논문 98-01-10

신경망 학습 코드에 따른 오프라인 필기체 한글 인식률 비교

Comparisons of Recognition Rates for the Off-line Handwritten Hangul using Learning Codes based on Neural Network

金 美 英*, 趙 鏞 範**
(Mi-Young Kim* and Yong-Beom Cho**)

요 약

본 논문은 필기체 한글의 특징을 추출한 후 이를 신경망을 이용하여 인식하였다. 한글의 특징 추출을 위해 5×5 윈도우 방법을 사용하였는데, 이는 3×3 윈도우 방법을 수정한 것이다. 추출된 특징을 이진화 코드로 변환하여 신경망의 입력으로 사용하며, 백프로퍼게이션 알고리즘으로 학습시켰다. 수직 모음, 수평 모음, 자음 인식을 위한 3개의 신경망을 각각 구성하였고, 결과를 비교하기 위하여 3가지 학습 방법을 사용하였다. 3가지 학습 방법은 고정 코드 방법, 학습 코드 방법 I, 학습 코드 방법 II이고 학습 코드 방법 II가 가장 좋은 결과를 보였다. 이 경우 수직 모음과 수평 모음은 100%의 인식률을, 자음은 93.75%의 인식 결과를 보였다.

Abstract

This paper described the recognition of the Off-line handwritten Hangul based on neural network using a feature extraction method. Features of Hangul can be extracted by a 5×5 window method which is the modified 3×3 mask method. These features are coded to binary patterns in order to use neural network's inputs efficiently. Hangul character is recognized by the consonant, the vertical vowel, and the horizontal vowel, separately. In order to verify the recognition rate, three different coding methods were used for neural networks. Three methods were the fixed-code method, the learned-code I method, and the learned-code II method. The result was shown that the learned-code II method was the best among three methods. The result of the learned-code II method was shown 100% recognition rate for the vertical vowel, 100% for the horizontal vowel, and 98.33% for the learned consonants and 93.75% for the new consonants.

I. 서 론[1]

컴퓨터의 발달과 함께 많은 양의 데이터를 짧은 시간 내에 처리할 수 있게 되었으나, 사람이 쉽게 행할

* 現代電子産業株式會社
(Hyundai Electronics Industries Co., Ltd.)
** 建國大學校 電子工學科
(Dept. of Electronic Eng., Kon-Kuk Univ.)

※본 논문은 1995년도 건국대학교 산업기술연구원 산학협동 연구과제 연구비에 의하여 연구되었음.
接受日: 1998年4月16日, 修正完了日: 1998年8月24日

수 있는 패턴 인식과 같은 지능적인 처리는 아직 힘든 실정이다. 컴퓨터가 이러한 인간의 인식 능력을 갖게 된다면, 많은 수작업의 자동화 뿐만 아니라 인간과의 자연스러운 연결이 가능해지기 때문에, 많은 연구자들이 이를 실현하기 위해 노력하고 있다. 온라인 한글 인식에서는 필기시에 여러 정보를 얻을 수 있기 때문에 오래 전부터 많은 방법이 제안되어 왔고, 특히 1990년대에 들어서면서 신경망을 이용한 연구가 활발히 진행되었다[1-4]. 또한 오프라인 인쇄체 한글 인식에서도 신경망을 효과적으로 이용하였다[5-6]. 그러나 오프라인 한글 인식에서는 입력 장치를 통해 얻어진 데이터를 사용하므로 온라인에서 얻을 수 있는 패턴의 정보가 손실되며 이로 인해 전처리 과정을 필요로 하게 된다. 또한 전처리 과정에서 부분적인 에러를 동반하게 되므로 온라인 한글 인식에 비해 어려움이 훨씬 크다.

기본 자소의 조합으로 이루어지는 한글은 그 조합된 수가 14,000여자에 이르기 때문에 글자 단위로 인식하는 경우는 인식 패턴의 수가 너무 방대하여 인식의 어려움이 있다. 따라서 자소 단위로 인식하는 방법이 효과적이나 이 경우에는 자소들 간의 접촉점 문제를 해결해야 하며, 인식의 결과가 자소 분리 방법에 크게 의존하게 된다. 접촉점에 따른 자소 분리에 관한 연구도 발표되고 있으나[7], 이러한 문제들이 오프라인 필기체 한글 인식을 어렵게 만들고 있다. 본 논문에서는 학습코드에 대한 인식률을 조사하기 위해 자소 사이의 접촉점이 발생하지 않도록 입력 패턴에 제약점을 두었다.

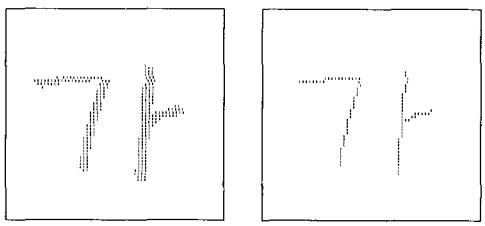
오프라인 필기체 한글 인식에 있어서 입력 패턴을 그대로 신경망을 이용하여 학습하는 경우, 문자의 위치나 크기, 그리고 필기자의 개성 등에 민감할 뿐만 아니라, 입력 패턴의 전체 픽셀(pixel)값을 신경망의 입력으로 사용하게 되므로 방대한 양의 입력을 가지게 되어 학습에 어려움이 따른다. 본 논문에서는 전처리 과정을 통하여 특징을 추출하고 추출된 특징을 이진화 코드로 변환하여 신경망의 입력으로 사용함으로써 적은 양의 입력으로 필기체 한글을 인식하는 장점을 가진다. 또한 특징을 추출하기 때문에 입력 패턴의 크기에는 별로 영향을 받지 않게 된다.

