論文97-34C-1-4

Connection Machine CM-2상에서 신경망군(群)의 병렬 구현

(Parallel Implementation of A Neural Network Ensemble on the Connection Machine CM-2)

金 大 鎭 *

(Daijin Kim)

요 약

본 논문은 Connection Machine CM-2상에서 물체 인식을 위해 개발된 신경망군(群)의 병렬 구현을 다룬다. 구현 과정은 여러 개의 신경망이 서로 다른 초기 가중치로부터 출발하여 동시에 구현되며, 모든 학습 샘플들도 각 신경망에 한꺼번에 가해진다. 따라서, 본 구현을 단일 신경망 구현과 비교해보면, 최대 $O(N \cdot m \cdot n)$의 (여기서, N, m과 n은 각각 신경망수, 학습 샘플수, 그리고 신경망의 노드수를 나타낸다.) 연산 속도 개선 효과를 얻을 수 있다. 제안한 신경망群을 대표적 SIMD 병렬 컴퓨터인 CM-2상에 2차원 물체 인식 문제에 응용하여 연산 속도 향상과 수정된 역전파 학습에 의한 수렴 특성을 확인하였다.

Abstract

This paper describes a parallel implementation of a neural network ensemble developed for object recognition on the Connection Machine CM-2. The implementation ensures that multiple networks are implemented simultaneously starting from different initial weights and all training samples are applied to each network by one sample per a copy of each network. When compared with a sequential implementation, this accelerates the computation speed by $O(N \cdot m \cdot n)$, where N, m, and n are the number of networks in the ensemble, the number of training samples used, and the size of each network, respectively. The speedup in the computation time and the convergence characteristics of the modified backpropagation learning procedure were evaluated by two-dimensional object recognition problem.

I. 서 론

크기, 이동 및 회전에 불변인 물체 인식을 수행하는 신경망群을 이전 논문[1]에서 제안하였다. 이에 따르면 한 신경망群이 16개의 동일한 신경망으로 구성되며, 각 신경망은 4계층의 전향 역전파 구조를 가진다. 그리고, 불변 인식이 한 물체의 여러 크기, 이동, 및 회전의 조합들을 학습시켜 얻어지는 것이 아니고, 신경망 자체가 가진 구조적 특성에 의해 얻어진다. 즉, 90°의 배수($90° \cdot n$, $n=0,1,2,3$)의 회전 불변성은 신경망내 첫 번째 계층과 두 번째 계층 사이의 특수한 결합 방식에 의해서 얻어지고 크기 및 이동 불변성은 서로 다른 해상도와 인접하는 거리를 달리하는 여러 개의 column들을 중첩시킴으로서 얻어진다. 90°의 배수($90° \cdot n$, $n=0,1,2,3$)이외의 각도에 대한 회전 불변성은 학습에 의해 얻어지므로 회전에 대한 인식 능력은 0°와 90°사이에 몇 개의 회전한 물체를 학습시키느냐에 의해 결정되며, 크기 및 이동에 대한 인식 능력의 정확성은 신경망내 사용된 column의 수에 의해서 좌우된다.

일반적으로 크기 및 이동에 대한 인식 능력을 향상

* 正會員, 東亞大學校 컴퓨터工學科

(Dept. of Computer Engineering, DongA University)

※ 이 논문은 1996학년도 동아대학교 학술연구 조성비(일반과제)에 의하여 연구되었음.

接受日字:1996年8月6日, 수정완료일:1997年1月18日

시키는데는 다음 2가지 방법이 있다. (1) 아주 미세한 해상도를 가지며 서로 좁은 간격을 갖는 많은 column을 사용하는 것과 (2) 여러 개의 신경망으로 구성된 신경망群[2]을 사용하는 것이다. 본 논문에서는 첫 번째 방법이 갖는 폭발적인 복잡성 증가 때문에 두 번째 방법을 채택하였다. 나아가 두 번째 방법은 신경망群이 갖는 consensus에 바탕한 집합적 결정에 의해 단일 신경망보다 오인식할 가능성이 줄어든다는 이점이 있다.

신경망群을 사용하는 경우의 한가지 단점은 서로 다른 일반화 특성을 갖는 여러 개의 신경망을 가지는데 따른 학습 시간의 과도한 증가이다. 이를 극복하는 한 방법이 신경망群을 병렬 컴퓨터 상에서 시뮬레이션 하는 것이다. 본 논문은 신경망群을 Connection Machine CM-2[3] 상에서 병렬 구현하는 것을 다루고 있다. CM-2는 SIMD 병렬 컴퓨터의 일종으로 65,536개의 프로세서를 갖는다. 각 프로세서는 단위 비트 연산을 수행하며 64K 비트의 국부 메모리를 갖고, 12차원 하이퍼 큐브 형태로 연결된다. 각 프로세서는 아주 유연한 연결 구조를 통해 통신이 가능하므로 시스템 구조를 응용하고자 하는 문제의 구조에 쉽게 매칭시킬수 있다. CM-2의 성공적 응용 사례[4-7]는 아주 다양한 분야에 걸쳐서 발견된다.

신경망群을 시뮬레이션 하는데 설정한 요구사항은 (1) 프로세서 사용율의 극대화, (2) 프로세서당 국부 메모리의 최소 할당, (3) 프로세서간 상호 통신의 최소화[8-10]이다. 첫 번째 요구 사항은 신경망群내 모든 신경망을 동시에 구현하고 모든 학습 샘플을 동시에 처리함으로서 가능해진다. 두 번째 요구 사항은 하나의 32비트 부동 소수점 실수를 32개의 물리적으로 인접한 프로세서들에 걸쳐 한 프로세서당 한 비트씩 저장하는 공유 메모리 기법에 의해 만족된다. 따라서, 32개의 프로세서가 하나의 32비트 가중치를 공유하면서 각 프로세서는 이 가중치의 한 비트만을 저장하게 된다. 세 번째 요구 사항은 한 유닛에 들어오는 (또는 나오는) 모든 가중치를 한 프로세서에 저장함으로서 만족된다.

Zhang 등은[10] CM-2상에서 역전파 신경망 알고리즘의 효율적인 시뮬레이션을 보였는데 이들은 프로세서간 통신을 NEWS 그리드[12]를 통한 2차원 최근접 이웃(nearest neighbor) 통신을 이용하였다. 2-D 그리드의 사용은 한 프로세서가 다른 프로세서와 통신을 할 때 그 프로세서의 어드레스를 계산하거나 검사할 필요가 없기 때문에 신경망의 구현 시간을 단축 시킬 수 있다. 이들이 구현한 신경망은 64K개의 프로세서를 사용하는 경우 최대 180 MIPS(초당 백만개 단위의 곱셈 및 덧셈의 수, million interconnection per second)의 연산 성능을 보였다. 그러나, 2차원 그리드의 사용은 각 층의 유닛 수가 2의 배수일 때만 가능하다는 점에서 여러 형태의 신경망 구현을 어렵게 만든다.

본 연구에서는 이를 극복하기 위해 연산 속도를 약간 희생하더라도 여러 가지 신경망의 구현을 가능하게 하기 위해 프로세서간 통신을 NEWS 그리드[12]를 통한 2차원 최근접 이웃(nearest neighbor) 통신대신에 router에 의해 제공되는 일반적인 통신을 채택하였다. 나아가 본 연구의 주제인 신경망群의 구현이 그들이 구현했던 신경망 그룹의 단순한 확장이 아님은 다음 두 가지 사실로부터 알 수 있다. 첫째, CM-2상에 한 신경망群의 구현은 CM-2의 가상율(virtual ratio)이나 한꺼번에 구현되는 신경망 그룹의 수 등과 같은 여러 요인에 의해 영향을 받으며, 둘째, 신경망 그룹간의 적당한 경계 방안이 요구된다는 점이다.

본 논문의 구성은 다음과 같다. 2장에서는 불변 물체 인식을 위한 신경망群의 구조를 간단히 설명한다. 3장에서는 CM-2상에 신경망群의 구현을 자세히 설명한다. 4장에서는 구현한 신경망群을 2차원 물체 인식 문제에 적용하여 실험 결과를 학습 시간 및 학습 특성 면에서 논의한다. 마지막으로 결론이 뒤따른다.

Ⅱ. 불변 물체 인식을 위한 신경망群

크기, 이동, 및 회전에 불변인 물체 인식을 위한 신경망群은 Kim과 Suk[1]에 자세히 설명되어 있다. 신경망群은 불변 물체 인식을 위해 특별하게 설계된 여러 개의 동일한 신경망들로 구성되어 있다. 실제, 사용되는 column 수의 제약과 하드웨어적 연결에 의해 얻어지는 제한된 회전 불변 능력은 하나의 신경망에 의한 불변 인식 능력에 한계를 나타낸다. 이러한 제약을 극복하고자 신경망群을 제안하였는데 이는 보다 나은 크기, 이동, 및 회전 불변성과 더 나은 잡음 면역성(immunity)을 보인다.

