the right of the value is $\alpha/2$. For instance, $z_{\alpha/2}$ is 1.645 in case of 95% confidence interval. The limitation of these two expressions is not critical for our application, since the sample size in this algorithm is at least 256 (which corresponds to 8-bit counters).

Confidence intervals shrink with the increase in the number of samples. Fig. 8 shows the upper and the lower limit of 95% confidence interval ($\alpha/2 = 0.025$ and $z_{\alpha/2} = 1.645$) for $p =$

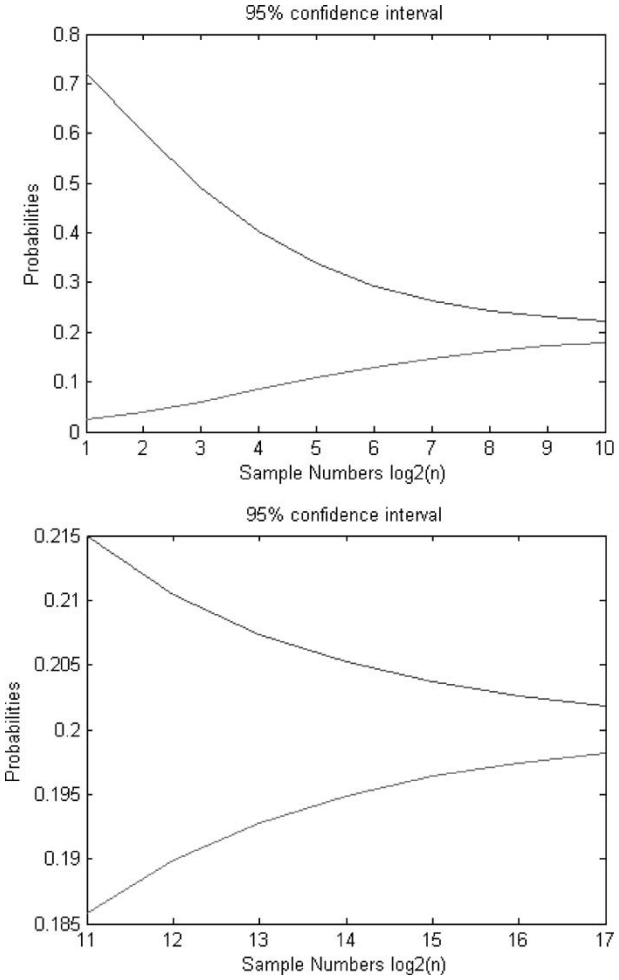

Fig. 8. Limits of the confidence intervals for $p = 0.2$.

0.2 as a function of the logarithm of the sample number (number of the counter bits).

Fig. 9 shows the size of the 95% confidence interval (computed as a difference between the upper and the lower limit deviation from the estimated probability) for selected probability values as a function of the logarithm of the sample number.

Accuracy of the probability estimates in the proposed algorithm is limited by the precision of the counters used. For instance, in Example 1, we used 8-bit precision, so the size of the confidence interval is set by the maximum count (256) rather than the number of samples used (2000). According to the estimates from Fig. 9, with probability set to 0.2 and 256 samples, we can expect the accuracy of estimate to stay within ± 0.04. This accuracy was attained by estimates shown in Fig. 2.

However, if the same data was used to estimate probabilities with 12-bits counter, then the probability estimates were much closer to the specified values as shown on Fig. 10.

Probability estimate accuracy further increases as the counter size grows as is shown in Fig. 11, where probability estimates of classes 2 and 4 are shown for the counter of 16 bits. As we can see, they are estimated with the relative error of ± 0.002 which is within the limits of the confidence interval for 2^{16} sample size (see Fig. 8).

For smaller number of samples, probability estimates are within the limits of the confidence interval evaluated from (15) and (16). The full precision of the proposed algorithm is reached only after the total number of samples received reaches $n_{\max}$. Once this limit is reached, the precision does not increase with additional samples received.

III. Dealing With Initial Conditions

As can be observed on Fig. 3, until the total sum reaches its maximum value, individual counts do not represent probabilities, and during this phase division by n_{tot} is required to estimate the probabilities. To alleviate this problem, we modified the DBPE algorithm. The idea behind this modification is to dynamically adjust the individual class counts n_i such that $n_{\text{tot}} = \sum_{i=1}^{c} n_i$ can reach $n_{\max}$ sooner than in DBPE algorithm. Therefore, in steps 3. and 5. of the probability estimation algorithm we will increment and decrement n_i by c_i instead of 1. Initially, c_i is set to $n_{\max}$, and gradually (with the increase of the sample index) it is reduced to 1. It is equivalent to assigning the corresponding class probability to 1 after the first observed sample. Then, if two samples are observed from two different classes, each class probability is reduced to 1/2, with four samples observed, probabilities are assigned based on the samples distribution, and the adjustment of each probability is by 1/4, and so on. As the number of samples grows, adjustments of probabilities expressed by changes in the value of n_i become finer and finer by using smaller increment counter c_i, until c_i reaches the final value of 1. The changes of c_i are by the powers of 2 to avoid division. The resulting algorithm is as follows:

A. DPE Algorithm

1) Set the class count and the initial cumulative sums of all the classes to zero $n_i = s_i = 0$, for $i = 1, 2, \ldots, c$, and set the counter increment c_i to $n_{\max}$, set the deceleration index d_i to 1, set the deceleration counter d_c to 1;
2) For a training data d_t determine the class k to which this data belongs;
3) Increase the class count n_k by c_i;
4) Increase the cumulative sums of all the classes by n_i, $s_i = s_i + n_i$, $i = 1, 2, \ldots, c$;
5) For all the classes, if $s_i > n_{\max}$ set $s_i = s_i - n_{\max} - c_i$ and reduce the corresponding class count n_i by c_i;
6) If the deceleration counter equals to the deceleration index, set $c_i = \max(c_i/2, 1)$, $d_i = n_{\max}/c_i$, $d_c = 0$;
7) If no more training data, stop, else increase the deceleration counter d_c by 1 and go to 2.

In this algorithm, the deceleration index specifies how many samples need to be received before the counter increment is divided by 2 (to refine the probability adjustment step), and the deceleration counter counts how many samples elapsed since the last adjustment. Notice that when c_i reaches 1, then DPE algorithm continues the same way as BDPE algorithm. In the DPE algorithm all divisions are divisions by a power of 2, so they can be easily implemented by a binary shift. Results of this new algorithm are shown on Fig. 12, which shows the corresponding counts of each class counter as a function of the training sample index for 8-bit counters.

Like the original method, this algorithm produces a stable total sum (shown in Fig. 13), which allows us to use the indi-

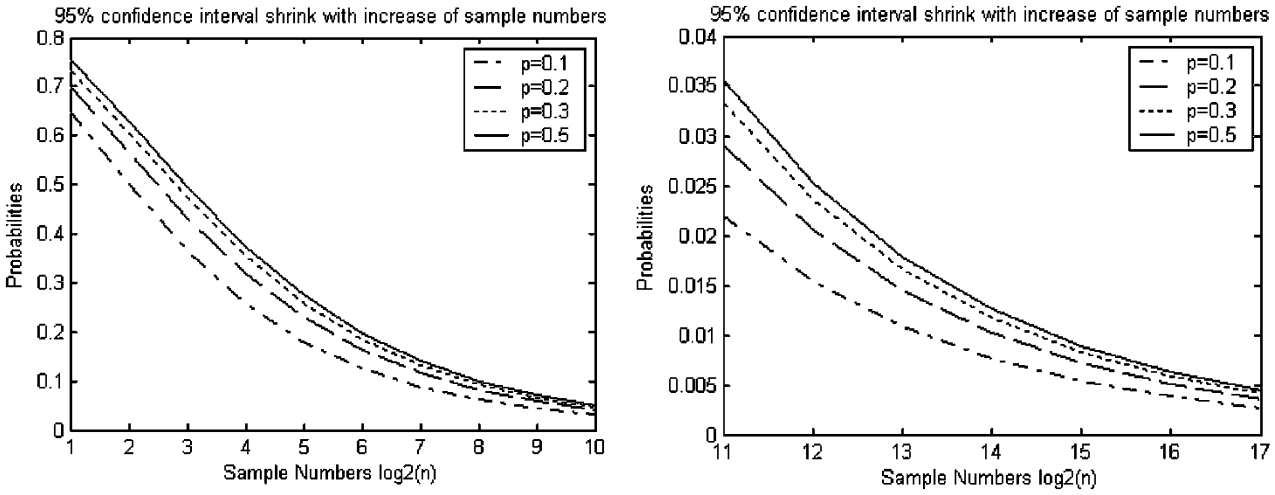

Fig. 9. 95% confidence intervals for different probabilities.