본 논문의 내용은 먼저 한글의 구조적 특성을 이용하여 6가지 형식으로 분류하고 5×5 윈도우를 이용하여 분리된 자소의 특징들을 추출한다. 특징으로는 끝점과 그것의 방향 성분, 분기점, 그리고 굴곡점이 추출되며, 이것을 신경망의 입력으로 사용하기 위해 이진화 코드로 변환한다. 자소로 분리할 때 알 수 있었던 자소 위치를 이용하여, 현재 분리되는 자소가 자음인지 수직 모음인지 또는 수평 모음인지 알 수 있게 된다. 자음과 모음의 모든 자소를 학습 패턴으로 사용하여 단일 신경망을 구성하는 것보다는 학습 패턴의 수가 적을수록 좋은 학습 효과를 보이므로, 보다 적은 패턴 수를 갖도록 3개의 신경망을 각각 설계한다. 수직 모음(ㅏ, ㅑ, ㅓ, ㅕ, ㅣ, ㅐ, ㅒ, ㅔ, ㅖ)을 인식하는 수직 모음 신경망(Vv_BP), 수평 모음(ㅗ, ㅛ, ㅡ, ㅜ, ㅠ) 인식의 신경망(Vh_BP), 초성 및 종성 자음을 인식하는 자음 인식의 신경망(C_BP)으로 나눈다.

II. 전처리 및 자소 분리

II-1. 전처리 과정

필기체 한글 인식을 위한 입력 패턴의 크기는 64×64 픽셀이며, 전처리 과정으로는 이치화(binarization)와 세선화(thinning)의 과정을 거친다. 그림 1(a)는 이치화된 패턴이며 그림 1(b)는 세선화 결과를 보여준다. 세선화 과정은 Blum에 의해 제안되었던 medial axis transformation(MAT) 방법을 사용하였다[8].



(a) (b)

그림 1. (a) 이치화 패턴 (b) 세선화 패턴

Fig. 1. (a) binarization pattern (b) thinning pattern

II-2. 한글 구조와 자소 분리

한글은 기본 자소의 조합으로 구성된다. 기본 자소의 기준은 연구 방향에 따라 조금씩 차이가 있으나, 본 논문에서는 자음 14개(ㄱ,ㄴ,ㄷ,ㄹ,ㅁ,ㅂ,ㅅ,ㅇ,ㅈ,ㅊ, ㅋ,ㅌ,ㅍ,ㅎ)와 모음 14개(ㅏ,ㅑ,ㅓ,ㅕ,ㅗ,ㅛ,ㅜ,ㅠ,ㅡ,ㅣ, ㅐ,ㅒ,ㅔ,ㅖ)로 정의한다. 한글은 자소의 조합 구성에 따라 6가지 형식으로 나누어 질 수 있으며[9] 그림 2와 같다. 여기서 Cf는 초성 자음, Vh는 수평모음, Vv는 수직 모음, Cl은 종성 자음을 나타낸다. 수직 모음과 수평 모음의 존재 유, 무로 다음의 6가지 형식으로 분류하고, 그 위치에 따라 자음과 모음이 구분되어질 수 있으므로 자소 단위로 분리할 수 있다.

