論文 92-29B-2-6

신경회로망을 이용한 한글 한자 혼용 문서 인식에 관한 연구

(A Study on Korean and Chinese Character Document Reader Using Neural Network)

金 祐 成*, 方 勝 楊*

(Woo Sung Kim and Sung Yang Bang)

要 約

기존의 대부분의 한글 문자 인식에서는 한글의 구조적 형태에 의해 6가지로 분류하여 자소별로 인식하는 방법을 사용하여 왔다. 본 연구에서는 한글과 다른 문자를 동일하게 취급하여 문자단위로 인식함으로써 한글과 한자가 혼용된 문서를 인식할 수 있는 신경회로망을 설계하였다. 우선 SOFM 모델을 수정하고, 한글 및 한자의 유형을 분류하는데 이를 이용하였다. 분류된 유형 내에서 문자를 인식하는데는 APC를 사용하였고 따라서 학습 속도가 빠르며 추가학습이 가능하다는 장점이 있다. 실험 결과 한글과 한자가 혼용된 문서를 약 94% 이상 인식하였다.

Abstract

In the most studies of Korean character recognition so far, they first classify the characters to 6 types according to their structures and then recognize the characters by identifying their basic components named "jaso." In this study, we propose a method which recognizes the characters without using structure types and is applied to reading documents containing both Korean and Chinese characters. We first classify Korean and Chinese characters by using a modified SOFM model. Then we recognize the characters in each class by using an APC neural network which has the advantage of fast learning speed and the capablity of additive learning. An experimental result demonstrated the usefulness of the approach with the recognition rate of 94%.

I. 서 론

최근 사회가 정보화됨에 따라 다량의 정보가 통용되고 있으며 이에 따라 음성 인식이나 문자 인식과 같이 인간과 기계 사이의 정보교환 수단에 관한 연구가 활발히 진행되어 왔다. 음성 인식은 그 자체의 특성으로 인해 실용화에 이르기까지는 아직 요원한 반면 문자 인식은 비교적 많은 성과를 거두어 왔다. 특히 인쇄체의 경우는 실용화 단계에까지 이르렀고 필기체의 경우도 펜 컴퓨터에 응용될 수 있는 온라인(on-line) 인식 분야에서 많은 진전을 이루어 왔다. 인쇄체 중 현재까지 연구가 진행되고 있는 분야는 문서 인식, 즉 사람과 컴퓨터가 문서를 이용하여 대화할 수 있도록 해주는 분야이다. 하나의 문서를 입력시키기 위해서는 사람이 일일히 타이핑(typing) 하거나 이미지 스캐너(image scanner)를 이용하여 받은 영상 그대로 저장하는 방법이 있다. 전자의 경우 많은 시간 및 인력이 요구되는 반면 후자의 경우는 입

*正會員, 浦項工科大學校 電子計算學科
(Dept. of Computer Sci., POSTECH)
接受日字: 1991年 11月 18日

력은 간단하지만 화일의 용량이 너무 크다는 단점이 있다. 그래서 대두된 방법이 스캐너를 통해 받은 영상을 문자 코드의 형태로 바꾸어 저장하는 문서 인식이다.

현재까지 개발된 문서 인식 시스템은 다중 크기, 다중 활자체 한글 및 영문 문서를 인식할 수 있다.[1,2] 그러나 기존의 대부분의 방법이 한글의 구조적 특성, 즉 받침의 유무 및 모음의 형태 등의 정보를 이용하여 문자를 인식하여왔고, 따라서 한글이 아닌 다른 문자가 혼용되었을 때는 이 문자가 한글인지 아닌지를 먼저 판별하고 이에 따라 문자를 인식하여야만 했다. 그러나 현재 많은 문서에서 한글 이외의 다른 문자, 즉 영문 및 한자 등이 혼용되어 있으며 따라서 이런 문서를 빠르게 인식하기 위해서는 한글과 그 이외의 다른 문자를 구분하지 않고서도 인식할 수 있는 방법이 요구된다. 따라서 본 연구에서는 한글과 한자를 구분하지 않고 인식할 수 있는 방법을 제안하고자 한다.

Ⅱ. 연구의 배경

우리나라는 중국, 일본, 동남아 일부 등과 함께 한자 문화권에 속해 있으며 수천년전부터 이미 한자를 사용해 왔다. 한자는 표의적인 상형문자이기 때문에 각 문자 하나하나마다 내포된 의미가 다르고 그 수도 방대하여 배우기도 어려운 문자이다. 또한 한 문자가 다른 문자와 결합을 하여 새로운 문자를 형성하는 경우도 있고 문자들 간의 유사도나 문자 구조상의 복잡도도 다른 여느 문자보다 높기 때문에 기계를 이용하여 한자를 인식한다는 것은 어려운 문제이다. 우리나라에서는 한자의 이런 난해한 점으로 인해 오래전에 한글이 만들어져 사용되고 있지만 그 이전부터 사용해 오던 한자의 영향을 아직도 많이 받고 있는 것은 사실이다. 아직까지 대부분의 사람들의 이름은 한자어인 경우가 많다. 실제로 많은 문서에서 사람의 인명은 한자로 표기하는 경우가 많고 우리말 큰사전을 살펴보면 여기에 실려 있는 전체 우리말 140,464개 중에서 약 60.8%에 해당하는 85,527개가 한자말인 것으로 나타나 있다.[3]

이처럼 한자가 우리 문화에 깊숙히 침투해 있음에도 불구하고 현재 국내에서 한자에 대한 연구는 극히 빈약한 실정이며[4] 더구나 한글과 한자를 동시에 인식하는 것에 대한 연구는 전혀 없었다. 이는 일본과 같이 한자의 사용 빈도가 우리나라보다 높은 나라에서 그 필요성에 의해 더 먼저 그리고 많은 성과를 이루어 왔다는 점을 이유로 들 수도 있겠지만, 우리나라에서 한글만의 구조적 특성에 너무나 의존된 연구를 진행해 왔기 때문이라는 것이 더 큰 요인인 것으로 생각된다. 즉 한글은 몇개의 소수 자소들이 극히 제한된 위치에서 상호결합하여 문자를 형성하게 된다. 국내에서는 한글의 이러한 단순성을 이용하여 한글의 구조적 특성에 맞는 인식 방법, 즉 각 문서를 구성하는 자소를 분리하여 이 자소별로 인식하는 방법이 주류를 이루어 왔다. 그러나 이러한 방법을 사용할 경우 한글은 잘 인식할 수 있지만 한글과 한자, 혹은 영문 등이 같이 혼용된 문서를 인식하려면 먼저 그 문자가 한글, 한자, 혹은 영문인지를 먼저 구분해야 하는 부가적인 과정을 거쳐야만 한다.