1. 크기, 이동, 및 회전에 불변인 신경망 구조

신경망群을 구성하는 각 신경망의 기본 구조는 그림

1과 같은 4층의 column 구조를 갖는다. 여기서 column의 의미는 Hubel[13] 등에 의해서 제안된 시각 피질내의 선택적 셀을 모방한 hypercolumn의 의미와 동일하다. 한 column의 4층은 각각 입력층, rf층(receptive field layer), 은닉층, 및 출력층이라고 부른다. 입력층의 각 유닛은 시각 영역내 유한 면적을 포함하는 국부 rf를 가지며 rf 영역내 모든 입력값(stimuli)을 받아들인다. 어떤 기하학적 제약 조건(여기서는 동일 원주 상에 서로 90° 각도를 갖는)을 만족하는 네 개의 입력 유닛들은 하나의 rf 유닛을 구성한다. rf층과 은닉층, 은닉층과 출력층간의 유닛들은 완전 결합된(fully connected) 형태를 갖는다.

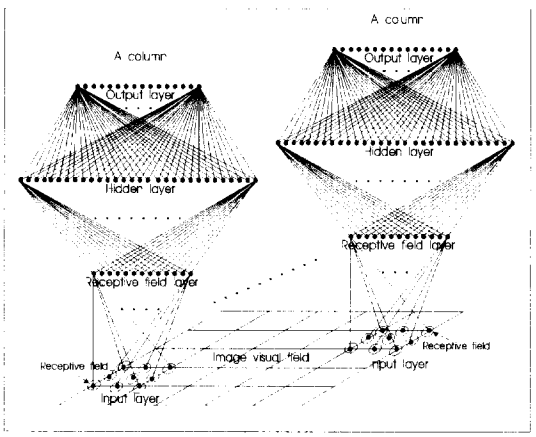

그림 1. column의 기하학적 구조
Fig. 1. A geometrical structure of column.

입력층과 rf층 사이의 특별한 연결 구조는 90° 각도의 회전 불변성을 얻는다. 크기 불변성은 시각 필드상의 정해진 위치마다 동일한 가중치를 갖지만 공간 분해능이 서로 다른 여러 개의 column들을 쌓아둠으로서 얻어진다. 이들중 입력 물체의 크기와 가장 잘 맞는 한 column이 가장 좋은 반응을 나타낼 것으로 기대된다. 이동 불변은 같은 공간 분해능을 갖는 여러 개의 column들을 일정 간격만큼 떨어지면서 중복하여 놓음으로서 얻어진다. 이들중 입력 물체의 위치와 가장 가깝게 놓인 column이 가장 좋은 반응을 나타낼 것으로 기대된다.

2. 신경망群

위에서 언급한 구조를 갖는 하나의 신경망에 의해 얻어지는 불변성은 실제 크기 및 이동 불변성의 경우 사용하는 column 수의 제약과 회전 불변성의 경우 90°의 배수만이 가능한 제약 때문에 한계를 지니고 있다. 따라서, 하나의 신경망으로는 한 물체의 크기, 이동, 회전 정도가 심할 때는 인식 능력의 저하가 매우 크다. 이러한 불변 인식 능력을 제고하기 위해 본 논문에서는 동일한 구조를 갖지만 각기 다른 인식 능력을 갖는 여러 개의 신경망으로 구성된 신경망群[2]을 사용한다. 그림 2는 신경망群을 사용함으로서 신경망에 의한 일반화 특성이 개선되는 예를 보인다. 이 그림에서, U는 가능한 모든 입·출력쌍 집합, T는 학습 샘플 집합을 의미한다. k개의 신경망에 학습 샘플 집합 T를 독립적으로 사용하여 각기 다른 가중치 벡터 $\omega_1, \omega_2, \cdots, \omega_k$를 생성한다고 하자. 이들 가중치 벡터는 입력 공간 $G_1, G_2, \cdots, G_k$에 대해 서로 다른 일반화 특성을 나타낸다. 그림 2에서 보듯이 여러 개의 신경망으로 구성된 신경망群에 의해 담당되어지는(covered) 입력 공간의 면적은 신경망의 수가 증가할수록 점차 확장됨을 알 수 있다. 따라서, 한 신경망에 의해 한 입력이 오인식되더라도 다른 신경망에 의해 올바르게 인식할 수도 있으므로 전체적으로 인식률을 개선하는 효과를 얻을 수 있다.

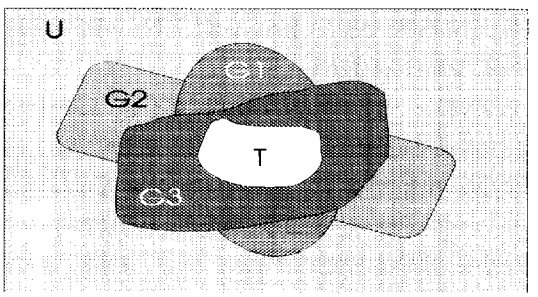

그림 2. 신경망群에 의한 일반화 특성 향상
Fig. 2. Improvement of generalization performance by a NN ensemble.

독립적으로 학습된 신경망에 의해 얻어진 결정들로부터 최종 결정에 도달하는 방법에는 여러 가지가 있을 수 있다. 가장 흔히 쓰이는 결정 규칙들로는 다른 어느 것보다 더 많은 신경망에 의해 결정된 것을 선택하는 plurality-voting 규칙과 과반수 이상의 신경망에 의해 결정된 것을 택하는 majority-voting 규칙이 있다. 만약 이러한 결정에 실패하면 그 결정 결과는 오인식(誤認識)으로 생각한다. 본 논문에서는 그림 3과 같이 두 가지 결정 규칙을 연속적으로 사용하였다. 먼저, majority voting 규칙을 사용하여 결정하며 이것

이 한 결정을 얻는데 실패하면 plurality-voting 규칙을 사용한다.

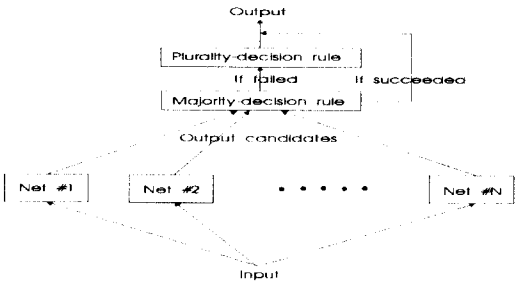

그림 3. 신경망群에 적용된 결정 규칙
Fig. 3. The decision strategy of a NN ensemble.

Ⅲ. CM-2상에 신경망群의 병렬 구현

본 논문에서 제안한 병렬 구현은 다음 세 단계의 계층적 구성으로 되어 있다. 상위 단계(level 3)인 신경망群은 N개의 신경망 그룹으로 되어 있고, 다음 단계(level 2)인 신경망그룹은 n개의 동일한 신경망들로 구성되어 있으며, 하위 단계(level 1)인 각 신경망은 m개의 동일한 column으로 구성되어 있다. 구현시, 하나의 column은 기본 구성 요소로 CM-2상의 한 프로세서에 대응한다. 한 프로세서는 한 층당 하나씩 네 개의 유닛과 이들 사이의 연결하는 가중치 시냅스들을 포함한다. 그림 4는 세 단계간의 관계를 도식적으로 나타낸 것이다.

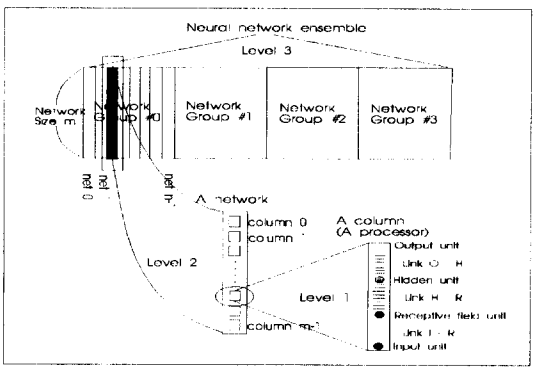

그림 4. 신경망群의 계층적 구조
Fig. 4. Layered organization of a NN ensemble.