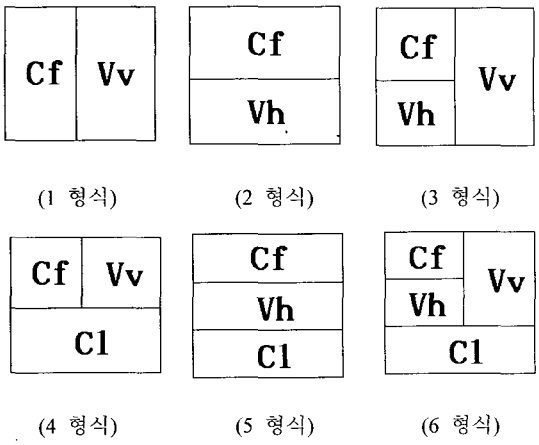


그림 2. 한글의 6가지 형식

Fig. 2. The six forms of Hangul

수직 모음과 수평 모음의 존재 여부와 종성의 존재 여부로 6가지 형식이 구분되어진다. 그러나 필기체 한글뿐만 아니라 인쇄체의 경우에서도 초성의 아래 긴 수평 획이나 종성의 위 부분의 긴 수평획 등이 수평 모음과 구별되기 어렵기 때문에 기존의 방법[9]에서의 분류는 큰 분류 에러를 발생시킬 수 있다. 따라서 본 논문에서는 먼저 수직 모음의 존재 유, 무를 알아낸 후 초성을 분리해 내고 이를 제거한다. 초성의 경우는 6가지 한글의 형식 중 어느 형식에서나 존재하므로 초성의 기본적인 위치를 이용하여 이를 분리한다. 수직 모음의 존재에 따라 (1,3,4,6)형식과 (2,5)형식으로 분류되며 (2,5)형식은 반드시 수평 모음이 존재하므로 수평 모음의 위치 정보를 이용하여 수평 모음을 찾고 이를 분리한다. 수평 모음 분리 후 종성의 존재 여부로 2형식과 5형식을 분류한다. (1,3,4,6)형식의 경우 초성 분리 후 1형식은 자소의 분리가 모두 이루어진 경우이므로 입력된 패턴에서 어떤 픽셀도 남아 있지 않게 되고, 4형식의 경우는 종성이 존재하고 (3,6)형식의 경우는 수평 모음이 존재한다. 종성과 수평 모음의 구분은 입력 초기에는 결정되어 질 수 없었으나 수직 모음과 초성 자음이 분리, 제거된 후에는 수직 모음의 X축의 값을 이용하여 분류할 수 있다. 이 경우, 입력 패턴의 수평축을 X축으로 수직 축은 Y축으로 정의한다.

수직 모음의 X축상의 값 x에서 가장 큰 위치값을 max(x), Y축상의 값 y에서 가장 큰 값을 max(y)라 하고, 64×64 크기의 입력 패턴을 Pattern[64][64]라고 정의하면 수평 모음과 종성 자음의 구분은 다음과 같다.

$$\text{if } \left(\sum_{x=0}^{\max(x)} \sum_{y=0}^{\max(y)} Pattern[x][y] \right) > threshold \text{ then 입력}$$

패턴 : 3, 6 형식

else 입력 패턴 : 4 형식

(식 1)

수평 모음을 가지는 (3,6)형식의 경우에서, 또다시 종성이 존재한다면 이는 6형식으로 구분되어지고 존재하지 않는 경우는 3형식임을 알 수 있다. 이러한 방법을 통해 6가지 형식을 분류하여 자소를 각각 분리한다.

III. 한글의 특징 추출

한글의 자소들은 시작과 끝 그리고 분기와 굴곡의 다양한 관계로 이루어져 있다. 본 논문에서 자소 인식을 위해 사용한 특징들로는 끝점(End-point), 분기점(Branch-point) 그리고 굴곡점(Bent-point)을 사용하였고 자소 분리시 이를 추출하고 각각에 대하여 적절히 특징 코드를 할당하였다. 특징 추출에는 그림 3(a)과 같

은 5×5 윈도우를 이용한다. 3×3 윈도우를 이용하는 경우는 다음에 이동할 위치 결정이나 특징 추출의 결과가 바로 이웃 픽셀 분포에 영향을 받기 때문에 세선화 결과에 크게 의존하게 된다. 그러므로 좀 더 좋은 결과를 보이는 세선화 알고리즘의 사용이 필요하게 된다. 7×7 윈도우의 경우는 이웃 픽셀의 수가 많아 그 중 어느 하나의 픽셀이 '1'로 존재할 경우의 수가 많아질 뿐만 아니라 분리하고자 하는 자소가 이웃 자소에 너무 근접하게 있는 경우는 윈도우 범위가 넓기 때문에 근접된 이웃 자소를 구분하기가 어려워진다. 따라서 본 연구에서는 5×5 윈도우을 이용, sur[0] ~ sur[15]까지 16개의 이웃 픽셀(neighbor pixel)을 정의하여 특징을 추출한다.

sur[14]	sur[15]	sur[0]	sur[1]	sur[2]
sur[13]				sur[3]
sur[12]		**p(x,y)**		sur[4]
sur[11]				sur[5]
sur[10]	sur[9]	sur[8]	sur[7]	sur[6]

(a)

14	⑮	0	①	2
⑬				③
12		**p(x,y)**		4
⑪				⑤
10	⑨	8	⑦	6

(b)

그림 3. (a) 5×5 윈도우 (b) 세선화 결과를 보완하기 위한 5×5 윈도우

Fig. 3. (a) 5×5 window (b) 5×5 window for compensating the thinning result

전처리 과정에서 사용된 세선화 알고리즘은 단 하나의 픽셀로 연결되어지지 못하는 단점을 가지고 있다. 본 논문에서는 이러한 세선화 결과를 보완하기 위해서, 그림 3(b)에서 정의된 원 숫자의 픽셀이 '1'인 경우 그것의 이웃하는 픽셀이 '1'의 값을 갖으면 '0'으로 만들어준다. 예를 들어, sur[1]의 픽셀이 '1'일 때 sur[0]나 sur[2]중의 어느 하나가 '1'이면 sur[1]의 픽셀을 '0'으로 대치시킨다. 또 윈도우의 중심의 위치가 p(x,y)일 때, 다음의 윈도우 중심의 위치는 sur[0] ~ sur[15] 중에서 '1'의 값을 갖는 위치가 되며, 다음 위치로 중심을 옮기면서 현재 위치의 5×5 윈도우 안의 모든 픽셀을 '0'으로 만들어 준다. 따라서 지나온 자리의 픽셀 값들은 모두 제거된다.