Ⅲ. 문서 인식

1. 개괄 및 문자 추출

본 연구에서는 한글과 한자가 혼용된 문서를 인식할 수 있는 시스템을 위한 기초를 구성하였다. 전체적인 시스템의 구조는 그림 1에 나타낸 바와 같이 문서 영상으로부터 문자를 추출해 내는 과정, 추출된 각각의 문자의 유형을 분류하는 과정, 분류된 각 유형내에서 문자를 인식하는 과정으로 나눌 수 있다. 문서 영상에서 문자를 추출해 내는 과정은 수평 투영을 통해 문자행을 추출하고 추출된 각 문자행에 대해 수직투영을 통해 개개의 문자를 분리해 내게 된다. 따라서 본 연구에서 인식할 수 있는 문서는 기본적으로 가로쓰기로 구성되고, multicolumn이 아닌 문서이어야 한다. 이때 문서 영상의 입력시에 생긴

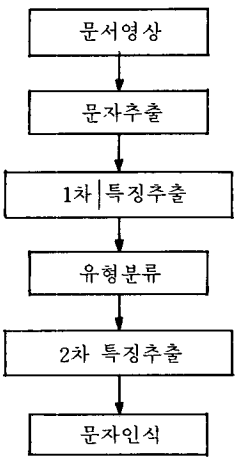

그림 1. 전체 시스템의 구성도
Fig. 1. Block diagram of the total system.

잡음이나 혹은 원래 입력문서 내의 문제로 인해 연속되는 문자를 사이에 접촉이 발생한 경우를 볼 수 있다. 이러한 접촉은 수평 모음의 연속 등 그 형태를 6가지로 분류할 수 있는데[5] 본 연구에서는 인식 대상 문자를 한글과 한자, 그리고 마침표로 제한을 하였기 때문에 일단 문자행이 추출되면 그 높이 정보에 의해 각 문자의 너비를 예측할 수 있다. 그래서 그 예측된 부근에서 가장 낮은 수직 투영값을 갖는 위치에서 강제 분리를 수행하였다. 따라서 한 문자행에는 같은 크기의 문자들만이 존재하여야 한다. 실제로 우리가 일상 접하고 있는 대부분의 문서에서도 한 행에 크기가 다른 몇개의 문자가 동시에 존재하는 경우는 많이 볼 수 없다. 즉 윗첨자, 아랫첨자를 사용하는 경우가 있긴 하지만 일부 제한된 분야에 한정되므로 이러한 경우는 배제하였다. 그리고 이렇게 추출된 문자에 대해서는 모두 동일한 크기, 즉 40×40의 크기로 정규화 과정을 거치도록 한다. 여기서 40×40의 크기는 Qnix lfmt 형식으로 h4크기의 문자를 300dpi를 갖는 이미지 스캐너로 입력받았을 때의 크기이며 이보다 큰 문자는 정규화를 통해 축소시킬 수 있다. 따라서 1fmt의 h4크기 이상의 문자로만 구성된 문서이어야 하는데 이 크기는 우리가 일상적으로 접하는 대부분의 문서를 포함할 수 있는 크기이다. 그 외에 현재 인식할 수 있는 활자체는 한글의 경우는 명조체와 고딕체, 한자는 명조체인데 한글의 경우 많은 활자체가 존재한다고 볼 수도 있지만 가장 많이 사용되는 활자체가 명조체와 고딕체이고 또 다른 활자체들도 이 두 활자체로부터의 변형이라 볼 수 있기 때문에 그런 제한을 두었다. 그러나 어떠한 새로운 활자체도 추가학습할 수 있기 때문에 다중 활자체를 인식한다고 볼 수 있다.

2. 문자의 특징 추출

문자의 특징 추출이란 입력된 각 문자 영상으로부터 실제 인식이 용이하도록 어떤 다른 특징을 뽑아내는 과정을 말한다. 즉 입력된 이진 영상은 보통 40×40 이상의 크기를 갖게 되는데 이를 그대로 이용하여 문자인식을 할 경우 많은 기억 용량과 시간을 소비하므로 이를 좀 더 낮은 차원의, 그리고 인식이 쉬운 특징을 뽑아내어 사용하는 것이다. 보통 문자인식에 많이 쓰이는 특징으로는 mesh vector와 방향 성분(directional component)을 들 수 있다. mesh vector는 blurring 효과에 의해 정보를 압축시키고 약간의 위치 이동을 흡수할 수 있다는 장점이 있다. 방향 성분이란 이진 영상에 대해 수평, 수직 그리고 대각선 방향으로 투영(project)시켜 이 값을 이용하는 방법이다. 이는 실제로 한글을 2차원 FFT하면 방향 성분이 그 글자의 주요 특징을 이루고 있다는 사실로부터 유추된 방법이다.[6] 그러나 이 방법으로는 문자의 위치 이동이 심한 경우에 인식율이 저하된다는 단점이 있다.

기존의 문자 인식에서 유형분류를 위해 mesh vector를 사용하고 다시 문자인식을 위해서도 mesh vector를 특징으로 그대로 사용한 경우가 있다.[7] 그러나 이 경우 유형분류를 통해 같은 유형으로 분류된 문자는 모두 mesh vector 특징상으로 볼 때는 모두 유사한 문자들이다. 이렇게 mesh vector 특징상에서 유사한 문자들을 인식할 때도 역시 mesh vector를 사용하여 인식한다는 것은 별로 좋은 방법이라 할 수 없다. 따라서 유형분류시에 사용하는 특징과 문자 인식시에 사용하는 특징은 상호 보완적인 관계에 있는 것이 이상적이라 할 수 있으나 본 연구에서는 상호 보완적이라고는 할 수 없지만 새로운 특징을 추출하여 유형분류를 하는데 사용하였다.