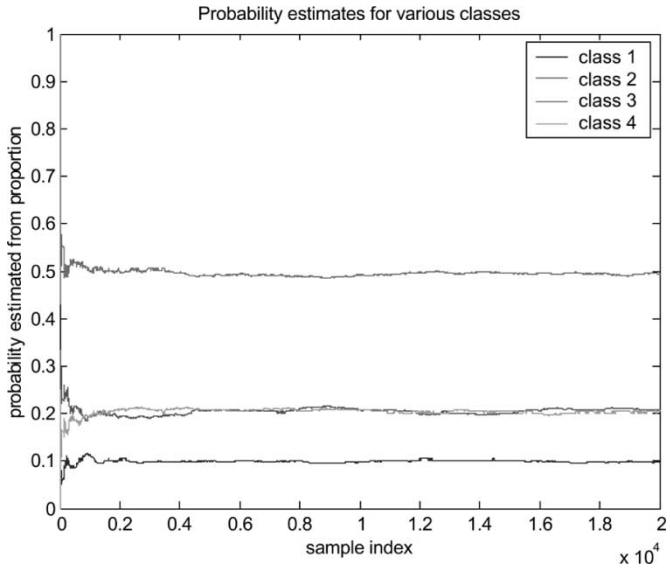

Fig. 10. Class probabilities estimated by 12-bit counters.

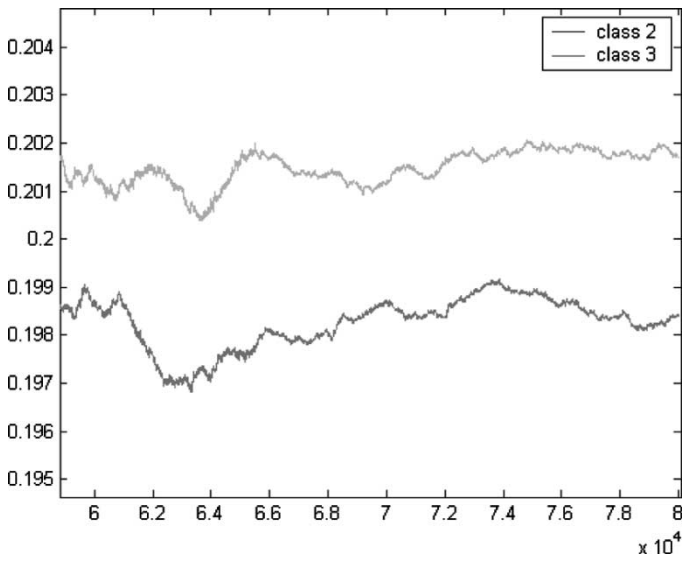

Fig. 11. Class probabilities estimated by 16-bit counters.

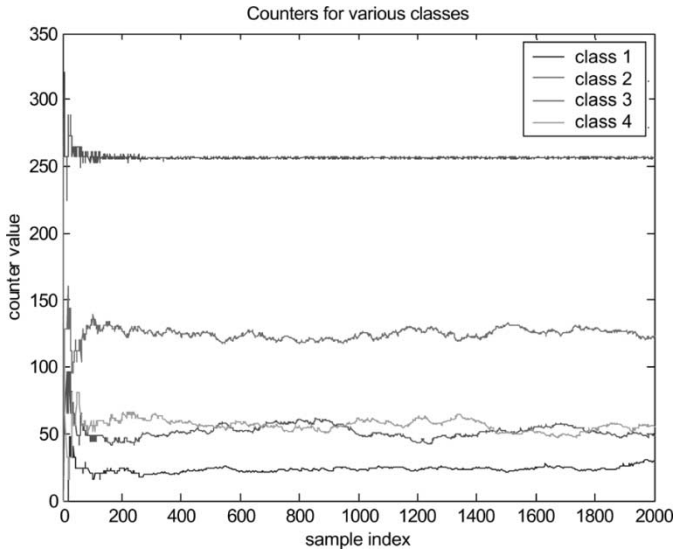

Fig. 12. Counter values n_i for all classes and the total count n_{tot} for 8-bit counters.

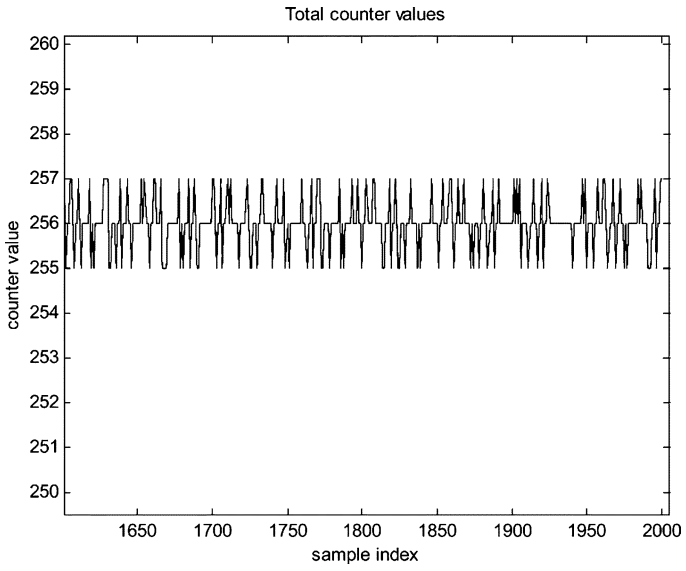

Fig. 13. Total count values in a steady state for 8-bit counters.