III-1. 끝점 특징과 특징 코드 할당

온라인 한글 인식에서 모든 끝점은 바로 인식될 수 있으나, 오프라인의 경우 그것의 정의를 내려주어야 한다. 그림 3(a)에서 나타난 16개의 이웃 픽셀들의 합을 sum(x,y)로 나타내고, 끝점의 조건은 다음과 같이 정의할 수 있다. 먼저 시작하는 끝점인 Start point의 조건은 sum(x,y)이 1이고, 끝나는 점인 End point의 경우는 이미 지나온 자리가 제거됨으로 인해 sum(x,y)이 0이다.

Start point : p(x,y) = 1 and sum(x,y) = 1 (식 2)

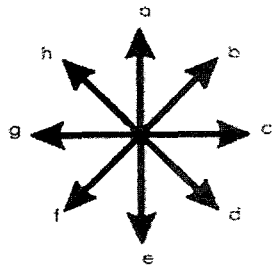
이때, $\text{sum}(x,y) = \sum_{i=0}^{15} sur(i)$

End point : p(x,y) = 1 and sum(x,y) = 0 (식 3)

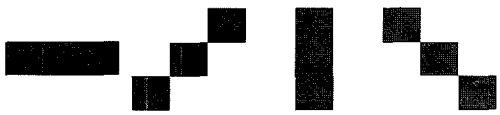
끝점이 그림 3에서 보인 5×5 윈도우의 중심 p(x,y)에 있었을 때, 이웃 픽셀 sur[0]에서 sur[15]까지 어느 하나가 '1'일 수 있는 16개의 끝점 패턴이 존재한다. 이것을 그림 4(a)에서 정의된 8방향의 방향성분으로 나눈다. 먼저 sur[15], sur[0], sur[1]은 'a'방향, sur[2]는 'b'방향, sur[3], sur[4], sur[5]는 'c'방향, sur[6]은 'd'방향, sur[7], sur[8], sur[9]는 'e'방향, sur[10]은 'f'방

향, sur[11], sur[12], sur[13]은 'g'방향, sur[14]는 'h'방향으로 정의한다. 이러한 8방향은 그림 4(b)와 같이 4개의 획을 나타낸다. 즉 수직획(a, e 방향)과 수평획(c, g 방향), 오른 방향 사선획(b, f 방향)과 왼 방향 사선획(d, h 방향)으로 나뉘고 각각을 2비트(bits)로 표현할 수 있으며, 다음과 같이 코딩한다.

a, e(10), b, f(11), c, g(01), d, h(00)



(a)



(b)

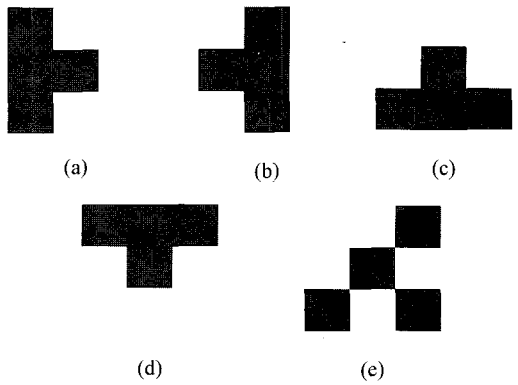
그림 4. (a) 끝점의 8방향 (b) 4개의 획성분
Fig. 4. (a) 8 directions of the End-point (b) 4 strokes

자음의 'ㅁ'이나 'ㅇ'과 같이 끝점이 존재하지 않는 경우에는 끝점이 할당되지 않기 때문에, 끝점 존재의 유, 무 판별 코드를 1 비트 추가해 준다. 따라서 방향코드 표현 2 비트와 유, 무 판별 코드 1 비트로 각 끝점마다 3 비트를 할당한다. 자음의 경우에 끝점의 수가 최고 5개까지 존재하며 이것을 표현하기 위해 각 끝점마다 3비트씩, 15비트의 끝점 입력 코드를 생성한다. 끝점의 15비트는 처음에 '100 100 100 100 100'로 초기화한다. 각 3비트의 첫번째 비트는 끝점이 존재하지 않는 경우 '1'이 되며, 나머지 두 비트는 모두 '0'이 된다. 끝점이 발견되면 3비트의 첫 비트를 '0'으로 하고 나머지 두개의 비트는 그 끝점에 알맞은 방향 코드를 할당한다. 예를 들어 자음 'ㄴ'의 경우 먼저 (a, e)방향의 끝점이 찾아진다. 이것의 코드는 '10'이며 첫비트를 '0'으로 만들어준다. 또 (c, g)방향의 끝점(방향코드 '01')이 발견되므로 'ㄴ'의 입력 15비트는 '010 001 100 100 100'가 된다. 모음의 경우 수직 모음은 끝점의 수가 최고 6개까지 존재하므로 Vv_BP 신경망은 18비트의 끝점 입력 코드가 필요하다. 또한 수평 모음은 4개까지 존재하므로 Vh_BP 신경망은 12개의 끝점 입력코드를 갖는다.