여기서 한가지 고려해야 할 사항은 가중치를 어떻게 갱신하는가 하는 문제로서 크게 두 가지 갱신 방법 (온라인 모드와 배치 모드)이 있다. 온라인 갱신 방법은 가중치가 매 학습 샘플이 가해진 후 갱신되며, 배치 모드는 모든 학습 샘플이 가해진 후 누적된 gradient에 의해 한 번 갱신된다. Becker와 Le Cun[17]은 배치 모드가 온라인 모드보다 요구되는 학습 반복 횟수가 적으며 모멘트 항의 부가에 의한 수렴 속도의 개선 효과가 배치 모드에서 보다 현저하게 나타남을 실험적으로 보였다. 나아가, 한꺼번에 모든 학습 샘플을 동시에 받아들이며 가중치가 PARIS[12]내 *scan_add* 및 *scan_copy* 함수와 같은 병렬 prefix 연산에 의해 쉽게 갱신 가능하므로 CM-2상의 구현에서는 배치 모드가 더욱 적당하다.

1. 신경망 수준 구현 (Level 1)

사용한 신경망은 4계층의 전향 역전파 구조를 가지는 데 두 번째 층과 세 번째 층, 세 번째 층과 네 번째 층은 완전 결합 형태이며, 첫 번째 층과 두 번째 층은 두 번째 층의 한 유닛이 첫 번째 층의 유한 개의 유닛과 결합한 국부적 결합 형태를 갖는다. 그림 5는 각 층별로 52, 13, 32, 및 16개의 유닛을 갖는 한 신경망의 내부 배치를 나타낸 것이다. CM-2의 한 프로세서는 한 column과 관련된 모든 정보를 저장한다. 예를 들면, 프로세서 j는 i번째 층의 j번째 모든 유닛 $N_{i,j}$ $(i=0,1,2,3)$과 i번째 층의 모든 유닛들로부터 $(i+1)$번째 층 j번째 유닛 $N_{i+1,j}$으로의 모든 시냅스 $\boldsymbol{W}_{i,k}^{i+1,j}$ $(i=0,1,2$이며 $k=0,1,\cdots,n_i$, 여기서 n_i는 I번째 층의 유닛수)을 저장하고 있다.

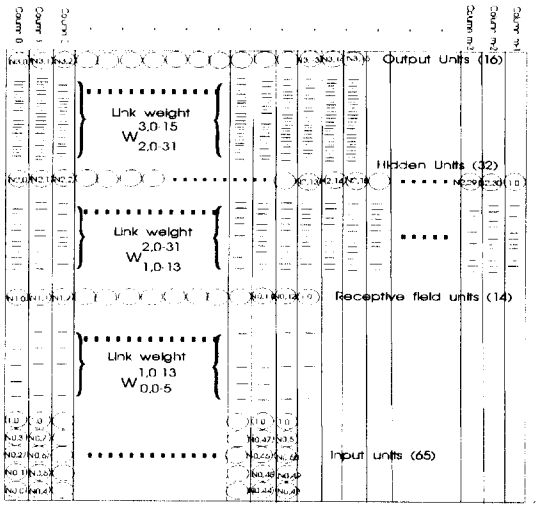

그림 5. 한 신경망의 내부 배치
Fig. 5. Internal layout of a neural network.

역전파 학습 알고리즘은 (1) 출력을 계산하기 위한 전향 연산, (2) 오차의 역전파, (3) 오차에 기반한 가중치 갱신의 세 단계로 이루어져 있다. 연산 시간은 일반적으로 단위 시간당 가중치 갱신 횟수(WUPS, weight updates per second)[11]에 의해서 비교된다. 한 신경망 그룹내 모든 신경망에 의해서 수행되는 연산은 각 신경망에 가해지는 입출력이 다른 것을 제외하고는 동일하다. 다음 설명은 한 신경망 내에서 일어나는 각 연산을 자세히 설명한 것이다.

1) 단계 1 : 전향 연산

본 단계에서는 각 유닛의 출력을 아래 식과 같이 신경망의 하위 층으로부터 활성화 함수를 전향적으로 연산함으로서 얻는다.

$$o_{i+1,j} = sigmoid\left(\sum_{k=0}^{n_i} w_{i,k}^{i+1,j} \times o_{i,k}\right) \text{for } i = 1,2,3. \quad (1)$$

지금부터, i번째 층의 j번째 유닛을 (i,j)로 나타내기로 한다면, $o_{i+1,j}$는 유닛 $(i+1,j)$의 출력, $w_{i,k}^{i+1,j}$는 유닛 (i,k)로부터 유닛 $(i+1,j)$로의 가중치, n_i는 i번째 층의 유닛수를 나타낸다. 실제 구현시는, rf층(i=1)에서는 회전 불변성을 얻기 위해 네 개의 내부 상태의 sigmoid 출력의 합과 내부 상태를 회전시키는 부가적인 연산이 요구된다.

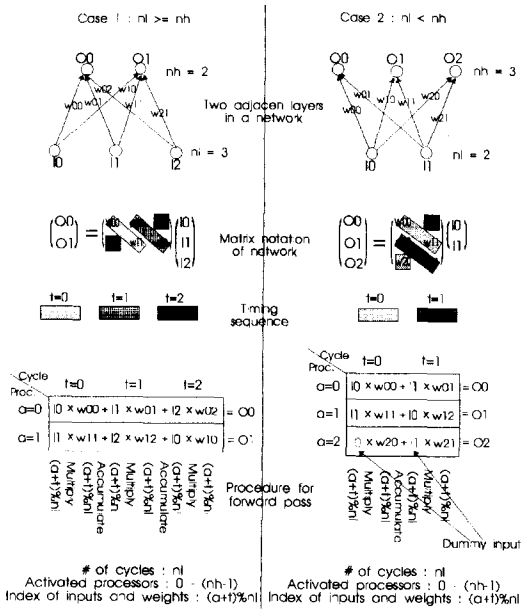

그림 6. 한 신경망에서 전향 연산의 예

Fig. 6. An illustration of computing the forward pass.

다음은 두 가지 대표적 전향 연산의 예($n_l \geq n_h$과 $n_l < n_h$의 경우, 여기서 n_l과 n_h는 각각 l번째와 $h(=l+1)$번째 층의 유닛수를 나타낸다.)를 보인 것이다. 그림 6은 (1) $n_l=3$ 과 $n_h=2$의 경우, (2) $n_l=2$과 $n_h=3$의 경우를 나타낸 것이다. 다른 모든 인접층 사이에서도 비슷한 연산이 일어난다. 전향 연산의 구현시 특이한 점들을 기술하면 다음과 같다.

(1) 프로세서중 프로세서 번지가 n_h-1보다 작은 프로세서만 전향 연산시 사용된다.

(2) $n_l < n_h$의 경우, (n_h-n_l)개의 dummy 입력이 처음 n_l개의 입력에 반복적으로 연이어 삽입한다. 예를 들어 $n_l=3$, $n_h=7$인 경우, n_h개의 활성 프로세서가 가지는 초기입력은 $i_0, i_1, i_2, i_0, i_1, i_2$ 그리고 i_0이다.

(3) 한 유닛은 바로 아래층의 모든 유닛과 n_l개의 가중치에 의해 연결된다. h번째 층의 한 유닛의 출력은 l번째 층의 모든 유닛들의 출력의 가중치 합이다. 따라서, h번째 층의 한 유닛의 출력은 매 사이클마다 l번째 층으로부터 한 입력씩 받아 연산하므로 한 출력을 계산하는데 n_l 사이클이 걸린다.

(4) h번째 층의 각 유닛의 출력은 l번째 층에서 오는 입력에 가중치를 곱하고 이들을 누적함으로서 얻어진다. t번째 사이클에서 h번째 층의 a번째 유닛은 한 부분합 $i_{l,(t+a)\%n_l} \times w_{l,(t+a)\%n_l}^{h,a}$을 연산한다. 여기서, %, a, $i_{l,(t+a)\%n_l}$, 그리고 $w_{l,(t+a)\%n_l}^{h,a}$는 각각 mod 함수, 프로세서 번지를 나타내는 병렬 인덱스, 유닛 $(l,(t+a)\%n_l)$의 입력값, 유닛 $(l,(t+a)\%n_l)$에서 유닛 (h,a)사이의 가중치를 나타낸다. 연산 과정을 위와 같이 함으로서 주어진 사이클 동안 h번째 층의 각 유닛은 l번째 층의 각기 다른 입력을 사용하여 부분 합을 연산하므로 각 유닛들 사이의 병렬 연산이 가능하다.

(5) 각 프로세서 입력은 n_l개의 프로세서 주위로 초기 입력값을 다음과 같이 회전함으로서 얻어진다. 초기에는 각 프로세서 입력값은 $p(p=0,1,\cdots,n_h-1)$번째 프로세서가 $(p\%n_l)$번째 입력값을 갖는다. t번째 사이클에서는 이전 사이클의 $((n_l-t+a)\%n_l)$번째 프로세서에서 사용된 입력이 a번째 입력으로 가해진다.

(6) $n_l > n_h$의 경우, 처음 n_h출력만이 유효하다.