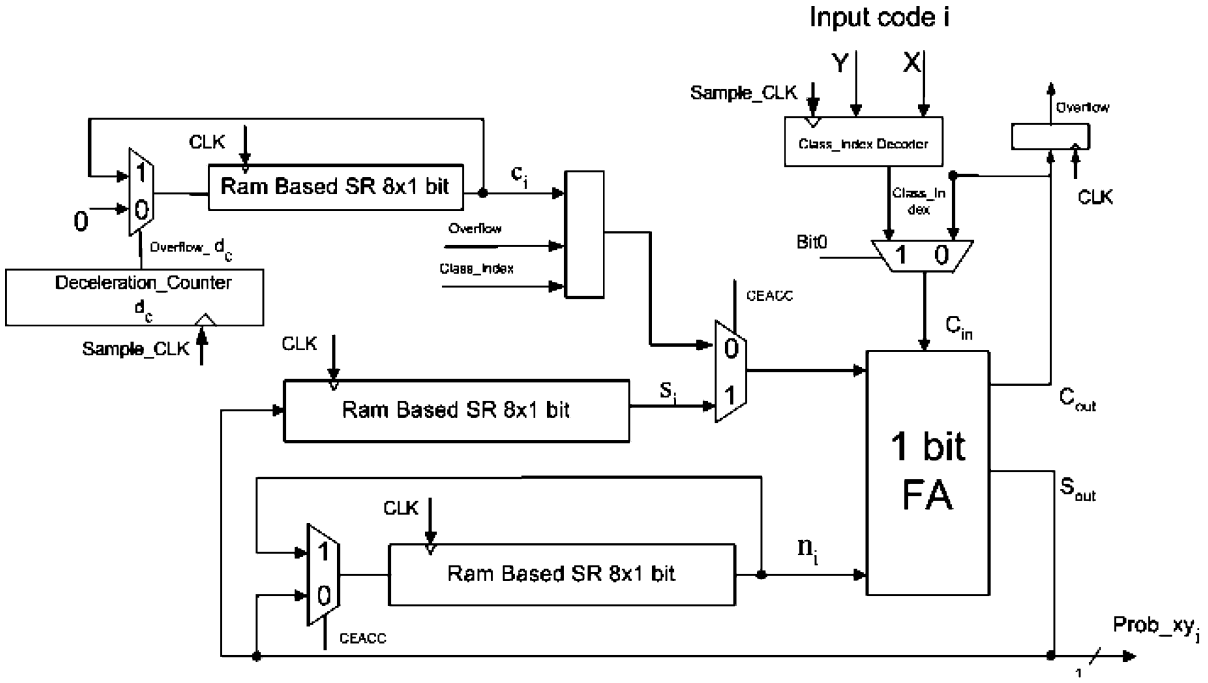

Fig. 14. DPE in a serial structure.

vidual class counts as proportions without division. However, unlike in the original method this stable total sum is reached much earlier.

IV. Hardware for DPE

Computing speed and design area is of a great concern in this implementation of the DPE. The proposed structure consists of two main parts: the process controller and probabilities estimator (PE). The process controller controls the data flow of the PEs; the PE classifies the input data and estimates probabilities of the corresponding class. The input data class is coded by n bit binary code, such that maximum of 2^n class probabilities can be estimated. The number of the PE's is in proportion to the number of classes.

A simple sequential DPE with training data d_t coded as 2-dimensional input (X, Y), and 2^2 outputs $Prob\_xy_i (i = 00, 01, 10, 11)$ that estimate probabilities of the four classes coded by the *input code* i, is shown in Fig. 14. Many classification problems involve high dimensional inputs and a large number of classes [32]. This structure can be easily expanded to n-dimensional input codes that can be used to classify 2^n classes. The training data enter the DPE at the rising edge of the sample clock signal, Sample_CLK. The sample clock cycle is defined as the time needed by the DPE to complete its calculations. CLK is the global clock which determines speed of the circuit operation. Eight-bit class probabilities are calculated and output serially from this module at the falling edge of a data valid signal.