III-2. 분기점 특징과 특징 코드 할당

분기점의 조건은 다음과 같으며, 그림 5와 같이 5가지의 패턴이 존재한다.

p(x,y) = 1 and sum(x,y) = 2 (식 4)



(a) (b) (c)

(d) (e)

그림 5. 5가지 분기 패턴
Fig. 5. Five patterns of Branch-point

분기점의 경우 하나의 자소안에 동일한 분기 패턴이 2개가 동시에 존재할 수 있다. 각 패턴에 대해 3비트를 할당한다. 첫 비트(3비트 중의 최상 비트)는 모든 분기점 특징 비트를 '0'으로 하지 않기 위해 '1'의 값으로, 나머지 두 비트는 '0'으로 초기화한다. 위의 패턴 중에 어느 하나를 가지면 그것의 표현 3비트에서 2번째 비트를 '1'의 값으로 한다. 같은 분기 패턴이 또 하나가 존재한다면 그것의 마지막 비트를 '1'

로 해준다.

수직 모음의 경우는 'ㅏ' 분기와 'ㅓ' 분기만이 존재하므로 6개의 분기 특징 코드를 생성하도록 한다. 예를 들어 모음 'ㅐ'의 경우 분기점의 입력 6비트는 '110 110'이며 'ㅑ'의 경우는 '111 100'이 된다. 수평 모음의 경우는 'ㅗ' 분기와 'ㅜ' 분기만이 존재하므로 수직 모음과 마찬가지로 6개의 코드를 생성한다. 'ㅛ'의 경우 '111 100'의 특징 코드를 갖는다. 자음은 'ㅏ', 'ㅓ', 'ㅗ', 'ㅜ' 과 'ㅅ'분기를 갖기 때문에 15개의 분기 특징 코드 비트가 필요하다. 'ㅍ'의 경우는 그림의 (c)패턴이 2개 (d)패턴이 2개이므로 '100 100 111 111 100'이 된다.

III-3. 굴곡점 특징과 특징 코드 할당

굴곡점이란 꺾이는 점을 말하며 자음에만 존재하고 모음에는 존재하지 않는다. 그러므로 자음을 학습시키고 인식하는 신경망에서는 굴곡점 표현 비트가 할당되며, 수직 모음이나 수평 모음 인식을 위한 다른 2개의 신경망에서는 굴곡 특성 입력 코드를 생성할 필요가 없다. 그러므로 각각의 인식을 위한 신경망을 따로 설계하는 것이 학습에 있어서 시스템의 에러와 소요 시간을 줄일 수 있게 되며 인식률 또한 높일 수 있다. 굴곡점의 조건은 다음과 같고 패턴은 그림 6에서처럼 4가지이다.

p(x,y) = 1 and sum(x,y) = 1 and 방향코드의 굴곡 (식 5)

그림 6. 4가지의 굴곡 패턴

Fig. 6. Four patterns of Bent-point

굴곡점은 모든 자음에서 단 한번만 나타난다. 그러므로 위의 4개의 패턴에 대해 각각 2비트씩을 할당하여 굴곡점의 표현 비트는 8비트이다. 이것은 '10 10 10 10'로 초기화하며 굴곡점이 발견되면 두번째 비트를 '1'로 바꿔 준다. 자음 'ㅂ'의 경우 그림의 (c)패턴 굴곡점과 (b)패턴 굴곡점이 존재하므로 입력은 '10 11 11 10'이 된다.

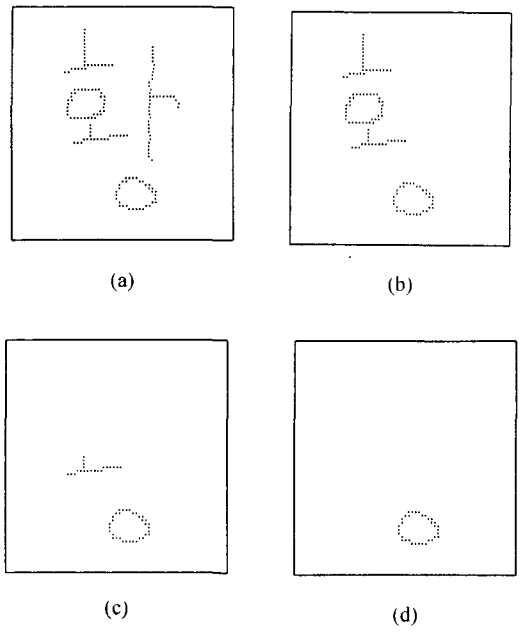
IV. 신경망을 이용한 인식

필기자에 의해 쓰여진 필기체 한글을 정규화 과정을 통해 64×64 크기의 입력으로 만들어 준다. 본 논문에서는 다른 3명에 의해 쓰여진 260개의 필기체 한글에 대해 인식률을 조사하였다. 특징을 추출하여 이진 코드로 변환하여 이를 신경망으로 학습한다. 학습 방법에는 고정 코드 방법, 학습 코드 방법 I, 학습 코드 방법 II를 사용하여 학습시키고 그 결과를 비교하였다. 또한 그 중에서 인식 결과가 좋은 방법을 선택하여 학습 횟수에 대한 인식률을 조사하였다.

IV-1. 자소 추출 결과

II-2장에서 설명했던 방법에 따라 6형식으로 분류하고 5×5 윈도우 방법을 이용하여 자소를 분리한다. 하나의 자소를 분리함과 동시에 특징들을 찾으며, 분리된 하나의 자소는 제거된다. 그림 7은 입력 글자 "황"에 대해 자소 분리 결과를 보인다. 필기체 한글 260자에 대한 자소 분리율은 다음 표 1과 같다. 자소 분리 결과를 보면 1, 2형식의 분리율은 97% 이상의 높은 분리율을 보이며, 5형식의 경우도 비교적 높은 분리율을 나타내고 있다. 이것은 수직 모음의 추출이 좋은 결과를 나타내기 때문이다. 필기체 한글 260자에 대하여 6가지 형식으로 분류된 패턴 245자에 있어서 각각 분리되어진 자소 개수는 다음과 같다.