여기서 사용한 특징은 우선 문자를 수평, 수직의 두 방향으로 투영시켜서 양 방향에 대해 히스토그램(histogram)을 얻는다. 그러나 이 히스토그램을 그대로 template matching시켜 사용할 경우에 같은 글자라 하더라도 약간의 위치 이동에 대해서도 많은 차이를 나타내게 된다. 히스토그램이란 문자의 2차원 정보로부터 1차원 정보를 뽑아낸 것이라 볼수 있다. 실제로 1차원 신호의 경우에 위치 이동에 무관하면서 template matching을 시키기란 그리 쉬운일이 아니며 1차원 신호인 음성 신호의 경우 포만트(formant) 주파수라 하여 음성 스펙트럼 상에서 에너지가 가장 높은 몇 주파수만을 골라내어 이를 이용하여 인식을 하는 방법이 있다. 그래서 여기서도 히스토그램 상에서 가장 높은 위치를 골라내어 그 위치와 히스토그램 값을 특징으로 한다. 그리고 다시 이 위치로부터 히스토그램을 좌 우로 따라가면서 히스토그램 값의 변화량의 합을 또하나의 특징으로 사용하였다. 그래서 좌우 두 개의 특징이 생기고 따라서 각 수평, 수직의 히스토그램에 대해 최대치와 그 위치, 그리고 좌우의 변화량의 합 두개, 그렇게 네개의 특징이 생기므로 전체적으로는 8차원의 정보를 이용하게 되는 것이다. 그림2에 한문자에 대한 수평 수직의 히스토그램과 최대치를 나타내었다.

그리고 분류된 유형 내에서 문자를 인식할 때는 현재 널리 쓰이고 있는 mesh vector를 이용하였다. 즉 40×40 크기로 정규화된 각 문자에 대해 5×8 크기의 영역으로 나누어 40차원의 mesh vector를 만들어 사용하였다. mesh vector는 그 특징을 추출하는

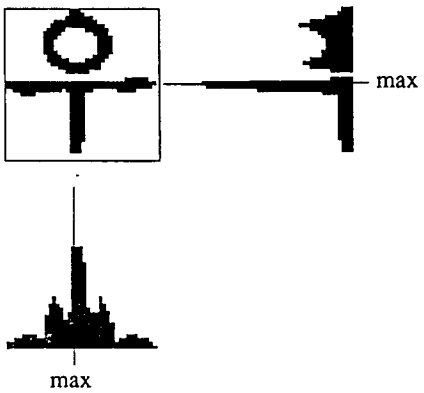

그림 2. 한 문자의 수평 수직 히스토그램과 최대 위치
Fig. 2. Horizontal and veritical histogram of a character image and thier max. points.

과정이 간단하고 전체적인 문자의 형태를 유지하면서 약간의 위치 이동을 흡수할 수 있기 때문에 많이 쓰이고 있다. 그리고 이보다 더 심한 위치 변화를 흡수하기 위하여 각 문자의 흑화소가 시작되는 좌상 기준점을 구하여 여기로 정렬시켜서 mesh vector 를 구하였는데 그림3에 그 예가 나타나 있다.

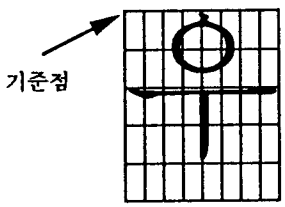

그림 3. 정렬된 mesh vector
Fig. 3. Aligned mesh vector.

3. 유형분류 및 문자인식

앞서도 잠시 언급한 바와 같이 기존의 국내 연구를 살펴보면 한글의 경우에 모음의 형태, 받침의 유무 등에 의해 한글을 6가지 유형으로 분류하고, 이 분류된 유형내에서 다시 자소를 분리하여 자소별 인식을 하는 경우가 많았다. 그러나 이 방법은 한글만이 갖는 구조적 특성을 많이 이용한 경우로 여기서와 같이 한글과 한자를 동시에 인식할 경우에는 적용할 수 없다. 또 한글을 그러한 방식으로 유형분류할 경우 각 문자에 대해 기대되는 출력값, 즉 그 문자의 유형을 곧바로 알 수 있으므로 이는 많은 heuristic한 정보를 포함하고 있다고도 볼 수 있다. 그러나 이런 heuristic한 정보를 이용하는 것이 반드시 좋다고는 볼 수 없으며, 또 한자를 동시에 인식할 경우에는 문자가 그 어느 유형에 속할지 명확히 알 수 없다. 따라서 유형 분류를 위해 unsupervised learning을 하는 Self-origanizing feature map[8](이하 SOFM)을 이용하였다. 그리고 분류된 유형 내에서 문자를 인식할 때는 APC[9]를 이용하였다.

이러한 구조는 기존의 신경회로망 중 CombNet[7]과 유사하다. CombNet은 한자 인식을 위한 신경회로망으로서 약 3,000자에 달하는 방대한 한자를 인식하기 위해 먼저 유형분류하고 다시 각 유형내에서 인식을 하게 된다. 여기서는 대상 문자가 한자이기 때문에 mesh vector를 특징으로 사용하되 SOFM을 이용하여 몇 개의 유형으로 양자화 (quantize)시키게 된다. 그래서 SOFM을 vector quantizer로 사용하였고 각 유형내에서는 back-propagation(이하 BP) 학습 알고리듬을 사용하게 된다. 그래서 일본 표준 코드인 JIS의 제1수준 한자 2965자를 144개의 유형으로 분류하고 이를 다시 BP로 인식한 결과 99.5%의 인식률을 얻었다.

그러나 CombNet에는 다음과 같은 문제가 있다. 우선 유형 분류를 위해 SOFM을 vector quantizer로 사용하였는데 이 SOFM은 학습시에 출력층의 뉴런수를 임의로 지정해 주게 되어있다. 그러나 SOFM의 학습이란 패턴의 분포에 따라 가장 많은 분포를 갖는 유형들의 중앙으로 각 출력층 뉴런들의 가중치를 조절해 가는 과정이라 볼 수 있고 따라서 패턴의 분포와 출력층 뉴런의 갯수와는 밀접한 관계가 있는 것이다. 즉 패턴의 분포에 비해 출력층 뉴런의 갯수가 너무 적으면 패턴의 분포를 제대로 표현할 수 없고, 반대로 출력층 뉴런의 갯수가 너무 많으면 출력층 뉴런의 가중치들이 패턴 공간상에서 유사한 위치에 존재하게 되므로 오인식의 가능성이 커진다. 실제로 그림4에 나타낸 것과 같이 크게 세개의 유형으로 분류될 수 있는 패턴들을 임의로 생성하여 네개 혹은 그 이상의 출력층 뉴런을 갖는 SOFM으로 학습시켰더니 불필요한 임의의 위치나 두 유형의 중간 부분에 출력층 뉴런이 위치하여 오인식을 유발시키는 경우를 볼 수 있었다.