In this serial structure, each DPE consists of two RAM based shift registers and one 1 bit full adder. These two shift registers, store the cumulative sums $\boldsymbol{s}_i$ and individual class counts $\boldsymbol{n}_i$. The class counts n_i change with the input data accumulation at various time instances. To update the values of all counters in one sample cycle, two processes are executed. The first process increases cumulative sums of all classes

$$s_i = s_i + n_i \tag{17}$$

and stores them by shifting the output of one bit adder back to s_i. If this operation causes the adder to overflow, the ***Overflow*** flag is set to 1, otherwise $Overflow = 0$. A binary signal ***Class_index*** is set to 1 when input data $\boldsymbol{d}_t$ represents given class (e.g., $(x, y) = (0, 0)$), which is detected by the *Class_index* decoder in Fig. 14, otherwise $Class\_index = 0$. The second process calculates n_i as follows [see (18) at the bottom of the page]. where $\hat{c}_i + 1 = c_i$. $\hat{c}_i$ replaces c_i defined in DPE algorithm and is initialized to $n_{\max} - 1$, since the maximum value of eight bits register is 255. $\hat{c}_i$ is divided by 2 when deceleration counter $\boldsymbol{d}_c$ overflows. The deceleration counter is initialized to $n_{\max} - 1$, increases by 1 every sample cycle, and is reloaded with the inverse of $\hat{c}_i$ when it overflows. Process control signal CEACC is set in the first process and is cleared during the second one. The count $\boldsymbol{n}_i$ is output in a serial format, and the falling edge of ***data_valid*** signal indicates validation of the count signal. At that point, all estimated probabilities can be read concurrently from the corresponding outputs.

$$n_i = \begin{cases} n_i - (\hat{c}_i + 1)^*(1 - Class\_index), & \text{when } Overflow = 1 \\ n_i + (\hat{c}_i + 1)^*(Class\_index), & \text{when } Overflow = 0. \end{cases} \tag{18}$$

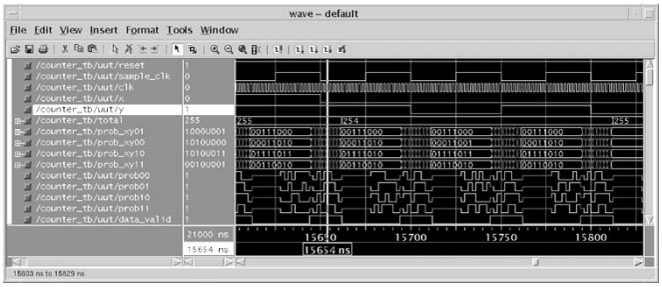

Fig. 15. Simulation results produced by the test bench.

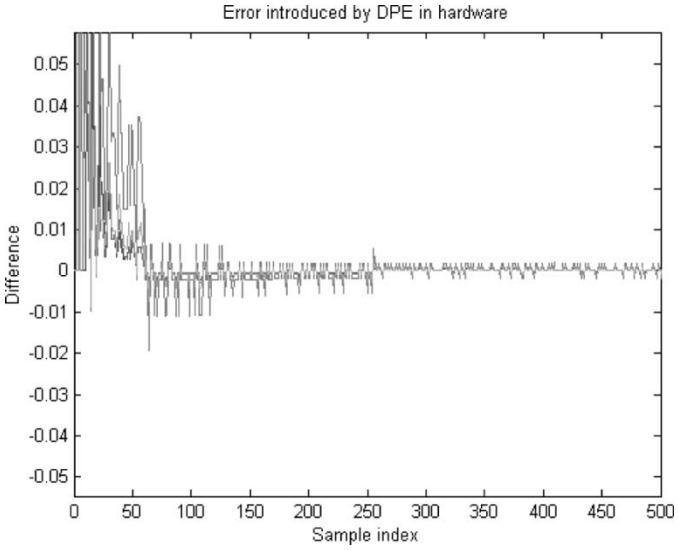

Fig. 16. Error introduced by DPE in hardware for 8-bit counters.

Functional verification and simulation were performed in Modelsim. Fig. 15 shows a snapshot of the waveform produced by the test bench *counter_tb*. The input vectors, x and y, were generated in Matlab.

Since eight-bit raw count value is used to approximate the probabilities, those approximations introduce errors in this implementation. The error induced by DPE in hardware is shown in Fig. 16. This figure shows the difference between simulated result obtained from Matlab and the hardware results. The maximum deviation is about 0.1% after 256 points, when enough training data have entered the DPE. In addition, the probabilities generated by the DPE in hardware meet the 95% confidence interval limits given by (15) and (16), as shown in Fig. 17.

A. Discussion of the Hardware Cost and Performance

The design was implemented on a Xilinx Virtex FPGA (XCV1000, speed grade -4). Fig. 18 shows the area required for the probability estimator's implementation with different number of classes incorporated in the design. The area cost is in proportion to the number of classes for probability estimation. Each class occupies from 26/4 slices to 98/32 slices, when the number of classes increases from 4 to 32.

