as *weighted contribution TSPA*.

The modification of the original algorithm is mainly based on the consideration that the equal contribution to the global connection error may not be the best choice, although that is a more accurate estimate of the theory of gradient descent. In addition, in the modified algorithm, the improvement for faster convergence rate can also be observed in the experiments even without the effect of the weighted contribution. However, such improvement cannot be guaranteed in all the cases.

Table 1 shows the comparison among the convergence rates of three variations of TSPA: the original one, the modified one with *equal contribution*, and the modified one with *weighted contribution*. As mentioned above, a better convergence rate in the second algorithm can be observed from Table 1 but it cannot be guaranteed in all cases (i.e., when $N_{ptn} = 160$). Yet the third algorithm is expermentally shown to have the best convergence rate. Since the training times per iteration of three algorithm are the same, the third approach needs much less total training time than the original one.

| $N_{ptn}$ | iterations | | |
|---|---|---|---|
| | original | equal contribution $a_i = 1$ | weighted contribution $a_i = N_i^2 + 2N_i$ |
| 32 | 9685 | 854 | 245 |
| 64 | 119163 | 1690 | 293 |
| 96 | 266976 | 1493 | 568 |
| 100 | 102802 | 2555 | 579 |
| 128 | 8699 | 4286 | 566 |
| 160 | 3236 | 4395 | 606 |
| 192 | 3674 | 3006 | 472 |
| 320 | 21456 | 8170 | 1769 |
| 480 | 113625 | 38875 | 12335 |

Table 1: Convergence iterations for different variations of TSPA.

# 4 Conclusions

Training set parallel algorithm (TSPA) for a neural network based automated Fingerprint Image Comparison (FIC) system is investigated in this paper. The target architecture is assumed to be a coarse-grain distributed memory parallel architecture. This type of parallelism has evenly distributed computation load and relatively small communication overhead. We obtained almost linear speedup up to 31.2 using 32 processors concurrently measured on a 32-node CM-5.

For the purpose of reducing the risk of slower convergence rate, a modified TSPA using weighted contributions of connections is proposed. Our experimental results show that a fast convergence rate can be achieved by using this modified algorithm without any loss of the high speedup performance. In this case, a significantly faster total training time can be obtained.

We trained the FIC system with 480 pairs of images by using the three variations of TSPA described in Section 3.1. The fingerprint training set are obtained from a CD-ROM distributed by NIST. The total training time and the observed accuracy for each variation are listed in Table 2[2]. Although the observed accuracy of the weighted contribution TSP algorithm is slightly lower than the other two TSP algorithm, it has considerably reduced the total training time.

| TSPAs | original | equal contribution | weighted contribution |
|---|---|---|---|
| total training times (hours) | 49.3 | 15.5 | 5.0 |
| recognition accuracy | 96.09% | 96.20% | 95.49% |

Table 2: Total training times and recognition accuracies of the three variations of TSPA.

# References


[1] Z. Miao, "Parallel Algorithms for Automated Fingerprint Image Comparison System", Thesis, Department of Electrical and Computer Engineering, West Virginia University, 1995.

[2] P. Baldi and Y. Chauvin, "Neural Networks for Fingerprint Recognition", in *Neural Computation*, Vol.5, pp.402-418, 1993.

[3] H. Ammar, S. Zeng and Z. Miao, "Parallel Processing and Fingerprint Image Comparison", To Appear, *International Journal of Modeling and Simulation*, 1996.


[2]The recognition accuracy listed in this table is just for the comparison of accuracy among three TSPAs. Because the unmatched image pairs and matched image pairs are not evenly distributed in the test sample set used, the actual recognition accuracy could be much lower if a different test sample set is used.

epoch/block, which is much less frequent than that of once per pattern in node parallelism. However, this implicates lower weight update frequency.

For the case of $B_{ptn} = 32$ in our experiment,

$$R_{tsp} = 6.94 \times 10^{-4} P$$

Obviously $S_{tsp} \approx P$ even when $P$ is very large.

## 3.1 Weighted Contribution Training Set Parallel Algorithm

The speedup of the TSP is much better than other types of parallelism. However, it suffers from the risk of slower convergence rate. This is due to the fact that block/batch mode which results in infrequent weight update is used in this parallelism. For the purpose of overcoming this drawback, the traditional TSPA is modified as described in this section. The experiments prove that the modified algorithm significantly reduces the number of iterations for convergence, meanwhile, it has a speedup as high as that of the original TSPA. Thus the total training time can be largely reduced in this case.

The description of the modified algorithm is as follows:

**S 1:** Each $P_i$ trains its local copy of the weight matrix with the assigned training subset using *pattern mode*.

**S 2:** After all the patterns in the subset are presented, the local weight matrix $W_i$ is collected from all the processors and perform the following weighted summation using global reduction function:

$$W = \frac{1}{P} \sum_{i=1}^{P} c_i W_i \tag{6}$$

where $c_i$ is a coefficient attached to each $W_i$. It serves as as a contribution factor to the global summation.