그래서 여기서는 처음에는 충분한 갯수의 출력층 뉴런으로 학습을 시키되 학습시키는 과정에서 패턴의 분포에 비해 불필요한 출력층 뉴런들을 제거시키면서 학습을 하는 방법을 제안하고자 한다. 즉 학습과정에서 임의의 주기로 각 출력층 뉴런들간의 위치

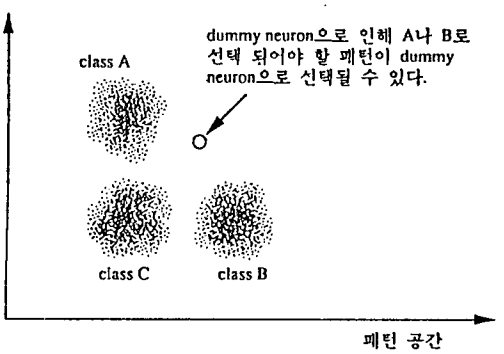

그림 4. Vector quantize시의 dummy neuron의 위치

Fig. 4. The location of a dummy neuron during the vector quantization.

를 검사하되 몇개의 뉴런들이 아주 가까운 거리를 두고 위치하게 되면 이 뉴런들을 하나로 묶고 다른 뉴런들은 지워버리는 것이다. 또 몇번의 학습동안 winner로 선택되는 횟수가 다른 뉴런들에 비해 월등히 적다면 그 뉴런도 학습에 영향을 받지 않는 뉴런으로 볼 수 있으므로 제거할 수 있다. 곧 새로운 패턴이 들어왔을 때 이 패턴이 어느 유형과 가장 유사한가를 결정하는 과정에서 발생할 수 있는 모호성을 없애자는 것이다. 이와 같은 학습 중간에 불필요한 뉴런을 제거할 경우 패턴들의 분포를 확실히 알 수 있으므로 가장 유사한 유형을 확실히 결정할 수 있을 뿐더러 제거된 뉴런들에 대해서는 학습을 시킬 필요가 없으므로 학습시간도 단축되는 효과를 나타낼 수 있다. 실제로 앞서 언급한 CombNet의 경우 학습시간이 많이 걸린다는 단점이 있다. 또 CombNet에서도 이렇게 유형분류 상에서 발생할 수 있는 오류를 복원하기 위해 어떤 문자가 들어왔을 때 하나의 유형만 결정하는게 아니라 유사도가 높은 몇개의 후보 유형을 결정하여 그 후보 유형들에 연결된 신경회로망들을 모두 활성화시킨다. 그리고 각 신경회로망 내에서 결정된 유사도와 그 유형과의 유사도를 결합, 비교하여 최종 문자를 결정하게 된다. 그러나 여기서 제안한 방법을 사용할 경우 유형분류시에 오인식의 확률이 적어진다는 장점이 있다.

또 CombNet에서는 문자 인식을 위해 BP를 사용하였지만 여기서는 APC를 사용하였다. APC를 사용함으로써 얻어지는 잇점으로는 우선 추가학습이 가능하다는 것이다. 즉 BP는 출력 패턴의 갯수가 임의로 한정되어 있기 때문에 추가학습을 할 수 없지만 APC에서는 학습과정에서 뉴런의 갯수를 필요에 따라 더 늘릴 수 있으므로 한번 학습이 완료된 후에도 새로운 출력 패턴을 생성할 수 있다. 그렇기 때문에 유형분류 신경회로망에서 발생할 수 있는 오류를 복원시켜줄 수 있다. 예를 들어 어떤 한 문자가 처음에는 유형 1로 학습되었으나 두번째 학습시에 약간의 위치 변화 혹은 잡음 등에 의해 유형 2로 판명되었을 경우 유형 2에도 그 문자에 대한 출력 뉴런을 하나 새로 생성시킴으로써 다음번에 발생할 수 있는 유형분류상의 오류를 복원시켜 줄 수 있는 것이다. 즉 CombNet에서는 어떤 한 문자에 대한 출력 뉴런은 오직 한 유형 내에만 존재할 수 있었으나 여기서는 APC를 사용하였기 때문에 출력 뉴런들이 각 유형 사이에서 중복되어 존재할 수 있는 것이다. 따라서 한 문자에 대해 유사한 몇개의 후보 유형을 선택할 필요없이 오직 하나의 유형만을 선택해도 충분히 학습을 받았다면 제대로 인식할 수 있다. 또 APC의 특징으로는 가중치를 변화시키지 않고 각 뉴런의 임계치만을 조절하고, 또 필요하면 새로운 뉴런을 생성하는 과정으로만 학습을 하기 때문에 학습속도가 빠르다는 것이다. 그러나 여기서는 이 APC가 각 유형의 갯수만큼 생기기 때문에 기존의 APC를 몇개의 모듈로 나눈 것이라 볼 수 있다. 기존의 APC는 존재하는 각 출력층의 모든 뉴런이 입력 패턴과 자신과의 거리를 비교하게 되는데, 이렇게 몇개의 모듈로 나누어 놓음으로써 비교해야 할 출력 뉴런의 갯수가 감소하게 되고 따라서 학습이나 인식의 시간이 더욱 단축된다고 볼 수 있다.

Ⅳ. 실험 및 결과

1. 실험 환경

본 실험 환경을 소개하자면 우선 PC상에서 Microtek의 300dpi의 해상도를 갖는 MSF-300C의 이미지 스캐너를 통해 문서를 입력 받는다. 이 이미지는 SUN spark 1+station으로 옮겨져 모든 문자 추출 및 정규화, 학습, 테스트 등의 모든 과정을 수행한다. 실험 대상 문자는 한글 사용 순위 상의 520자, 그리고 한자에 대해서는 문교부 지정 교육용 한자 중 중학교용 900자, 그리고 마침표를 포함하여 모두 1421 자이다. 그 밖의 문서에 대한 제한은 앞서 언급한 바와 같다.