Other straightforward methods to estimate probabilities from proportions may be used, such as the floating point accumulation and division. This alternative method requires both floating-point divider and adder, which come at a significant increase in the area comparing to the proposed approach. A single 32 bits

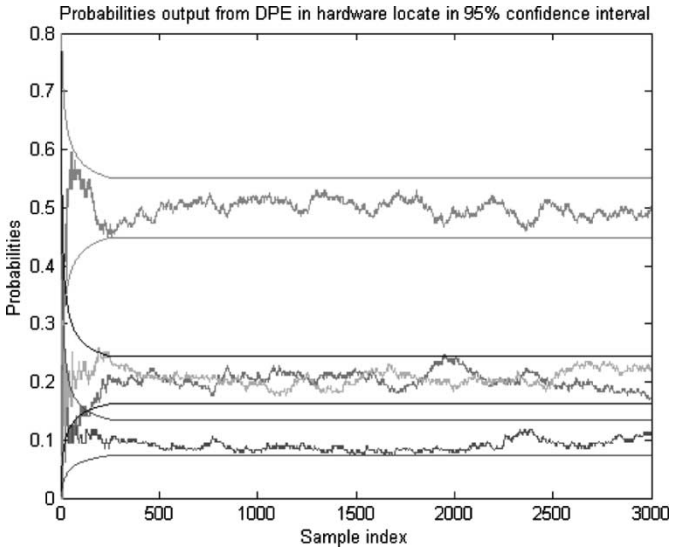

Fig. 17. Probabilities output from the DPE in the hardware for 8-bit counters.

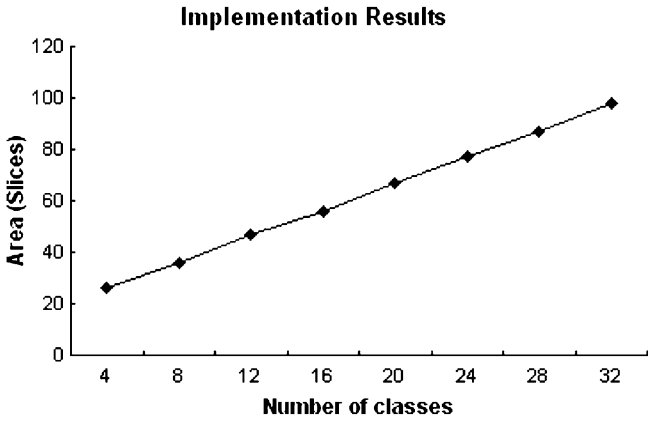

Fig. 18. Implementation results with different number of classes in terms of slices.

floating point divider implemented on Xilinx Virtex chip costs 1632 slices, which is five hundred times of the area of DPE, at the performance of 44 MHz [33]. Also the floating-point adder requires 590 slices with its implementation on Virtex chip at the performance of 46 MHz [34]. Implementing many such floating point devices for the concurrent operation of processing components would be prohibitively expensive.

Two parts of the logic, the process control logic, and the PE, contribute to the area cost of the design. Since the process control logic can be shared by all counts, the cost of each probability calculation unit decreases as the number of classes increases. The area of PE is nearly fixed. It only changes when for a large number of classes the dimensionality of the input data (number of bits of the class input code on Fig. 14), exceeds the maximum number of inputs of one LUT. This would require some increase in the design area. Combining all these factors, the area cost as a function of the number of classes is as illustrated in the Fig. 19. This area decreases with the increase of the class number. From this figure we can estimate the area increase per one class to be on the level of three slices in a Xilinx Virtex FPGA.

The timing performance of the DPE is also acceptable. It only needs 20 cycles from the rising edge of the sample clock to

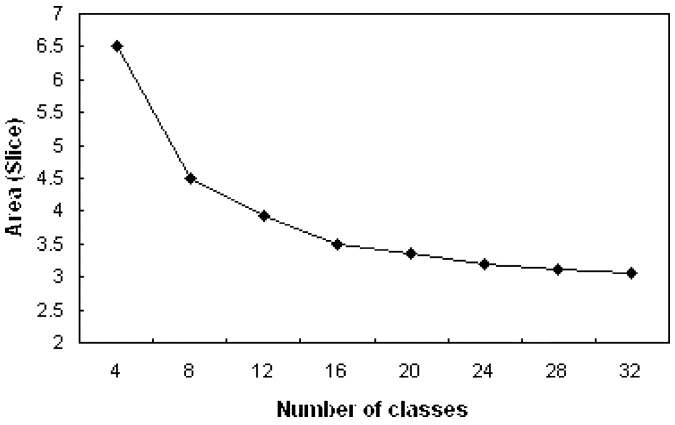

Fig. 19. Area of each DPE as a function of the class number.