수직 모음 : 158 개
수평 모음 : 153 개
자 음 : 368 개



(a) (b) (c) (d) (e)

그림 7. 입력 글자 "황"의 자소 분리 (a) 세선화 (b) 수직 모음 인식 후 제거 (c) 초성 자음 인식 후 제거 (d) 수평 모음 인식후 제거 (e) 종성 자음 인식후 제거

Fig. 7. "Jaso" separation of the input character "황" (a) thinning (b) The vertical vowel recognition and elimination (c) The first consonant recognition and elimination (d) The horizontal vowel recognition and elimination (e) The last consonant recognition and elimination

표 1. 자소 분리율

Table 1. The separation rate of "Jaso"

필기체 한글	1 형식	2 형식	3 형식	4 형식	5 형식	6 형식	전체
입력 문자 갯수	44	43	40	53	48	32	260
분리 문자 갯수	43	42	38	49	46	28	245
분리율(%)	97.73	97.67	95	92.45	95.83	87.5	94.23

IV-2. 신경망을 통한 분리된 자소 인식

한글 인식은 586 PC에서 Borland C 언어를 사용하여 3개의 신경망 알고리즘을 구현하였다. 위에서 6형식 분류를 통해 분리된 자소가 인식의 대상이며, 분리된 자음과 수직 모음, 수평 모음에 대해 각각의 인식 결과를 알아본다. 신경망을 이용한 인식은 다음과 같이 3가지 방법을 사용하였다. 첫번째 방법은 각신경망의 입력으로 사용되는 특징 코드 패턴을 각 자소가 표현되는 기본적인 특징 코드만 사용하여 학습시키고, 입력되는 필기체 한글의 인식률을 조사한 경우로 이를 고정 코드 방법(fixed-code method)이라 한다. 두번째 방법인 학습 코드 방법 I (learned-code I method)은 입력 한글에 대하여 전처리 과정을 거쳐 생성된 특징 코드에서 기본적인 코드와 조금씩 다른 모든 특징 코드만 입력 패턴으로 사용하여 이를 학습시킨 후, 학습에 사용된 입력 한글에 대하여 인식 결과를 알아보고 학습에 참여하지 않은 새로운 입력 한글에 대해 인식 결과를 알아보았다. 세번째 방법인 학습 코드 방법 II (learned-code II method)는 위의 학습 코드 방법 I에서처럼 전처리 과정을 거쳐 생성된 특징 코드를 입력 패턴으로 사용하는 경우이나, 학습 코드 방법 I의 경우에는 각 자소의 표현 코드 중에서 서로 다른 패턴만을 학습한 것에 반해 학습 코드 방법 II는 생성되는 특징 코드 패턴 모두를 입력으로 한다. 즉, 같은 코드 패턴이라도 계속해서 입력으로 학습하는 방법으로 학습의 질을 높인 경우이다. 또한 이 방법에서는 자음에 대한 인식만을 조사하였다. 즉 수직, 수평 모음 인식의 경우는 학습 코드 방법 I의 인식률이 100%이기 때문에 모든 특징 코드 패턴을 학습시킬 필요가 없다.

본 연구의 3가지 방법 모두 은닉층 유니트의 갯수는 입력의 2배로 하였고, 학습 횟수는 모두 5000번으로 제한하였다. 3개의 신경망의 전체적인 시스템 구성은 표 2와 같다. 고정 코드 방법을 이용한 인식의 결과는 표 3과 같다. 수평 모음의 인식은 100%이며, 수직 모음의 인식도 99.37%로 상당히 좋은 결과를 보인다. 이는 패턴의 수가 비교적 적고 굴곡점의 코드를 가지지 않아서 에러가 발생할 확률이 적어졌기 때문이다. 이에 비해 자음의 인식률은 86.14%로 만족스러운 결과는 아니다. 자음은 다양한 특징 코드를 갖고 있으며 모음에 비해 많은 패턴을 가지고 있기 때문에 인식률이 떨어지게 된다.

표 2. 각 신경망의 구조

Table 2. The structure of neural networks

		Vv_BP	Vh_BP	C_BP
입력 유니트 갯수		24	18	38
은닉 유니트 갯수		48	36	72
출력 유니트 갯수		9	5	14
학습 패턴 수	고정 코드 방법	9	5	20
	학습 코드 방법 I	11	6	79
	학습 코드 방법 II			240
학습 소요 시간	고정 코드 방법	3분 24초	57초	17분 30초
	학습 코드 방법 I	4분 32초	1분 14초	1시간 20분 6초
	학습 코드 방법 II			3시간 36분 23초
시스템 에러	고정 코드 방법	0.0297	0.0228	0.0307
	학습 코드 방법 I	0.0246	0.0207	0.0756
	학습 코드 방법 II			0.0696