2. 실험결과

우선 본 연구에서 제안한 수정된 SOFM의 효용성을 알아보기 위해서 패턴의 유형이 뚜렷이 구분되는 패턴들을 임의로 만들어 실험하였다. 즉 한변의 길이가 5인 정사각형의 네 꼭지점을 중심으로 하고 반

지름이 2인 원들을 만들어 그 원들의 내부에만 임의의 점 패턴을 분포시켰다. 기존의 SOFM으로 실험한 결과 처음에 약 10개의 출력층 뉴런을 주게 되면 약 2~3개의 뉴런이 정사각형의 네 꼭지점이 아닌 그들의 중간 부분에 임의로 생성되어 인식을 저해함을 알 수 있었다. 그러나 여기서 제안된 것으로 실험한 결과 초기의 출력층 뉴런 수를 약 30개까지 주어도 꼭지점 부근의 4개의 출력층 뉴런만 남기고 나머지는 다 지워버렸다. 그 이상의 뉴런 수로 실험해 보지는 않았으나 더 많은 뉴런을 주어도 결과는 동일한 것으로 예측된다.

다음에 실제로 패턴의 분포를 알 수 없는 경우, 즉 앞서 말한 1,421자로 실험을 하였다. 학습 데이타는 5세트의 폰트를 받아서 이를 평균한 값으로 100번 정도 학습을 시켰고 나머지 한 세트로 테스트하였다. 여기서 뉴런의 제거는 100번 학습 중 70번 학습이 끝난 후부터 시작하는 것으로 하였는데 이는 실험결과, 너무 일찍 뉴런의 제거를 시작하면 학습이 덜 된 상태에서 근접된 뉴런들이 지워지게 되어 너무나 적은 유형이 생기게 되고, 또 너무 늦게 뉴런의 제거를 시작하면 불필요한 뉴런들이 미처 다 지우지 못한다는 것을 알아냈고 이로부터 유추된 것이다. 우선 표1에 샘플링 주기, 즉 몇번의 학습 후에 불필요한 뉴런을 제거할 것인지를 결정하는 주기와 초기 출력층 뉴런수에 대해 분류한 유형의 갯수를 나타냈다. 여기서 알 수 있는 사실은 그 주기 및 초기 뉴런수의 많고 적음에는 별 영향이 없이 대체로 4~7개 정도의 유형으로 분류하였다는 것이고, 다만 여기서 4~7개를 결정하는 요인은 뉴런의 제거를 시작할 당시의 패턴의 분포라고 예측할 수 있다. 또 여기서 4~7개로 분류하였다는 사실은 대체로 패턴의 분포가 명확하지는 않지만 약 4~7개로 분류할 수 있음을 시사하는 결과이다. 그래서 실제로 처음부터 출력층 뉴런을 제거하지 않고 여러개로 변화시켜 가며 인식한 결과를 그림5에 나타냈다. 여기서 출력층 뉴런이 약 10개 이하일 때는 비교적 높은 인식률을 보였지만 그 보다 많아지게 되면 인식률이 상당히 저하됨을 알 수 있다. 즉 이 패턴의 분포는 10개 이하로 분류하는 것이 타당하다는 결과를 나타내고 있고 따라서 우리가 구한 4~7개라는 결과에도 부합되는 것이라 할 수 있다. 또 여기서 주목할 만한 것은 유형의 갯수가 같다면 인식률도 동일한 결과를 나타낸다는 것이다. 즉 처음부터 6개의 출력층 뉴런수를 갖도록 하고 뉴런의 제거를 하지않은 결과와 몇개를 주던 나중에 6개의 출력층 뉴런만 남기고 난 후의 결과를 비교하면 그들간의 인식률이 같다는 것이다.

표 1. 샘플링 주기 및 초기 뉴런수에 대한 유형 갯수

Table 1. No. of types to each sampling frequency and No. of types to each initial No. neurons.

sampling freq.	#of type	initial # of neurons	#of type
2	4	10	4
3	4	15	5
4	5	20	6
5	4	25	6
6	4	30	7
7	6	35	4
8	6	40	6
10	6	45	6

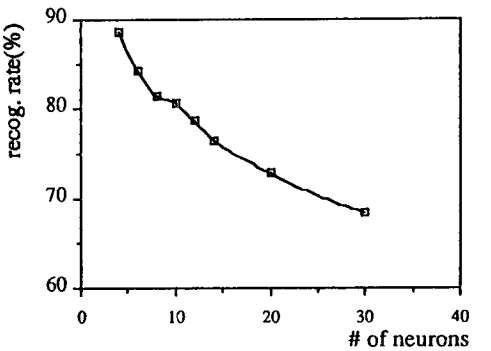

그림 5. 출력층 뉴런 수와 유형분류율과의 관계

Fig. 5. Relationship between No. of output neurons and type classification rate.

이는 여기서 제안한 방법이 패턴의 분포를 적절히 고려하여 꼭 필요한 곳에 출력층 뉴런을 위치시키도록 제대로 학습되었음을 보여준다. 그리고 최종적으로는 6개의 유형으로 분류된 결과를 선택하였는데 이는 4 또는 6개의 유형으로 분류하는 경우가 가장 많았고 이 중 가능한 한 많은 유형으로 분류하는 것이 전체 시스템의 속도 향상에 기여할 수 있는 방법이기 때문이다.

이때 유형분류율은 약 84.24%로 이 결과는 그리 좋은 결과라 볼 수 없을 수도 있지만 원래 SOFM 자체가 미학습에 대한 일반화의 능력이 다른 모델에 비해 많이 떨어지기 때문에 나타난 결과라 볼 수 있다. 그러나 여기서는 한 문자가 하나의 유형에만 존재하는 것이 아니라 학습에 따라 여러 유형에 중복되어 존재할 수 있도록 하였기 때문에 그림6에 나타