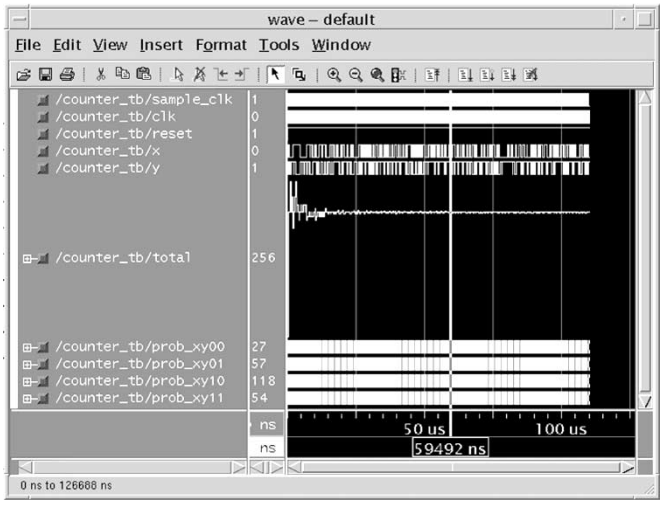

Fig. 20. Timing diagram from the initial condition.

the output of all the probabilities. The critical data path of the design extends from the inputs X and Y to the output of the last bit of each probability value. The largest portion of this delay occurs in the two additions. Their delay is about 232 nsec. The timing diagram of DPE including the initial condition is shown in Fig. 20. The obtained waveforms correspond to the simulation results.

V. Conclusion

This paper presents design and analysis of a DPE. The designed circuit uses a minimum hardware area (about three slices per class) to concurrently estimate class probabilities on an unlimited number of input data without division. The probabilities are estimated without division, which significantly reduces hardware requirements and the processing time. Class probabilities are estimated dynamically with 20 clock cycles per each update. This circuit is suitable for massively parallel computations required in on-line machine learning and in implementation of artificial neural networks.

References

[1] N. Izeboudjen, A. Farah, S. Titri, and H. Boumeridja, "Digital implementation of artificial neural networks: From VHDL description to FPGA implementation," in *Proc. Int. Work-Conf. Artificial and Natural Neural Networks, IWANN'99*, vol. 2, June 1999, pp. 139–148.

[2] J. Waldemark, M. Millberg, T. Lindblad, K. Valdemark, and V. Becanovic, "Implementation of a pulse coupled neural network in FPGA," *Int. J. Neural Syst.*, vol. 10, pp. 171–177, June 2000.

[3] J. G. Elredge and B. L. Hutchings, "RRANN: A hardware implementation of the backpropagation algorithm using reconfigurable FPGAS," in *Proc. IEEE Int. Conf. Neural Networks*, June 1994, pp. 77–80.

[4] S. Kumar, J. Ghosh, and M. M. Crawford, "Best-bases feature extraction algorithms for classification of hyperspectral data," *IEEE Trans. Geosci. Remote Sensing*, vol. 39, pp. 1368–1379, July 2001.

[5] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," *IEEE Trans. Syst., Man, Cybern. B*, vol. 28, pp. 417–425, June 1998.

[6] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Trans. Evol. Comput.*, vol. 4, pp. 380–387, Nov. 2000.

[7] M. Bichsel and P. Seitz, "Minimum class entropy: A maximum information approach to layered networks," *Neural Networks*, vol. 2, pp. 133–141, 1989.

[8] R. R. Coifman and M. V. Wickerhauser, "Entropy-based algorithms for best basis selection," *IEEE Trans. Inform. Theory*, vol. 38, pp. 713–718, Mar. 1992.

[9] J. Starzyk and Z. Zhu, "Software simulation of a self-organizing learning array system," in *Proc. 6th IASTED Int. Conf. Artificial Intelligence and Soft Comp.(ASC 2002)*, Banff, Alberta, Canada, July 17–19, 2002.

[10] F. Kanaya and K. Nakagawa, "Correspondence on the practical implication of mutual information for statistical decision making," *IEEE Trans. Inform. Theory*, vol. 37, pp. 1151–1156, July 1991.

[11] S. Kullback and R. A. Leibler, "On information and sufficiency ," in *The Annals of Mathematical Statistics*, 1951, vol. 22, pp. 79–86.

[12] M. Girolami, "Mercer kernel based clustering in feature space," *IEEE Trans. Neural Networks*, vol. 13, pp. 780–784, Apr. 2001.

[13] S. Fiori, "Hybrid independent component analysis by adaptive LUT activation function neurons," *Neural Networks*, vol. 15, no. 1, pp. 85–94, Jan. 2002.

[14] M. Girolami, "An alternative perspective on adaptive independent component analysis algorithms," *Neural Comput.*, vol. 10, no. 8, pp. 2103–2114, 1998.