**S 3:** $W$ is broadcast back to all the processors and replaces the local weight matrix in each processor.

The local weight matrices on all processors are initialized as same, so that they are kept consistent after each epoch.

The contribution coefficient $c_i$ is calculated from the following functions.

$$a_i = {N_i}^2 + 2N_i \tag{7}$$

$$c_i = \frac{a_i}{\sum_{i=1}^{P} a_i} \tag{8}$$

where $N_i$ is the number of *"unlearned"* patterns in the subset whose decision outputs are beyond error tolerance of the network in processor $P_i$. Therefore $c_i$ means the contribution of the local connections to the global connection is proportional to the number of unlearned patterns in its subset. Since $N_i$ of each subset changes during the whole training procedure, $c_i$ also changes dynamically.

The coefficient $c_i$ can be viewed as strengthening the effect of connection errors from the subsets which have larger number of unlearned patterns. After weighted by $c_i$, the connection errors from the subsets which have larger number of unlearned patterns dominate the convergence direction. This is because generally, the convergence behavior of the subset with larger number of unlearned patterns is closer to the real convergence behavior of the whole training set. In this case, the subset with larger number of unlearned patterns is "more important" than those with smaller number of unlearned patterns. The emphasis on the "important" subsets is shown to be an efficient accelerator for faster convergence rate.

The derivation of $a_i$ in Equation 7 is based on the experimental results. The experimental results show that if the connection errors is weighted by this function, a faster convergence rate can be achieved than some other expressions tested, such as $a_i = N_i$ or $a_i = {N_i}^2$. More accurate expressions for $c_i$ still need to be investigated analytically.

There are two distinctions between the original TSPA and the modified TSPA:

1. The modified algorithm tries to simulate the behavior of pattern mode weight update strategy. Each local copy of the network is trained by its training subset and updates its connections (weights) using pattern mode. At the end of an epoch, each local copy submits its *connections* to the *global connections*. Instead, the original TSPA which uses gradient descent theory does not update the connections during each epoch if batch mode is being used. Each local copy submits *connection errors* to the *global connection errors*. However, the pattern mode used in the modified TSPA is only limited to its training subset.

2. In the original TSPA, connection errors from the local copies of the network are evenly contributed to the global connection error. While in the modified TSPA, the number of unlearned patterns in the training subset of each processor is considered as a weight factor for the connection contribution. Therefore, the modified TSPA is termed

field and moves across all the image by 5 pixels each step.The two neighboring receptive fields have an overlap of 2 pixels to approximate a continuous convolution operation. Thus each $32 \times 32$ compressed central region is transformed into three $6 \times 6$ output arrays, one for each filter. The output of filter $j$ at position $(x, y)$ in one of these arrays is given by (for instance for $A$):

$$z^j_{x,y}(A) = f(\sum w^j_{x-r,y-s} I_{r,s}(A) + t^j_{r,s}) \qquad (1)$$

where $I_{r,s}(A)$ is the pixel value in the compressed central region of image $A$ at the position of $(r, s)$, $f$ is the sigmod function

$$f(x) = \frac{1}{1 + e^{-x}},$$

$w_{x-r,y-s}$ is the weight of synaptic connection from the position of $(r, s)$ in the compressed central region to the position of $(x, y)$ in the array of filter outputs, and $t^j$ is a threshold. The summation in Equation 1 is taken over the $7 \times 7$ patch coincident with the receptive field of the filter at the $(x, y)$ location. The threshold and the weights of $7 \times 7$ receptive field are characteristic of the filter, so that they can also be viewed as the parameters of a translation invariant convolution kernel. They are shared within an image but also across the images $A$ and $B$. These $7 \times 7 + 1 = 50$ parameters are adjustable during network training. Denote $i$ as the position of $(x, y)$. For the outputs of each filter $j$, the squared difference between two images is

$$\Delta z^j_i(A, B) = [z^j_i(A) - z^j_i(B)]^2. \qquad (2)$$

Denote $\Delta z = \Delta z(A, B)$ as the array of all $\Delta z^j_i(A, B)$ for all position $i$ and filter $j$.

The purpose of the decision part of the network is to estimate the probability $p = p[M(A, B)/\Delta z(A, B)] = p(M/\Delta z)$ of a match between images $A$ and $B$, given the evidence $\Delta z$ provided by the convolution filters. The decision part can be viewed as a binary Bayesian classifier.

The detailed description of the FIC system architecture can be refered in [3].

# 3 Training Set Parallel Algorithms for the FIC System

Training set parallelism (TSP) is a coarse computation granularity approach. It exploits parallelism at higher level of abstraction. By contrast, the node parallelism which requires network decomposition is termed as low level partitioning scheme. In TSP, the neural network is duplicated on each processor. A subset of the training set is allocated to each processor. Training by batch/block is required as the weight update strategy. The block length (number of patterns in each block) should be larger than the number of available processors to be able to fully utilize all the processor resources.