낸 바와 같이 유형분류율은 추가학습을 할수록 점차 향상되는 결과를 볼 수 있었다. 그림6에는 명조체에 대한 유형분류율과 인식률을 나타내었으며, 최종 유형분류율과 인식률은 각각 96.41%, 95.29% 였다. 여기서 한가지 주목할 사항은 총 인식률이 95.29%로 이는 유형분류율에서의 오류를 포함한 수치이므로 이 오류를 제외한, 즉 APC만의 인식률을 환산해 보면 약 98.83%가 된다. 이는 기존의 APC의 약 97%보다 높은 결과이다. 물론 기존의 그 결과는 명조체 한글 990자에 대해 4세트로 학습한 결과로서 여기서도 4세트 학습 후의 APC만의 인식률을 계산해 보면 약 98.96%로 역시 향상된 결과이다. 이렇게 인식 대상 문자수가 990자에서 1420자로 증가했음에도 불구하고 인식률이 증가된 것은 APC를 모듈화 함으로써 발생된 결과라고 볼 수 있다. 그리고 4세트 학습 후의 인식률 98.96%에서 한 세트 더 학습한 후 98.83%로 약간 떨어진 것은 APC가 추가학습을 하면서 각 뉴런이 임계치를 감소시키기 때문에 기존에 학습한 것을 잊어버릴 수 있음을 나타내는 증거이다. 따라서 학습과정에서 학습한 데이타를 잊어버리지 않도록 하는 과정, 즉 예를 들면 이전에 학습한 것을 다시 한번 학습시키는 방법이나 혹은 그 외에 좀 더 체계적인 방법이 요구된다.

그리고 그림7에 추가학습에 따라 생성된 중간층 뉴런의 갯수를 나타내었다. 일부 구간에서 생성된 중간층 뉴런 갯수가 증가된 경우가 있긴 하지만 대체적으로 학습을 진행함에 따라 생성되는 뉴런 수가 감소된 것을 알 수 있었다. 명조체 한글 한자 총 1,421자를 6세트 학습시키는데 생성된 총 중간층 뉴런수는 약 2,319개로 이는 인식 대상 문자의 약 1.63배에 달하는 수치이다. 또 인식하는데 걸린 시간은 점차 학습을 할수록 증가되어 최종적인 인식시간은 163.80초가 소요되었다. 이는 학습을 할수록 중간층 뉴런이 추가되어 인식하는데 비교해야 할 뉴런수가 많아졌기 때문이다. 이 결과는 약 1초에 약 8.67자를 인식한다는 것으로 기존의 APC의 약 3자에 비해 많이 향상되었으며, 그 이유는 유형분류를 통해 APC를 몇개의 모듈로 분리함으로 인한 결과이다.

다음에는 새로운 폰트에 대한 학습 가능성에 대한 실험을 하였다. 그림8에는 앞서 한 명조체 1,421 자에 대한 5세트의 학습을 마친 후 다시 고딕체 한글 520자에 대한 추가학습을 한 결과를 나타냈다. 여기서도 역시 처음에는 86.54%이었으나 학습을 추가함에 따라 인식률이 증가되어 4번의 학습 후에 인식률이 93.85%까지 증가하였다. 즉 한번 명조체에 대한

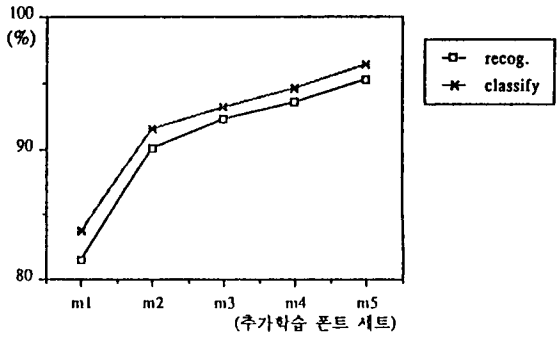

그림 6. 명조체 총 인식률 및 유형 분류율
Fig. 6. Final recognition rates and type classification rates for Myungjo font.

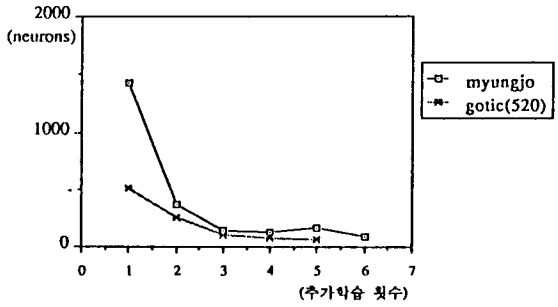

그림 7. 추가학습에 따라 생성된 뉴런수
Fig. 7. No. of neurons increased according to additive learnings.

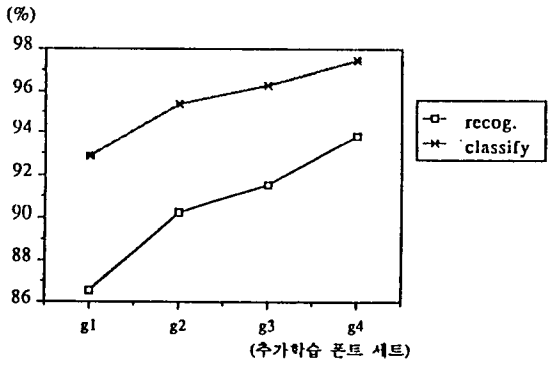

그림 8. 추각학습된 고딕체 인식률 및 유형 분류류형
Fig. 8. The final recognition and type classification rates for additively learned Gothic font.

학습을 한 상태이기 때문에 고딕체에 대한 추가학습에서는 명조체 때의 인식률보다 약간 높은 인식률을 나타낸 것으로 볼 수 있다. 그리고 역시 추가된 중간층 뉴런수를 그림7과 같이 나타냈는데 여기서도 역시 학습이 진행됨에 따라 추가되는 뉴런수가 감소함을 알 수 있었다. 최종적으로 추가된 뉴런수는 인식하고자 하는 520자의 약 1.94배 정도였다. 이는 앞서의 1.64배 보다 많은 결과인데 우선은 명조체와 고딕체가 mesh vector 특징상에서 많은 차이를 보이고 있다는 것을 알 수 있고, 또 하나의 이유는 APC의 학습 알고리듬상 첫번째 학습시에 생성된 뉴런들이 가장 큰 임계치, 즉 인식반경을 가질 수 있기 때문이다. 즉 처음에 생성된 뉴런들은 유사한 뉴런들이 많지않기 때문에 넓은 인식 반경을 갖고 있다가 학습을 통해 적절한 반경으로 줄여나가지만, 유사한 뉴런이 많은 상태에서 새로이 추가된 문자는 좁은 인식 반경을 가질 수 밖에 없기 때문에 그에 대응되는 만큼의 많은 추가 뉴런을 필요로 하기 때문인 것으로 분석된다. 그리고 인식하는데 걸리는 시간도 1초에 약 6.29자를 인식하여 앞서의 인식 시간보다 더 느린 결과인데 이는 앞의 실험후에 추가학습을 통해 새로운 뉴런이 더 추가되어 비교해야 할 뉴런이 더 많아졌기 때문이다. 따라서 추가학습에도 어떤 한계가 있을 것이라는 예측을 할 수 있지만 어쨌든 이 실험을 통해 새로운 폰트를 학습할 수 있음을 알 수 있었고, 따라서 어떤 새로운 폰트라도 추가할 수 있다는 결론을 얻었다.