[15] L. C. Parra, C. Spence, P. Sajda, A. Ziehe, and K.-R. Müller, "Unmixing hyperspectral data," *Advances Neural Info. Processing Syst.*, pp. 942–948, 2000.

[16] A. J. Bell and T. J. Sejnowski, "An information maximization approach to blind separation and blind deconvolution," *Neural Comput.*, vol. 7, no. 6, pp. 1129–1159, 1995.

[17] S. Amari, A. Cichocki, and H. H. Yang, "A new learning algorithm for blind signal separation," in *Advances in Neural Information Processing Systems 8*. Cambridge, MA: MIT Press, 1996.

[18] S. Fiori, "Nonsymmetric PDF estimation by artificial neurons: Application to statistical characterization of reinforced composites," *IEEE Trans. Neural Networks*, vol. 14, pp. 959–962, July 2003.

[19] M. K. Titsias and A. C. Likas, "Shared kernel models for class conditional density estimation," *IEEE Trans. Neural Networks*, vol. 12, pp. 987–997, Sept. 2001.

[20] R. Gaeda, J. Cerda, F. Ballester, and A. Mocholi, "Artificial neural network implementation on a single FPGA of a pipelined on-line backpropagation," in *Proc. IEEE 13th Int. Symp. System Synthesis*, 2000, pp. 225–230.

[21] "Logic Core Generator: Pipelined Divider V2.0," Xilinx Corporation, 2000.

[22] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," *J. Amer. Stat. Assoc.*, vol. 22, pp. 209–212, 1927.

[23] C. J. Clopper and E. S. Pearson, "The use of confidence or fiducial limits illustrated in the case of the binomial," *Biometrika*, vol. 26, pp. 404–13, 1934.

[24] P. Bickel and K. Doksum, *Mathematical Statistics*. Englewood Cliffs, NJ: Prentice-Hall, 1977.

[25] C. J. Stone, *A Course in Probability and Statistics* Duxbury, Belmont, 1995.

[26] D. R. Cox and E. J. Snell, *Analysis of Binary Data*, 2 ed. London, U.K.: Chapman and Hall, 1989.

[27] T. J. Santner and D. E. Duffy, *The Statistical Analysis of Discrete Data*. Berlin: Springer-Verlag, 1989.

[28] C. R. Rao, *Linear Statistical Inference and Its Applications*. New York: Wiley, 1973.

[29] A. Agresti and B. A. Coull, "Approximate is better than "exact" for interval estimation of binomial proportions," *The Amer. Statistician*, vol. 52, pp. 119–126, 1998.

[30] L. D. Brown, T. Cai, and A. DasGupta, "Interval estimation for a binomial proportion," *Statistical Science*, vol. 16, pp. 101–133, 2001.
[31] E-Handbook of Statistical Methods. [Online]. Available: http://www.itl.nist.gov/div898/handbook/prc/section2/prc241.htm
[32] S. Kumar, J. Ghosh, and M. M. Crawford, "Hierarchical fusion of multiple classifiers for hyperspectral data analysis," *Pattern Analysis Applicat.*, vol. 5, pp. 210–220, 2002.
[33] "Alliance Core: DFPDIV Floating Point Divider," Xilinx Corporation, 2001.
[34] "Alliance core: DFPADD Floating Point Adder," Xilinx Corporation, 2001.

Janusz A. Starzyk (SM'83) received the M.S. degree in applied mathematics and the Ph.D. degree in electrical engineering from Warsaw University of Technology, Warsaw, Poland, in 1971 and 1976, respectively.

From 1977 to 1981, he was an Assistant Professor at the Institute of Electronics Fundamentals, Warsaw University of Technology, Warsaw, Poland. From 1981 to 1983, he was a Postdoctorate Fellow and Research Engineer at McMaster University, Hamilton, Canada. In 1983, he joined the Department of Electrical and Computer Engineering, Ohio University, Athens, where he is currently a Professor of electrical engineering and computer science. He has cooperated with the National Institute of Standards and Technology in the area of testing and mixed signal fault diagnosis. He has been a consultant to AT&T Bell Laboratories, Sarnoff Research, Sverdrup Technology, Magnolia Broadband, and Magnetek Corporation. His current research is in the areas of self-organizing learning machines, neural networks, rough sets, VLSI design and test of mixed signal CMOS circuits, and reconfigurable design for wireless communication.

Feng Wang received the B.S. degree in electronics engineering from Fudan University, Shanghai, China, in 1997. He is currently working toward the M.S. degree at Ohio University, Athens.

From 1997 to 2002, he was a Design Engineer with Texas Instruments joint venture, and Intel China Software Laboratory, China, where he was engaged in system design. His research interests include reconfigurable design, machine learning and VLSI design.