그림 7은 rf층으로부터 은닉층으로의 한 전향 연산의 연산 과정 구현을 나타낸 것이다. 기본 연산인 곱셈-누

산-회전이 n_l번 반복 연산된다. 두개의 인접하는 층 사이의 전향 연산을 의사 코드로 표현하면 아래와 같다.

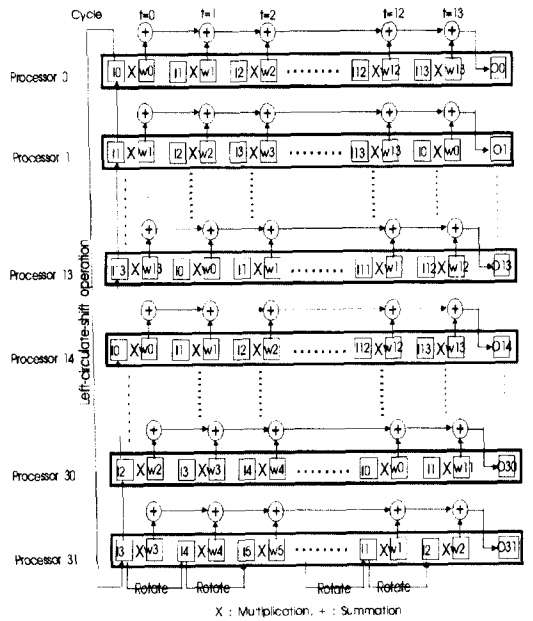

그림 7. 인접 두 층 사이의 전향 연산의 구현예
Fig. 7. Specific implementation of the forward between two layers.

Procedure : forward_pass()

set_status $(0, n_l, 0, n_h)$; activate $n_h \times n_l$ procs.

$o_h[\ a] \leftarrow \mathbf{0.0}$; initialize output.

if($n_l < n_h$) $i_l[\ a] \leftarrow i_l[\ a\%n_l]$; load inputs.

for($t=0$; $t<n_l$; $t++$)

$i_l[\ a] \leftarrow i_l[(n_l - t + a)\%n_l]$; rotate inputs.

$w'[\ a] \leftarrow w[(t+a)\%n_l]$;

retrieve $((t+a)\%n_l)$th

weight in the ath processor.

$o_h[\ a] \leftarrow i_l[\ a] \times w'[\ a]$; accumulate partial

weighted sum in each processor.

}

$o_h[\ a] \leftarrow 1.0/(1.0+\exp(-o_h[\ a]))$; compute

sigmoid function.

여기서, 각 유닛의 어드레스를 나타내는 병렬 인덱스 a는 $0 \le a < n_h$의 범위 값을 가지며, 첨자 h와 l은 각각 인접하는 두 층의 상위 및 하위층을 나타낸다.

2) 단계 2 : 역전파 연산

본 단계에서는 상위 층으로부터 오차를 역전파 시키면서 각 층의 오차를 연산한다. 출력층의 각 유닛의 오차는 아래 식과 같이 연산된다.

$$\delta_{3,k} = o_{3,k} \times (1 - o_{3,k}) \times (t_{3,k} - o_{3,k}) \tag{2}$$

여기서, $t_{3,k}$는 출력층의 k번째 유닛의 원하는 출력값이다. 다른 층의 각 유닛의 오차는 아래 식과 같이 상위 층으로부터의 오차를 역전파 시킴으로서 얻어진다.

$$\delta_{i,k} = o_{i,k} \times (1 - o_{i,k}) \times \sum_{j=0}^{n_{i+1}} w_{i,k}^{i+1,j} \times \delta_{i+1,j} \tag{3}$$

여기서, $\delta_{i,k}$는 유닛 (i,k)에서의 오차를 나타낸다.

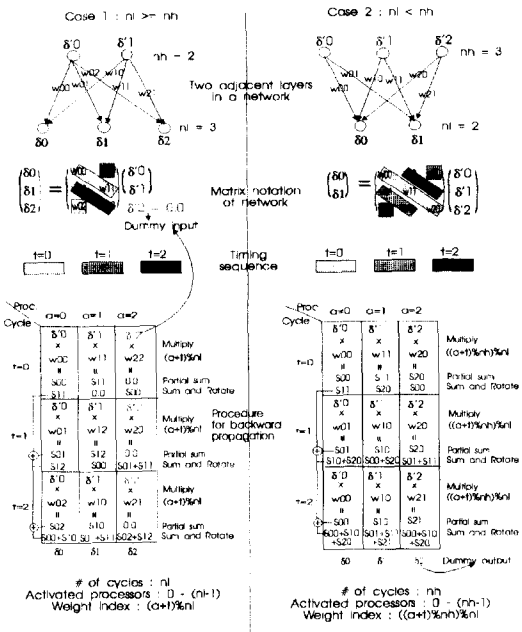

그림 8. 한 신경망에서 역전파 연산의 예
Fig. 8. An illustration of computing the backward propagation.

앞서의 전향 연산에서와 같이 역전파 연산 역시 두 가지 다른 경우($n_l \ge n_h$과 $n_l < n_h$의 경우, 여기서 n_l과 n_h는 각각 l번째와 $h(=l+1)$번째 층의 유닛수를 나타낸다.)를 고려해야 한다. 그림 8은 (1) $n_l=3$ 과 $n_h=2$의 경우, (2) $n_l=2$과 $n_h=3$의 경우를 나타낸 것이다. 다른 모든 인접층 사이에서도 비슷한 연산이 일어난다. 전향 연산의 구현시 특이한 점들을 기술하면 다음과 같다.

(1) 프로세서중 프로세서 번지가 0에서 max(n_l, n_h)-1 사이의 프로세서만 역전파 연산시 사용된다.

(2) $n_l > n_h$의 경우, $(n_h - n_l)$개의 dummy 입력 오차가 삽입된다. 예를 들어 $n_l=5$, $n_h=3$인 경우, n_l개의 프로세서의 초기 오차는 $\delta_0, \delta_1, \delta_2, 0.0$, 그리고 0.0이다.

(3) l번째 층의 한 유닛에서의 오차는 h번째 층의 모든 유닛의 오차들의 가중합이다. 따라서, l번째 층의 한 유닛은 매 사이클마다 h번째 층으로부터 하나의 역전파 오차를 받아 연산하므로 한 오차를 계산하는데 max(n_l, n_h) 사이클이 걸린다.

(4) 출력층의 경우, 오차는 프로세서간 통신 없이 각 프로세서에 얻어진 출력값과 원하는 목적값에 의해 직접 연산 가능하다.

(5) 다른 층의 경우, 각 유닛의 오차는 상위층에서 역전파되어오는 오차에 가중치를 곱하고 이 가중오차를 누적함으로서 얻어진다. t번째 사이클에서 l번째 층의 $\boldsymbol{a}$번째 유닛은 한 부분합 $\delta_h[\boldsymbol{a}] \times w[(t+\boldsymbol{a})\%n_l]$(여기서, $n_l < n_h$인 경우, $\delta_h[\boldsymbol{a}] \times w[((t+\boldsymbol{a})\%n_h)\%n_l]$)을 연산하고 이를 지금까지 누적된 값에 더한다. 새로 얻어진 오차는 하위층의 $(\boldsymbol{a}+1)\%\max(n_l, n_h)$번째 프로세서로 역전파된다.

(6) $n_h > n_l$의 경우, 처음 n_l개의 오차만 유효하다.

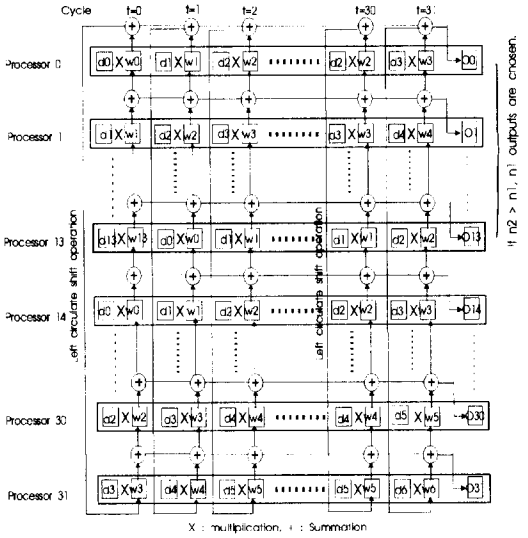

그림 9. 인접 두 층 사이의 역전파 연산의 구현예
Fig. 9. Specific implementation of backward propagation between two layers.

그림 9는 은닉층으로부터 rf층으로의 한 역전파 연산의 연산 과정 구현을 나타낸 것이다. 기본 연산인 곱셈-누산-회전이 max(n_l, n_h)번 반복 연산된다. 두개의 인접하는 층 사이의 역전파 연산을 의사 코드로 표현하면 아래와 같다.