다음은 실제 문서를 통한 인식 실험을 하였다. 사용한 문서는 총 717자 중 인식 대상인 1,421자에 소속되지 않는 12자를 제외하고 나머지 705자에 대한 결과를 알아보았다. 이 중 한글은 508자 이었고 한자는 197자 이었는데 전체 인식률은 약 94.3% 정도로 총 1,421자로 테스트한 결과에 비해 다소 떨어진 결과를 나타내었다. 이는 1,421자로 테스트한 실험에서는 학습한 문자들이 모두 실제 문서에 비해 비교적 안정된 문자, 즉 위치이동 등이 별로 없고 문자들 간에 접촉이 없는 문서에서 추출된 문자들이었기 때문에 약간 더 좋은 결과를 보인 것이다. 그러나 이 문제점은 실제 문서를 통해 학습을 하면 충분히 해결될 수 있는 문제이다. 그래서 이를 고려하여 실제 문서로부터 추출된 문자를 학습할 수 있도록 구현하였다. 또 총 717자를 인식하는데 걸린 시간은 72.45초로 1초에 약 9.89자를 인식한 결과이다. 이는 중간층 뉴런을 위치시킬 때 한글을 앞에 놓고 뒤에 한자를 놓고, 먼저 한글로 판명난 문자는 그 뒤의 뉴런과는 비교를 하지 않는 방법을 썼기 때문에 일반적인 문서에서는 실제 문자인식 시스템 보다 빠른 속도를 보이게 되는 것이다.

실제 문서에서 오인식의 사례를 보면 한글의 경우는 주로 모음의 오인식이 많았다. 즉, 'ㅓ'를 'ㅣ'로, 'ㅡ'을 'ㅜ'로, 'ㅡ'를 'ㅗ'로 오인식하는 경우가 대부분이었고, 자음의 경우는 'ㄱ'을 'ㄲ'으로 오인식하는 경우가 있었다. 모음의 경우는 'ㅣ', 'ㅏ', 'ㅓ'의 오인식이 가장 많았고, 다음으로는 'ㅡ', 'ㅗ', 'ㅜ'의 오인식이 많았다. 이 경우는 수평, 혹은 수직 모음 부근에 있는 하나 혹은 두획에 의한 오인식으로 볼 수 있으며, 따라서 유형분류시에 구한 히스토그램 상에서 최대값을 갖는 부분을 중심으로 그 부근의 가중치를 높일 필요가 있다고 본다. 자음의 경우는 'ㅌ'과 'ㄹ', 'ㅇ'과 'ㅁ', 'ㅂ' 사이에서 많은 오류가 발생하였다. 이는 mesh vector의 해상도를 좀 더 높게 조정하면 해결될 수 있으리라 본다.

그리고 한자의 경우는 문자가 복잡해질 경우에 주로 오인식이 많았고 간단한 문자의 경우는 '字'를 '宇'로, '寸'을 '才'로 오인식하는 경우가 있었다. 이러한 오류도 역시 한글 자음의 경우와 같이 40×40 크기의 문자를 40차원의 mesh vector로 줄였기 때문에 발생한 것으로 볼 수 있다. 또 복잡한 한자의 오인식도 역시 동일한 오류이며 이러한 오인식을 없애려면 좀 더 고해상도의 mesh vector를 이용해야 할 것이다. 그 밖에 한글과 한자를 오인식한 경우는 '포'를 '空'으로, '分'을 '於'으로 인식하는 경우가 있었는데, 이것도 역시 mesh vector가 대략적인 문자의 형태를 인식하는 데는 좋으나 아주 세세한 특징까지 요구되는 문자에 있어서는 적당하지 않다는 것을 보여주고 있다. 결국 mesh vector의 해상도를 높이거나 아니면 유형 분류를 위해 mesh vector를 사용하고 인식을 위해서는 다른 특징, 즉 어떤 특징점이나 그 이외의 세부 특징을 사용하는 것도 하나의 방법이라 할 수 있다.

V. 결론 및 고찰

본 연구에서는 한글과 한자를 동시에 인식할 수있는 문서 인식 시스템을 구현하였다. 아직까지 국내에 한글과 한자를 동시에 인식하는 시스템은 전무한 실정인데 이는 기존의 국내 문자 인식 연구가 너무나 한글의 구조적 특징만을 고려하여 이를 많이 이용했으므로 다른 나라의 문자를 동시에 인식할 수가 없었기 때문이었다. 그래서 한글만의 구조적 정보를 전혀 고려하지 않는 방법을 사용하여 한글과 한자를 동시에 인식할 수 있는 시스템을 구현하였다. 또 다중

활자체 문서 인식에 대한 연구도 많이 있었는데 여기서도 새로운 활자체에 대한 추가학습을 할 수 있음을 보였으며 이를 확장하면 다중 활자체에 대한 인식도 가능하다. 또 활자체 뿐만 아니라 영문이나 일어 등 어떤 다른 나라의 문자라 하더라도 문자단위로 추출하는 부분만 보완하면 인식이 가능하다. 이처럼 한글과 다른 나라의 문자를 동시에 인식하는 시스템을 처음으로 제안하였다는 것이 본 연구의 의의라 할 수 있다.

기존의 국내 연구에서는 한글을 구조적 특성에 따라 6가지로 분류하는 경우가 많은데 본 논문에서는 unsupervised learning을 통해 신경회로망이 스스로 한글과 한자에 대해 유사도를 결정하여 6가지 유형으로 분류하는 결과를 얻었다. 이는 본 실험과 같이 기대되는 출력값을 모를 경우에는 SOFM과 같이 unsupervised learning을 사용하는 것이 타당함을 보인것이다. 또 vector quantizer로서의 SOFM 모델의 비효율성을 지적하고 이를 수정함으로써 그 효용성을 보였다. 문자 인식 뿐 아니라 모든 패턴 인식 분야에서 대규모의 패턴들을 구분하기 위해서는 모듈화가 필연적이라 할 수 있고, 이러한 모듈화를 위해서는 여기서 제안한 수정된 SOFM이 많이 응용될 수 있을 것이다.