표 3. 고정 코드 방법을 이용한 자소 인식률

Table 3. The recognition rate of the "Jaso" by using the fixed-code method

필기체 한글	Vv_BP	Vh_BP	C_BP
분리 자소 갯수	158	153	368
인식 자소 갯수	157	153	317
인식률(%)	99.37	100	86.14

학습에 사용되었던 필기체 한글의 자소에 대하여 학습 코드 방법 I을 이용한 인식의 결과는 표 4(a)이고 학습 코드 방법 II를 이용한 결과는 표 4(b)이다. 두 방법 모두 고정 코드 방법의 결과보다 좋은 결과를 보인다. 그러나 이 경우는 학습 패턴의 수가 많아서 학습을 위한 시간이 고정 코드 방법에 비해 현저히 많이 소요되는 단점이 있다. 수직 모음과 수평 모음은 100%의 인식 결과를 보이는 것에 비해 자음 인식률이 다소 떨어진다. 따라서 학습 코드 방법 II를 이용하여 자음의 경우만 학습시켜 보았다. 결과는 표 4(b)와 같으며 향상된 인식 결과를 알 수 있다. 학습에 사용되지 않았던 한글 자소에 대한 인식 결과는 표 5에서 보였다. 수직, 수평 모음은 100%의 인식률을 보이며 자음의 경우는 위와 마찬가지로 학습 코드 방법 II가 더 좋은 인식률을 보인다.

3가지 학습방법 중 가장 결과가 좋은 학습 코드 방법 II를 사용하여, 자음 인식을 위한 신경망의 학습 횟수를 2배로 하여 10,000번 학습시키고 이것의 인식 결과를 알아보았다. 학습 횟수를 5,000번에서 10,000으로 2배 늘려 학습시킴으로써 2배의 학습 시간이 소요되었으나, 시스템 에러는 0.0696에서 0.0667로 거의 줄어들지 않았다. 이는 시스템이 수렴해가고 있음을 나타내며, 더 학습시킨다 하더라도 시스템 에러가 줄어들지 않을 것으로 보인다. 또한 자음 인식률이 5,000번 학습한 경우와 동일하다. 이것은 학습에 의해 연결강도의 조정이 더 이상 크게 변화하지 않음을 의미한다.

표 4. 학습에 사용된 자소에 대한 인식 결과

Table 4. The recognition rate of "Jaso", which is used for the learning patterns

(a) 학습 코드 방법 I

(a) The learned-code I method

필기체 한글	Vv_BP	Vh_BP	C_BP
분리 자소 갯수	103	102	240
인식 자소 갯수	103	102	234
인식률(%)	100	100	97.50

(b) 학습 코드 방법 II

(b) The learned-code II method

필기체 한글	C_BP
분리 자소 갯수	240
인식 자소 갯수	236
인식률(%)	98.33

표 5. 학습에 사용되지 않은 한글 자소에 대한 인식 결과

Table 5. The recognition rate of "Jaso", which is not used for the learning patterns

(a) 학습 코드 방법 I

(a) The learned-code I method

필기체 한글	Vv_BP	Vh_BP	C_BP
분리 자소 갯수	55	51	128
인식 자소 갯수	55	51	118
인식률(%)	100	100	92.19

(b) 학습 코드 방법 II

(b) The learned-code II method

필기체 한글	C_BP
분리 자소 갯수	128
인식 자소 갯수	120
인식률(%)	93.75

결과들의 비교에서 수평 모음과 수직 모음의 경우 상당히 좋은 인식률을 보였다. 이는 학습 패턴이 적은 편이고 특징 코드도 자음 인식 신경망에 비해 적기 때문이며, 또 다른 이유는 둘 다 굴곡 특징을 코딩하지 않는다는 것이다. 5×5 윈도우를 이용하여 획을 따라가다 보면 모음의 경우도 굴곡으로 판단할 수 있는 경우가 발생하는데, 만약 굴곡으로 생각되어도 이것을 모음 인식 신경망의 입력으로는 사용하지 않으므로 영향을 받지 않는다. 따라서 모음 인식을 위한 두 개의 신경망에 굴곡이 코딩되지 않기 때문에 자, 모음 모두 28개의 자소를 함께 인식하는 단일 신경망보다 3개의 신경망을 각각 구성하는 경우가 좋은 인식 결과를 보임을 알 수 있다. 자음의 경우는 입력 패턴의 수가 수직 모음과 수평 모음의 경우보다 많고 신경망의 입력으로 쓰인 특징 코드의 수도 많아서 학습에 있어서 많은 시간이 요구되며, 인식률이 모음의 경우보다 떨어진다.

입력된 글자는 기본 자소들의 조합이므로 만약 하나의 자소가 오인식되면 결국 이러한 자소들로 이루어진 입력 글자가 오인식되는 결과를 낳는다. 위의 3가지 방법 각각에서 자소 인식에 의해 인식된 경우를 자소 단위가 아닌 글자 단위로 인식률을 알아보면 표 6과 같다. 이는 표 1에서 6가지 형식으로 분류된 245자에 대한 인식 결과이다. 글자를 이루는 자소중의 어느 하나가 오인식되는 경우는 인간이 글을 읽는 방법처럼 단어나 문맥을 이용하여 보완해야 한다.