Procedure : backward_propagation()
set_status (0, n_t, 0, max(n_l, n_h)); activate
$n_t \times \max(n_l, n_h)$ processors.
if($n_l \geq n_h$) $\delta_h[\boldsymbol{a}] \leftarrow \mathbf{0.0}$ for $n_h < \boldsymbol{a} < n_l$; load dummy errors.
else {
$\delta'_l[\boldsymbol{a}] \leftarrow \mathbf{0.0}$; initialize the errors.
for($t=0, t<\max(n_l, n_h); t++$) {
if($n_l \geq n_h$) $w'[\boldsymbol{a}] \leftarrow w[(t+\boldsymbol{a})\%n_l]$
else $w'[\boldsymbol{a}] \leftarrow w[((t+\boldsymbol{a})\%n_h)\%n_l]$;
retrieve weights in the $\boldsymbol{a}$th processors.
$\delta'_l[\boldsymbol{a}] \mathrel{+}= \delta_h[\boldsymbol{a}] \times w'[\boldsymbol{a}]$; accumulate partial sum in each processor.
$\delta'_l[\boldsymbol{a}] \leftarrow \delta'_l[(\boldsymbol{a}+1)\%\max(n_l, n_h)]$; rotate the partial sum around processors.
}
}
$\delta[\boldsymbol{a}] \leftarrow o_h[\boldsymbol{a}] \times (1 - o_h[\boldsymbol{a}]) \times \delta'_l[\boldsymbol{a}]$; compute the errors.

여기서, 각 유닛의 어드레스를 나타내는 병렬 인덱스 $\boldsymbol{a}$는 $0 \leq \boldsymbol{a} < \max(n_l, n_h)$의 범위 값을 가지며, 첨자 h와 l은 각각 인접하는 두 층의 상위 및 하위층을 나타낸다.

3) 단계 3 : 가중치 갱신

본 단계에서는 다음 식에 의해 가중치를 갱신한다.

$$\Delta w_{i,j}^{i+1,k}(n+1) = \eta \times \delta_{i+1,k} \times o_{i,j} + \alpha \times \Delta w_{i,j}^{i+1,k}(n) \qquad (4)$$

여기서, $\eta, \alpha, \delta_{i+1,k}$, 및 $o_{i,j}$는 각각 학습율, 모멘트 상수, 유닛 $(i+1, j)$의 오차, 유닛 (i, j)의 출력을 나타낸다.

앞서와 같이 가중치 경우도 두 가지 대표적 예 ($n_l \geq n_h$과 $n_l < n_h$의 경우, 여기서 n_l과 n_h는 각각 l번째와 $h(=l+1)$번째 층의 유닛수를 나타낸다.)를 고려한다. 그림 10은 (1) $n_l=3$ 과 $n_h=2$의 경우, (2) $n_l=2$과 $n_h=3$의 경우를 나타낸 것이다. 다른 모든 인접층

사이에서도 비슷한 연산이 일어난다. 가중치 갱신의 구현시 특이한 점들을 기술하면 다음과 같다.

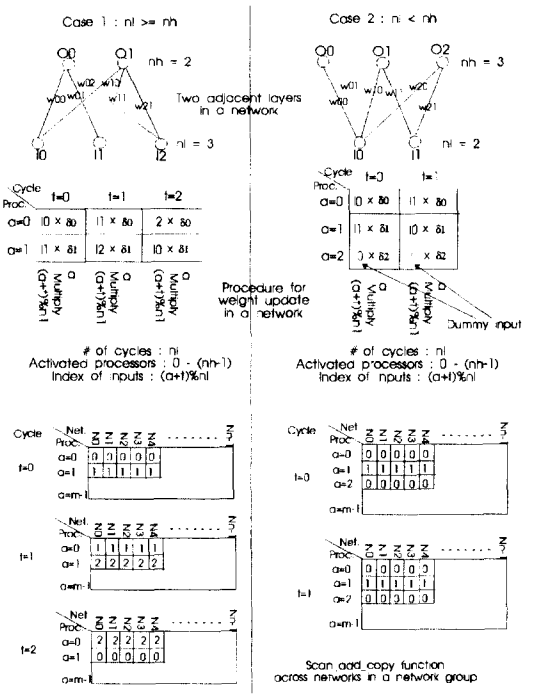

그림 10. 한 신경망에서 가중치 갱신의 예
Fig. 10. An illustration of computing weight updates.

(1) 프로세서중 프로세서 번지가 n_h-1보다 작은 프로세서만 가중치 갱신에 사용된다.

(2) $n_l < n_h$의 경우, (n_h-n_l)개의 dummy 입력이 처음 n_l개의 입력에 반복적으로 연이어 삽입한다. 예를 들어 $n_l=3$, $n_h=7$인 경우, n_h개의 활성 프로세서가 가지는 초기입력은 $i_0, i_1, i_2, i_0, i_1, i_2$ 그리고 i_0이다.

(3) 한 유닛은 바로 아래층의 모든 유닛과 n_l개의 가중치에 의해 연결되며, 가중치 갱신이 한 사이클당 한 가중치씩 갱신되므로 한 유닛의 가중치 갱신에 n_l 사이클이 걸린다.

(4) t번째 사이클에서 h번째 층의 $\boldsymbol{a}$번째 유닛의 가중치 증분은 $i_l[(t+\boldsymbol{a})\%n_l]\times\delta_h[\boldsymbol{a}]$에 의해서 연산된다. 이는 주어진 사이클 동안 h번째 층의 각 유닛은 l번째 층의 각기 다른 유닛의 출력을 사용하므로 각 유닛들 사이의 병렬적인 가중치 갱신이 가능하다.

(5) 다른 학습 샘플에 의한 가중치 증분은 부가적이고 신경망 그룹내 각 신경망들이 가중치를 공유하고 있으므로 모든 학습 샘플에 의한 가중치 증분을 얻기 위해 한 신경망 그룹의 열 방향으로 각 신경망에 의한 증분치들을 합한다. 가중치 갱신은 바로 전의 가중치 값에 이들 전체 가중치 증분합과 모멘텀 항을 더함으로서 얻어진다..

(6) $n_l > n_h$의 경우, 처음 n_h출력만이 유효한 가중치 증분으로 취급한다.

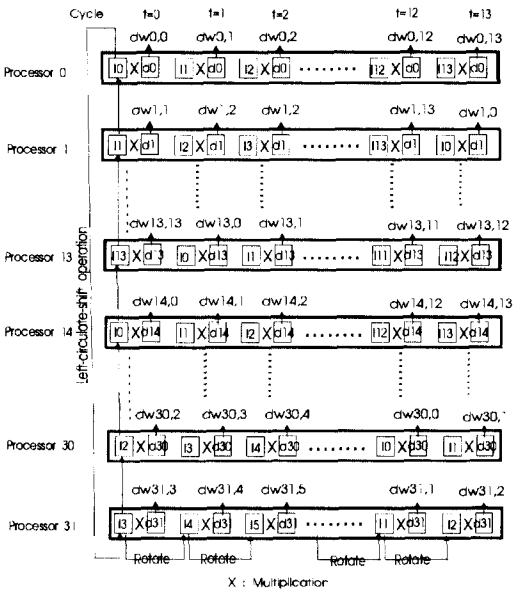

그림 11. 인접 두 층 사이의 가중치 갱신 구현예
Fig. 11. Specific implementation of the weight updates between two layers.

그림 11은 rf층으로부터 은닉층으로의 한 가중치 갱신의 연산 과정 구현을 나타낸 것이다. 기본 연산인 곱셈-스캔-회전이 n_l번 반복 연산된다. 두개의 인접하는 층 사이의 가중치 갱신을 의사 코드로 표현하면 아래와 같다. 여기서 scan_add_copy 연산은 한 신경망 그룹의 열 방향으로 가중치 증분을 더한 후 그 값을 신경망 그룹내 모든 신경망에 다시 보내는 연산을 의미한다.

Procedure : weight_update()
set_status $(0, n_l, 0, n_h)$; activate $n_l \times n_h$ procs.
if($n_l \geq n_h$) $i'_l[\boldsymbol{a}] \leftarrow i_l[\boldsymbol{a}]$; load inputs.
for($t=0$; $t<n_l$; $t++$)
$i_l[\boldsymbol{a}] \leftarrow i'_l[(n_l-t+\boldsymbol{a})\%n_l]$; rotate
initial inputs around processors.

$\Delta w_n[\ a] \leftarrow \eta \times \delta_h[\ a] \times i[\ a]$; compute weight changes.

$\Delta w_n[\ a] \leftarrow scan\_add(\Delta w_n[\ a])$;add weight.

$w_{n+1}[\ a] \leftarrow w_n[\ a] + \Delta w_n[\ a] + \alpha \times \Delta w_{n-1}[\ a]$;update the weights.