그리고 문자 인식부에서 APC를 사용하였고 결과적으로 APC를 모듈화한 것이라 볼 수 있다. 이렇게 APC를 모듈화함으로써 학습 및 인식 시간이 기존의 APC보다 더욱 빨라졌고 전체적으로 1초에 약 8~9자를 인식할 수 있는 시스템을 구현하였다. 이는 기존의 다른 연구에 비해 매우 빠른 편이라 볼 수 있고 학습 시간도 매우 빠르다.

그 밖에 문자의 유형분류를 위해 새로운 특징을 사용하였다. 이 특징은 문자라는 고차원의 정보로부터 필요한 몇 차원의 정보만을 추출한 것으로서 이렇게 저차원의 정보를 사용함으로써 학습 및 인식의 시간이 많이 단축되었음을 알 수 있었다. 이러한 특징은 많은 변형이 있을 수 있겠지만 문자 인식 뿐 아니라 모든 1차원 신호 처리 분야에서도 응용될 수 있을것으로 기대된다.

본 연구의 실용화에 대한 가능성을 살펴본다면 우선 적어도 한글 990자, 한자 1,800자 이상은 인식할 수 있어야 한다는 조건을 들 수 있다. 그러나 기존의 CombNet 자체가 한자 약 3,000자에 대해 성공적인 결과를 얻었듯 문자수가 약 3,000자까지 증가해도 인식할 수 있음은 예견할 수 있다. 즉 문자 수가 증가해도 여러 유형으로 분류되기 때문에 실제적으로 각 유형내에서 증가되는 문자 수는 그리 많지 않을 것이며 따라서 별 문제가 되지 않을 것이다. 다만 문제가 되는 것은 APC상에서 증가되는 뉴런의 갯수이다. 실험에서도 알 수 있었듯 명조체에 대한 증가분보다 고딕체 추가학습시의 뉴런 증가분이 더 많았다는 사실은 새로운 활자체 및 문자로의 확장 과정에서 문제가 생길 수 있음을 시사하고 있다. 그러나 인간의 뇌 구조 자체가 무수한 뉴런으로 구성되어 있듯 학습을 함에 따라 뉴런이 증가되는 것은 당연한 현상이라 볼 수 있다.

앞으로의 연구 방향은 우선 APC의 보완이다. 실험결과에서도 알 수 있었듯이 APC는 기존에 학습한 것을 잊어버릴 수 있기 때문에 추가학습에 따라 인식률이 약간 떨어질 수 있다. 이를 보완하는 것이 필요하고 또 중간층 뉴런의 갯수가 너무 많아지는 것을 막을 수 있는 방법도 필요하다. 즉 새로 생성되는 뉴런은 기존의 패턴 공간 상에서 다른, 뉴런과 겹쳐지지 않도록 작은 반경을 갖는 hypersphere를 생성하게 된다. 따라서 학습을 할수록 많은 뉴런이 생기게 되는 현상을 볼 수 있었으며 이에 대한 보완이 요구되어야만 한다. 그 밖에 전체적인 시스템상의 요구사항이라면 문자 추출시에 한글과 한자만 가능한 상태인데 문자의 폭이 다양한 다른 문자, 즉 영문 등도 동시에 추출이 가능하도록 하는 과정이 남아있다.

參 考 文 獻

[1] 권재욱, 조성배, 김진형 : "신경망 기법을 이용한 다중 크기 및 다중 활자체 한글 문서의 인식," 제3회 영상 처리 및 이해에 관한 워크샵발표 논문집, pp. 129－136, 1991년 2월.

[2] 도정인 : "한글 문서 인식 시스템의 개발,"정보과학회지, 제9권 제1호, pp. 22－32, 1991년 2월.

[3] 한글학회 : 우리말 큰사전, 을유 문화사, 1957년.

[4] 여진경, 이우일, 정호선 : "IDMLP를 이용한 한자 인식에 관한 연구," 전자공학회 논문지, 제28권 B편 제10호, pp. 37－43, 1991년 10월.

[5] 이현표, 양순성, 황교철 : "한글 문서에서의 낱자 분리 알고리즘," 한글 및 한국어 정보처리 학술발표 논문집, pp. 203－208, 1989년.

[6] 김상우, 전윤호, 최종호 : "신경회로망을 이용한 인쇄체 한글 문자의 인식," 전자공학회 논문지, 제27권 제2호, pp. 65－71, 1990년 2월.

[7] Iwata, A., Tohma, T., Matsuo, H. and Suzumura, N.: "A Large Scale Neural Metwork "CombNet" and its Application to Chinese Character Recognition," *Proc.*

of Inter, Neural Network Conf., pp. 83-86, 90 Paris, 1990.

[8] Kohonen T.: "The Self-Orginizing Map," *Proc. of the IEEE*, vol. 78, no. 9, pp. 1464-1479, Sep. 1990.

[9] 박영환, 방승양 : "패턴 분류용 신경회로망 APC의 설계 및 구현," 제2회 신경회로망연구회 학술대회 발표논문집, pp. 30－35, 1991년 6월.

著 者 紹 介

金 祐 成(正會員)

1967年 9月 14日生. 1990年 한국과학기술원 과학기술대학 전산학과 학사. 1992年 포항공과대학 전자계산학과 석사. 현재 포스데이타(주) 근무. 주관심분야는 신경회로망을 이용한 문자인식 및 뉴러컴퓨팅 등임.

方 勝 楊(正會員)

1942年 5月 8日生. 1966年 일본 Kyoto대학 전기공학 학사. 1969年 서울대학교 전기공학 석사. 1974年 University of Texas 전산학 박사. 1981年～1984年 한국전자기술연구소 실장 및 부장. 1984年～1986年 (주)유니온 시스템 연구소 소장 역임. 1986年～현재 포항공대과학 전자계산학과 교수. 주관심분야는 신경회로망 이론 및 패턴인식, 뉴러컴퓨팅, man-machine interface 등임.