표 6. 글자 단위 인식률

Table 6. The recognition rate per "word"

필기체 한글	고정 코드 방법	학습 코드 방법 I		학습 코드 방법 II	
		학습 참여 문자	학습 비참여 문자	학습 참여 문자	학습 비참여 문자
분리 문자 갯수	245	159	86	159	86
인식 문자 갯수	194	153	76	155	78
인식률(%)	79.18	96.23	88.37	97.50	90.70

V. 결 론

본 연구에서는 필기체 한글의 특징을 추출한 후 이를 신경망을 이용하여 인식하였다. 특징을 추출하여

이를 신경망의 입력으로 사용하게 되면, 적은 양의 입력 패턴으로 학습할 수 있는 장점을 가질 뿐만 아니라 특징을 추출하기 때문에 입력 패턴의 크기에 영향을 받지 않게 된다. 따라서 한글의 특징을 추출하고 이를 이진 코드로 변환하여 신경망의 입력으로 사용하는 방법이 오프라인 필기체 한글의 인식에 있어서 좋은 결과를 보임을 알 수 있다. 또한 한글의 6형식 분류에 있어서 새로운 방법을 제시하였고 이의 타당성을 실험을 통해 보였다. 또한 자음과 모음의 28자를 입, 출력으로 하는 단일 신경망을 이용하는 것보다 자소의 위치를 이용하여 3개의 신경망을 구성하는 것이 보다 적절한 방법임을 알 수 있었다. 본 논문에서 3가지 학습 방법을 비교하여 조사하였으며, 학습 코드 방법 II는 학습 코드 방법 I보다 3배나 많은 패턴을 학습시킴에도 불구하고 더 낮은 시스템 에러를 보였으며, 그 인식률 또한 향상되었음을 알 수 있었다.

6형식 분류와 특징 추출에 필요한 5×5 윈도우 방법은 픽셀을 이용하여 각 획을 따라가는 방법이기 때문에 전처리 과정의 결과에 민감해지는 단점을 가진다. 그러므로 좀 더 나은 전처리 과정을 사용한다면 자소 분리율 및 특징 코드 생성율을 향상시킬 수 있을 것이다.

참 고 문 헌

[1] C. H. Kim, S. Nishihara and T. K. Kim, "Adaptive recognition of on-line handwriting Hangul characters by using nueral networks," The Transactions of the Institute of Electronics and Communication Engineers of Japan, Vol. J-74D-II, NO. 10, pp. 1390 - 1397, Oct. 1991(In Japanese).

[2] 김 길중, 최 석, 잠 기곤, 윤 태훈, 김 재창, 박 의열, 이 양성, "순서 정보에 의한 한글 자획 온라인 인식을 위한 신경회로망에 관한 연구," 한국통신학회 논문지, 제 17권 제 12호, pp. 1380 - 1390, 1992년 12월.

[3] 성 태진, 박 승양, "문자조합 규칙 학습에 의한 온라인 한글 인식," 한국정보과학회 논문지, 제 20권 제 3호, pp. 304 - 316, 1993년 3월.

[4] 최 정훈, 권 희용, 김 춘석, 황 희용, "신경망 모델을 이용한 한글 필기체 온라인 인식," 한국정보과학회 논문지, 제 17권 제5호, pp. 540 - 549, 1990년 9월.

[5] 권 재욱, 조 성배, 김 진형, "계층적 신경망을 이용한 다중 크기의 다중 활자체 한글 문서 인식," 한국정보과학회 논문지, 제 19권 제 1호, pp. 69 - 79, 1992년 1월.

[6] 장 석잔, 강 선미, 김 혁구, 노 우식, 김 덕진, "자소 인식 신경망을 이용한 한글 문자 인식에 관한 연구," 대한전자공학회 논문지, 제 31권 B편 제 1호, pp. 81 - 87, 1994년 1월.

[7] 최 필웅, 이 기영, 구 하성, 고 형하, "접촉점에서의 국소 그래프 패턴에 의한 필기체 한글의 자소 분리에 관한 연구", 대한전자공학회지, 제30권 제 4 호, pp. 1 - 9, 1993년 4월.

[8] Rafael G. Gonzalez and Richard E. Woods, "Digital Image Processing", Addison Wesley.

[9] 김 형래, 박 인갑, 서 동필, 김 에녹, "CFG 방법을 이용한 필기체 한글에서의 자소 추출과 인식에 관한 연구", 대한전자공학회지, 제 29권 제9호, pp. 8 - 16, 1992년 9월.

저 자 소 개

金 美 英 (會員申請中)
1970년 11월 15일생. 1993년 건국대학교 전자공학과 졸업. 1995년 건국대학교 전자공학과 공학석사. 1995년 2월～현재 현대전자산업주식회사 메모리 개발연구소 주임연구원, 주 관심분야는 신경망, 메모리 칩설계 등임.

趙 鏞 範 (正會員)
1959년 5월 24일생. 1981년 경북대학교 전자공학과 졸업. 1988년 Univ. of South Carolina 전자 및 컴퓨터공학과 (석사). 1992년 Case Western Reserve Univ. 전자공학과 (공학박사). 1992년 5월～현재 건국대학교 전자공학과 부교수. 주 관심분야는 신경망 응용, VLSI 설계, 반도체 시뮬레이션 등임.