}

여기서, 각 유닛의 어드레스를 나타내는 병렬 인덱스 a는 $0 \le a < n_h$의 범위값을 가지며, 첨자 h와 l은 각각 인접하는 두 층의 상위 및 하위층을 나타낸다.

2. 신경망 그룹 수준 구현 (Level 2)

한 신경망 그룹은 n개의 동일한 신경망들로 구성되며, 각 신경망은 한 번에 하나의 학습 샘플을 처리한다. 따라서, 한 신경망 그룹은 한 번에 n개의 학습 샘플을 동시에 처리한다. 나아가, 한 신경망 그룹내 각 신경망은 열 방향으로 놓여 있는 프로세서들 사이에 한 가중치를 공유하고 서로 다른 학습 샘플에 의한 가중치의 변화는 서로 가산적이므로 동시에 가중치를 갱신할 수 있다. 그림 12는 한 신경망 그룹내 각 신경망의 가중치가 scan_add_copy 함수[12]에 의해 어떻게 더해지고 갱신되는가를 나타낸 것이다.

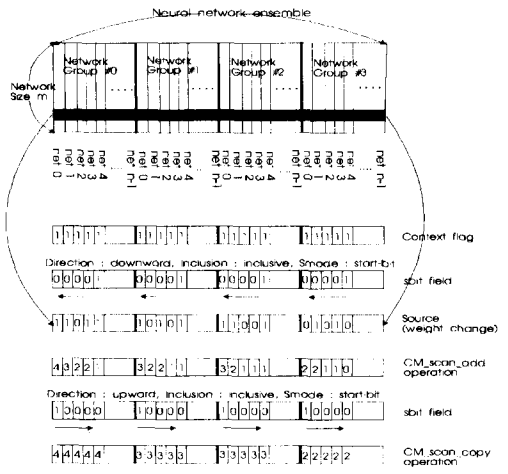

그림 12. Scan_add_copy에 의한 가중치 갱신예
Fig. 12. Illustration of weight update by a scan_add_copy function.

다음은 scan_add_copy 함수의 자세한 설명을 나타낸 것이다.

(1) 여러 개의 신경망 그룹이 CM-2에 동시에 구현되므로 각 신경망 그룹을 구별하는 방안이 요구된다. 이를 위해 그림 12에서처럼 *smode*를 시작 비트 모드로 설정하고, *sbit* 필드를 적절하게 설정하는 것이 요구된다.

(2) t번째 사이클에서 신경망 그룹내 a번째 열에 놓여진 모든 프로세서는 유닛 $(l, (t+a)\%n_l)$과 유닛 (h, a)사이의 가중치 증분을 연산한다. 각 신경망의 얻어진 가중치 증분은 모든 학습 샘플에 대한 가중치 증분을 얻기 위해 scan_add 함수에 의해 신경망 그룹내 신경망들의 열 방향으로 더해진다.

(3) 얻어진 신경망 증분의 합은 scan_copy 함수에 의해 신경망 그룹내 신경망들의 a번째 열에 놓인 모든 신경망의 $((t+a)\%n_l)$번째 가중치 증분으로 복사된다.

3. 신경망群 수준 구현 (Level 3)

한 신경망群은 N개의 신경망 그룹으로 구성되며, 각 신경망 그룹은 각 신경망의 크기가 m인 n개의 동일한 신경망으로 구성된다. 따라서 요구되는 가상 프로세서의 수($=nv$)는 $N \times n \times m$개다. 그러나 물리적 프로세서의 수($=np$)는 CM-2가 접속될 때 유용한 수에 의해 고정된다. CM-2를 사용할 때 이들 두 프로세서간의 비($\frac{nv}{np}$)를 virtual 비($=vpr$)라고 하며 $nv < np$일 때는 $vpr = 1$로 둔다.

신경망群 구현에는 vpr과 신경망 그룹내 신경망의 수($=n$)에 따라 두 가지 형태가 있다. PARIS내 공유메모리 연산(aset_32_shared 또는 aref_32_shared 함수)시, $32 \times vpr$개의 프로세서가 한 데이터를 공유하는데 이 값을($=32 \times vpr$)을 블록 폭($=bw$)이라고 부르면, 블록 폭과 한 신경망 그룹내 신경망의 수 사이의 관계에 의해 형태 1: $(bw \ge n)$과 형태 2: $(bw < n)$로 구분할 수 있다.

1) Configuration 1 : $(bw \ge n)$

이 경우, bw/n신경망 그룹이 한 블록으로 구현 가능하다. 한 신경망의 가중치 수를 k라고 가정한다. 그러면, bw/n개의 신경망 그룹의 가중치는 한 블록내의 신경망에 걸쳐 다음과 같이 저장된다. 즉, i번째 신경망 그룹의 k개의 가중치가 가중치 어레이의 $(i-1) \cdot k$에서 $i \cdot (k-1)$ (여기서, $i = 0, 1, \cdots, bw/n-1$)에 저장된다. 예를 들면, np, N, n, m, 및 k 가 각각 8,192, 8, 128, 64, 및 10이라고 하자. 그러면, $nv = 8 \times 128 \times 64 = 65,536$이고 $vpr = 65,536/8,192 = 8$이다. 각 블록의 크기가 256($= 32 \times 8$)이고 한 신경망 그룹내 신경망의

수가 128개이므로 한 블록내 2개의 신경망 그룹이 가능하다. 첫 번째 신경망 그룹의 가중치는 가중치 어레이의 0에서 9번까지에 저장되며, 두 번째 신경망 그룹의 가중치는 가중치 어레이의 10에서 19번까지에 저장된다.

2) Configuration 2 : ***(bw< n)***

블록 폭이 한 신경망 그룹내 신경망 수보다 작으므로, n/bw블록의 프로세서가 한 신경망 그룹을 형성한다. 한 신경망 그룹내 한 블록이 갖는 가중치가 다른 블록들로 복사된다. 예를 들면, np, N, n, m, 및 k가 각각 8,192, 4, 64, 32, 및 10이라고 하자. 그러면, $nv = 8\times64\times32 = 8,192$이고 $vpr = 8,192/8,192 = 1$이다. 각 블록의 크기가 32(= 32×1)이고 한 신경망 그룹내 신경망의 수가 64개이므로 한 신경망 그룹을 위해 두 블록이 요구된다. 신경망 그룹내 첫 번째 블록의 가중치는 가중치 어레이의 0에서 9번까지에 저장되며, 신경망 그룹내 두 번째 블록을 위한 가중치 어레이의 같은 주소로 복사된다.

4. 시스템 통합

위에서 언급한 세 가지 수준의 구현 - 신경망 수준, 신경망 그룹 수준, 신경망群 수준 -을 하나로 통합한 것이 그림 13에 나타나 있다. 이는 기하학적 큐빅 구조 $g(N, n, m)$(여기서, $N, n,$ 및 m은 각각 신경망 그룹수, 신경망수, 신경망 크기를 나타낸다.)를 갖고 전체적으로 $N\times n\times m$의 가상 프로세서를 요구한다. 각 프로세서는 완전 결합 신경망 구조의 경우 $4+3\cdot m$의 국부 메모리를 할당해야 한다.

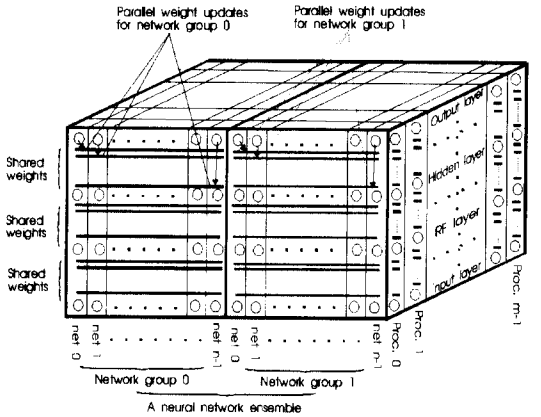

그림 13. CM-2상에 신경망群의 전체 구성
Fig. 13. Overall organization of neurual network ensemble on the CM-2.

다음은 제안한 신경망群을 학습시키는 과정을 의사 코드로 나타낸 것이다.

```
Procedure : training()
load training samples;
load initial random weights;
while(not convergent) {
    do parallel(all network groups) {
        do parallel (all networks)
            forward_pass();
            backward_propagation();
            weight_update();
        }
    }
}
```

Ⅳ. 실험 결과

CM-2상에 구현된 신경망群의 학습 시간 및 수렴 특성에 관한 여러 실험이 수행되었다. 신경망群에 의한 인식 능력의 향상은 다른 논문[11]에 잘 나타나 있다. 제안한 신경망群을 16개의 서로 다른 열쇠를 인식하는 문제에 적용하여 보았다. 사용된 신경망群은 서로 다른 초기값으로부터 학습한 결과 서로 다른 가중치 값을 갖는 16개의 신경망 그룹으로 이루어져 있고 각 신경망 그룹은 같은 가중치를 서로 공유하는 96개의 신경망으로 구성되어 있다. 각 신경망은 하위층부터 각 층이 52, 13, 32, 16개의 유닛을 4층의 전향 구조를 갖는다.