The TSP runs many copies of the FIC system concurrently by the distribution of the training set across processors. Batch mode is used as the weight update strategy in this diagram. Thus the weight is updated after the presentation of whole training subset in each processor. In the case of block mode being used, the weight is updated after the presentation of $\frac{B_{ptn}}{P}$ patterns in each subset, where $B_{ptn}$ is the number of patterns per block, and $P$ is the number of processors being used. Therefore, the number of message passings is $\frac{N_{ptn}}{B_{ptn}}$ times per epoch.

The training time for an epoch (using block mode) is given by

$$T^{tsp}_{epoch} = \frac{N_{ptn}}{P} T^{ser}_{ptn} + \frac{N_{ptn}}{B_{ptn}} t^{tsp}_{com} \qquad (3)$$

where $t^{tsp}_{com}$ is the communication time for one epoch. It is measured as 2.094ms per epoch on the CM-5 implementations.

The speedup of a parallel algorithm is defined as

$$S = \frac{T^{ser}_{epoch}}{T^{par}_{epoch}}$$

where $T^{ser}_{epoch}$ is the sequential training time per epoch, and $T^{par}_{epoch}$ is the parallel training time per epoch. Then the the speedup of TSPA (training set parallel algorithm) is

$$\begin{aligned} S_{tsp} &= \frac{T^{ser}_{epoch}}{T^{tsp}_{epoch}} \\ &= \frac{P}{1 + R_{tsp}} \end{aligned} \qquad (4)$$

where

$$R_{tsp} = \frac{t^{tsp}_{com}}{B_{ptn} T^{ser}_{ptn}} P \qquad (5)$$

is the *communication-computation ratio* of TSPA.

Ideally $t^{tsp}_{com}$ remains constant when $P$ increases[1]. It is rather small by comparison with $T^{ser}_{ptn}$. Importantly, the message passing requirement is once per

[1] Since global communication is efficiently implemented using the fat tree network in the CM-5, $t^{tsp}_{com}$ remains almost constant when $P$ is increased. In general, however, when using a general distributed memory architecture, the global communication time might significantly reduce the speedup when $P$ is large.

# Implementation of a Training Set Parallel Algorithm for an Automated Fingerprint Image Comparison System

**Hany H. Ammar and Zhouhui Miao**
Department of Electrical and Computer Engineering
West Virginia University, Morgantown, WV 26506, USA

## Abstract

*This paper addresses the problem of developing an efficient training set parallel algorithm (TSPA) for the training procedure of a neural network based Fingerprint Image Comparison (FIC) system. The target architecture is assumed to be a coarse-grain distributed memory parallel architecture. Theoretical analysis and experimental results show that TSPA achieves almost linear speedup performance. This parallel algorithm is applicable to ANN training in general and is not dependent on the ANN architecture. However, TSP is amenable to a slow convergence rate. In order to reduce this effect, a modified TSPA using weighted contributions of synaptic connections is proposed. Experimental results show that this algorithm provides a fast convergence rate, while keeping the high speedup performance obtained.*

*The above algorithms are implemented and tested on a 32-node CM-5.*

## 1 Introduction

The goal of this paper is to develop more efficient parallel algorithm based on training set parallelism (TSP). To reduce the risk of slower convergence rate of TSP, a modified training set parallel algorithm is investigated.

The paper is organized into four sections. Section 2 describes a neural network based FIC system architecture. Section 3 presents the implementations of traditional and modified training set parallel algorithms (TSPAs) for the FIC system. Testing results are then presented. Section 4 summarizes the conclusions of this paper.

## 2 The FIC System Architecture

The neural network based Fingerprint Image Comparison (FIC) system presented in this section is mainly based on an algorithm described by Baldi and Chauvin in [2] and modified in [3]. The system consists of two components: a preprocessor and a neural network based decision stage. When presented with a pair of fingerprint images, the system outputs an estimate of the probability that the two images originate from the same finger. The fingerprints are obtained from the NIST (National Institute of Standards and Technology) Fingerprint Database CD-ROM which contains 4000 (2000 pairs, two different rollings of the same finger) 8-bit grey scale images of randomly selected fingerprints. Each print is $512 \times 512$ pixels with 32 rows of white space at the bottom of the print.

The purpose of the preprocessing stage is to extract a central region from each one of the two input images, and to align the two central regions. The outputs of the preprocessor are two compressed central regions with $32 \times 32$ pixels each. The detailed algorithm of preprocessor is described in [3].

The decision stage is the neural network part of the algorithm. The network has a pyramidal architecture, with two aligned and compressed central regions as inputs along with a single output $p$. The bottom level of the pyramid corresponds to a convolution of the compressed central regions with respect to a set of filters or feature detectors. The decision layer implements results from a probabilistic Bayesian model for the estimation of $p$, based on the output of the convolution filters. Both the filtering and decision part of the network are adaptable and trained simultaneously.

The two central regions are first convolved with a set of adjustable filters. Three different filters are used in our application. Each filter has a $7 \times 7$ receptive