학습을 위해 16개의 서로 다른 열쇠를 이용하였는데 각 열쇠의 이미지는 64×64 픽셀 크기를 갖고 화상내 열쇠 크기가 대략 54×54가 되도록 정규화되어 있다. 인식을 위한 특징점으로는 각 열쇠내 구석점들이 사용되었다. 구석점은 Medioni와 Yasumoto의 방법[18]을 사용하여 경계선을 따라 곡률의 변화를 측정하여 얻었다. 그림 14는 사용된 16개의 열쇠 이미지의 경계선으로부터 얻은 구석점을 'x'표로 나타낸 것이다.

1. 수렴 특성

본 논문에서는 자세한 설명은 피했지만 은닉층의 유닛수는 학습에 걸리는 시간과 오인식율의 적당한 타협점으로 32개를 선택하였다. 여기서는 원래의 역전파 알고리즘과 모든 유닛내 sigmoid 함수의 기울기 파라메

터를 매 iteration마다 유닛내 오차값에 따라 탄력적으로 변하는 수정된 역전파 알고리즘[14]과의 수렴 특성을 비교하였다.

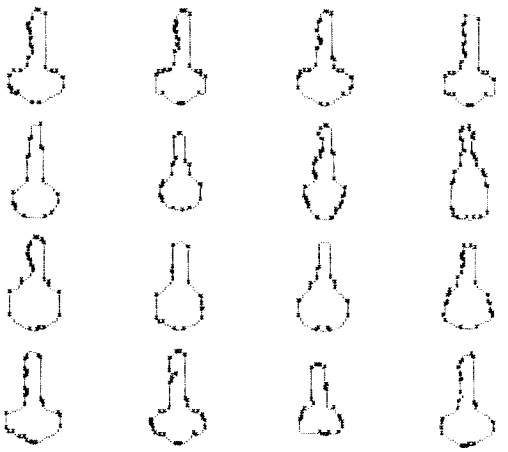

그림 14. 학습 샘플의 경계선 및 구석점
Fig. 14. Boundaries and corner points extracted from key images.

그림 15는 서로 다른 학습 파라메터 ($\eta=0.2$, $\alpha=0.3$과 $\eta=0.3, \alpha=0.5$)와 다른 학습 방법을 사용한 경우의 평균 자승 오차(MSE)를 나타낸 것이다. 여기서 평균 자승 오차는 전체 유닛에 의한 오차의 자승의 합을 전체 유닛수로 나눈 값이다. 이 그림으로부터 두 가지 학습 방법은 매우 상이한 수렴 특성을 보임을 알 수 있다. 원래의 역전파 알고리즘의 경우 오차는 학습 초기에 아주 완만하게 줄어들다가 어떤 시점이 지나면 오차가 급격히 줄어듦을 알 수 있다. 수정된 역전파 알고리즘의 경우 오차는 학습 초기부터 매우 넓은 범위에서 유동적이면서 줄어들다가 어느 시점이 지나면 거의 정체되어 줄지 않음을 알 수 있다. 이상의 차이점의 원인은 다음에서 찾을 수 있다. 원래의 역전파 알고리즘의 경우 초기 가중치의 불규칙성 때문에 각 유닛의 출력값이 0 또는 1에 치우쳐 있다. 따라서 초기 오차가 아주 큼에도 불구하고 $o\cdot(1-o)$에 비례하는 가중치 변화량이 작은 값을 가지므로 MSE가 초기에는 비교적 완만히 변한다. 수정된 역전파 알고리즘의 경우 sigmoid 함수의 기울기 파라메터를 조정하여 출력값을 0.5 근방에 머무르게 한다. 따라서, 학습 초기 MSE는 매우 불안정하지만 이러한 불안정성이 수렴을 가속시키는 원인이 된다. 어느 정도 학습이 진행되면 시스템은 안정되고 이후는 원래의 역전파 알고리즘과 비슷한 수렴 특성을 갖는다. 결론적으로 sigmoid 함수내 기울기 파라메터의 조정은 시스템의 안정성을 약간 희생하면서 대신 수렴 속도를 빨리 하는 역할을 하는 것을 알 수 있다.

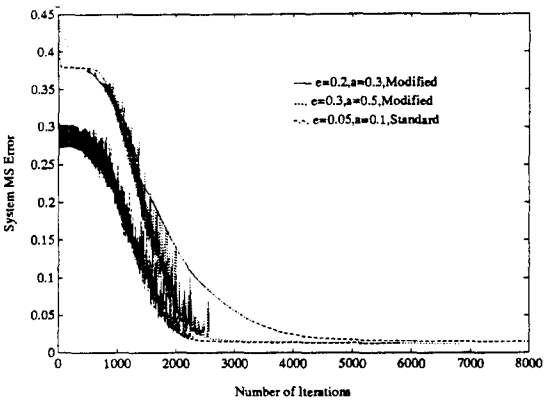

그림 15. 서로 다른 학습 방법에 대한 MSE
Fig. 15. System MSE for different learning methods.

2. 연산 시간 비교

본 연구에서는 CM-2가 갖는 병렬 연산성을 최대한으로 이용하고자 여러 개의 학습 샘플을 동시에 로드하여 여러 개의 신경망을 동시에 학습하여 학습 시간을 크게 줄였다. 연산 시간을 측정하기 위해 사용한 시험 신경망은 256개의 입력 유닛, 64개의 rf 유닛, 64개의 은닉 유닛, 그리고 64개의 출력 유닛으로 구성되어 있다. 이 시험 신경망이 64개의 프로세서에 매핑되어 있고 각 프로세서에는 148개의 가중치가 (20개의 입력-rf층 연결, 64개의 rf-은닉층 연결, 64개의 은닉-출력층 연결) 저장되어 있다. 따라서 신경망 전체는 9,472개의 가중치 연결이 (64x20개의 입력-rf층 연결 + 64x64개의 rf-은닉층 연결 + 64x64개의 은닉-출력층 연결) 있다.

한 신경망 그룹이 128개의 신경망으로 이루어져 있고 신경망들 사이의 가중치는 신경망 그룹내 신경망들의 열 방향으로 서로 공유하고 있다. 신경망群내 신경망 그룹수(=NNG)는 1, 4, 8, 16, 및 32로 변화시켜 보았다. 따라서 요구되는 가상 프로세서의 수는 64x128xNNG개이다. 여기에서 관심을 갖는 것은 신경망群에 의한 인식 능력이 아니라 구현한 신경망群의 연산 속도이므로 사용한 학습 샘플은 랜덤하게 발생한

난수를 이용하였다.

그림 16은 서로 다른 신경망 그룹수를 갖는 신경망群의 평균 iteration 시간을 나타낸 것이다. 본 실험에서는 8K개와 16K개의 물리적 프로세서를 사용하는 두 가지 경우에 대해서 iteration 시간을 측정하였다. 이 그림내 선분 위에 나타낸 숫자는 virtual 비를 의미한다. 이 그림으로부터 평균 iteration 시간은 물리적 프로세서에 의존하는 것이 아니고 virtual 비에 의존함을 알 수 있고, virtual 비가 증가함에 따라 선형적으로 증가함을 알 수 있다.

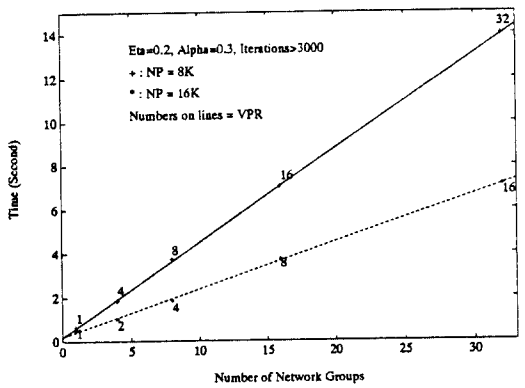

그림 16. 서로 다른 신경망群의 학습 시간
Fig. 16. Learning time of different NN ensembles.

신경망 시뮬레이션에서 연산 속도를 비교하기 위해 단위 시간당 가중치 갱신수 (WUPS, weight update per second)가 널리 쓰이는데 이에는 전향 연산, 오차 역전파 연산 및 가중치 갱신에 걸리는 시간을 포함한다. NNG=4이고 NP=8K일 때 (즉 vpr=4 일 때), 평균 iteration 시간은 1.828초로 전향 연산시 0.422초, 역전파 연산시 0.388초, 가중치 갱신시 1.018초가 각각 걸린다. 신경망群내 전체 가중치 연결수는 9,472 x 128 x 4 = 4,849,664개이다. 따라서 CM-2상의 본 시뮬레이션은 4,849,664/1.018 = 2,652,989 WUPS의 연산 속도를 나타냄을 알 수 있다. 만약 CM-2상의 64K개 프로세서를 모두 이용한다면, 8개의 신경망群을 동시에 구현할 수 있으므로 최대 연산 속도는 8 x 2,652,989 = 21,223,912 WUPS가 가능하다. 본 실험에서 전향 연산이 전체 연산 시간의 약 23%를 차지하므로 전향 연산만을 고려하면 92,277,878 IPS[1](interconnection per second)를 얻는 결과가 된다.

이 결과를 Zhang 등[10]에 의해 수행된 연산 속도 (=182,494,940 IPS)와 비교하면 약 반 정도의 연산 속도에 불과하다. 이와 같은 낮은 연산 속도의 원인은 다음에서 찾을 수 있다.

(1) 본 연구에서 제안한 구현은 라우터에 의해 제공되는 일반적인 프로세서간 통신 방법을 사용하였지만 Zhang 등에 의한 구현은 프로세서간 통신으로 NEWS 그리드 통신을 사용하였다. 따라서, 그들의 구현은 모든 층내의 유닛수가 2의 멱수만이 가능한 반면 본 연구에서 제안한 구현은 임의의 크기를 갖는 어떤 신경망도 시뮬레이션 가능한 장점이 있다.

(2) Zhang 등에 의한 구현은 모든 층간의 연결이 완전 결합형인 반면 본 연구에서는 층간의 결합이 부분 결합인 경우에도 시뮬레이션 가능하다. 양 구현에 있어 두 층 사이의 곱셈-회전-누산 연산수가 비슷하지만 Zhang 등에 의한 구현이 본 연구에서 구현한 신경망보다 훨씬 많은 가중치 연결을 가지고 있으므로 결과적으로 가중치 갱신수가 훨씬 크다. 실제 그들의 구현시 한 신경망내 가중치 연결은 65,536개 (=256x128 + 128x256)인데 반해 본 연구의 경우는 9,472개에 불과하다. 결과적으로 비슷한 가중치 연결을 가정하면 본 연구 결과의 연산 속도가 $\frac{92}{182} \times \frac{65,536}{9,472}$ =3.49배의 속도 개선을 이룰 수 있음을 알 수 있다.

표 1. 서로 다른 머신상의 역전파 알고리즘 구현 성능 비교
Table 1. Comparison of backpropagation implementation on different machines.

Machines	BP Performance(IPS)
CM-2(Zhang et al.)	180M
CM-2 (Ours)	92M
Cray X-MP	50M
WARP (10)	17(WUPS)
VAX 8600	2M
Sun 3	250K
Symbolic 3600	35K

표 1은 여러 가지 다른 머신 상에 구현된 역전파 알고리즘의 연산 성능을 비교한 것이다. 표에 나온 성능값들은 여러 논문[10,19]을 참고로 한 것이다. 이들 성능

1. IPS는 초당 수행되는 곱한 후 더하는 일련의 연산 횟수로 정의된다.

간의 직접 비교가 정당화된 것은 아니지만 본 구현에서 얻은 성능이 비교적 우수하다는 것을 확인할 수 있다.

V. 결 론

제안한 신경망群은 여러 개의 신경망의 학습을 요구하므로, 각 신경망을 기존의 순차적 머신에 하나씩 학습시키는데는 많은 시간이 걸린다. 그러나, 신경망群내 모든 신경망은 동일한 구조에 서로 다른 가중치를 가지므로 이들은 SIMD 구조의 병렬 컴퓨터상에서 같은 학습 샘플을 사용하여 서로 다른 초기 가중치 값을 가지고 동시에 학습 가능하다.

신경망群의 병렬 구현을 위해 CM-2를 사용하였는데 모든 학습 샘플이 한 신경망당 하나씩 동시에 가해졌고, 신경망群내 모든 신경망들이 서로 다른 초기 가중치를 갖고 동시에 학습되었다. 이를 기존의 순차적 머신과 비교해보면, N, n, 및 m을 각각 신경망群내 신경망수, 각 신경망에서 사용하는 학습 샘플수, 및 신경망내 층 당 유닛수라고 하면 전체 연산 속도를 $O(N\times n\times m)$배 향상시킬 수 있다. 본 연구에서 구현한 역전파 학습 알고리즘의 연산 속도는 약 92 MIPS로 Zhang 등[10]의 구현에 비해 반 정도이지만 Zhang 등의 구현이 각 층 당 유닛의 수가 2의 멱수를 갖는 신경망의 경우만 시뮬레이션 가능한 반면 본 연구에서의 구현은 어떤 형태의 신경망이라도 시뮬레이션 가능한 이점이 있다.

제안한 수정된 역전파 알고리즘의 경우, sigmoid 함수내 기울기 파라메터를 매 iteration마다 각 유닛내 존재하는 오차의 크기에 따라 적절히 조정함으로서 학습 초기에 시스템이 갖는 오차가 매우 불안정하게 오르내리지만 이러한 불안정성이 학습의 수렴을 가속시키는 성질을 나타내는 것을 확인하였다.

참 고 문 헌

[1] D. Kim and M. Suk, "Invariant object recognition using neural network ensemble," *Progress in Neural Networks Volume IV*, Ablex Publishing Corporation, 1997. In print.

[2] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-12, no. 10, pp. 993-1001, Oct. 1990.

[3] Thinking Machine Corporation, Connection machine model CM-2 technical summary, *Technical Report*, No. 86-14, Cambridge, MA, 1986.

[4] W. Lim, A Agrawal, and L. Nekludova, "A fast parallel algorithm for labeling connected components in image arrays," *Thinking Machine Corporation*, Tech. Rep. NA86-2, Dec. 1986.

[5] V. Narayanan and V. Pitchumani, "A parallel algorithm for fault simulation on the Connection Machine," *1988 IEEE International Test Conference*, pp. 89-93, 1988.

[6] O. A. McBryan, "The Connection Machine: PDE solution on 65536 processors," *Parallel Computing*, vol. 9, pp. 1-24, 1988-1989.

[7] L. W. Tucker, C. R. Feynman, and D. M. Fritzsche, "Object recognition using the Connection Machine, *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 871-977, June, 1988.

[8] E. Deprit, "Implementing recurrent back propagation on the Connection Machine," *Neural Networks*, vol. 2, pp. 295-314, 1989.

[9] C. R. Rosenberg and G. Blelloch, "An implementation of network learning on the Connection Machine," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, 1987.

[10] X. Zhang, M. Mckenna, J. P. Mesirov, and D. L. Walts, "An efficient implementation of the back-propagation algorithm on the Connection Machine CM-2," *Advances in Neural Information Processing Systems*, vol. 2, pp. 801-809, 1990.

[11] MIT Lincoln Lab., DARPA neural network study, *Final Report*, July, 1988.

[12] Thinking Machine Corporation, Connection machine parallel instruction set Cambridge, MA, 1988.

[13] D. H. Hubel, T. N. Wiesel, and M. P. Stryker, "Anatomical demonstration of orientation columns in Macaque monkey," *J. comparative Neurology*, vol. 177, no. 3, pp. 361-380, 1987.

[14] D. Kim and M. Suk, "Parallel Implementation: A neural network ensemble on the CM," *Progress in Neural Networks Volume V*, Ablex Publishing Corporation, to be accepted for publication.

[15] M. B. Reid, L. Sprikovska, and E. Ochoa, "Rapid training of higher-order neural networks for invariant pattern recognition." *Proc. IEEE 3rd International Conference on Neural Networks*, vol. 1, pp. 689-692, June, 1989.

[16] K. Yamada, H. Kami, and J. Tsukomo, "Handwritten neural recognition by multi-layered neural network with improved learning algorithm," *Proc, IEEE 1st International Conference o on Neural Networks*, vol. 2, pp. 259-266, June, 1987.

[17] S. Becker and Y. Le Cun, "Improving the convergence of back-propagation learning with second order methods," *Proceedings of the 1988 Connectionist Models*, pp. 29-37, June 17-26, 1988.

[18] G. Medioni and Y. Yasumoto, "Corner detection and curve representation using cubic B-splines," *Comput. Graph. Image Processing*, vol. 39, pp. 267-278, 1987.

[19] D. A. Pomerleau, G. L. Gusciora, D. S. Touretzky, and H. T. Kung, "Neural network simulation at warp speed: How we get 17 million connections per second," *IEEE Int. Conf. on Neural Networks*, July, 1988, San Diego, CA.

저 자 소 개

金 大 鎭(正會員) 第 34卷 B編 第 1號 參照
현재 동아대학교 컴퓨터공학과 조교수