https://doi.org/10.5392/JKCA.2018.18.07.495

GoogLenet 기반의 딥 러닝을 이용한 향상된 한글 필기체 인식

Improved Handwritten Hangeul Recognition using Deep Learning based on GoogLenet

김현우, 정유진
한국외국어대학교 컴퓨터, 전자시스템공학부

Hyunwoo Kim(noma000@hufs.ac.kr), Yoojin Chung(chungyj@hufs.ac.kr)

요약

딥 러닝 기술의 등장으로 여러 나라의 필기체 인식은 높은 정확도 (중국어 필기체 인식은 97.2%, 일본어 필기체 인식은 99.53%)를 보인다. 하지만 한글 필기체는 한글의 특성으로 유사글자가 많은데 비해 문자의 데이터 수는 적어 글자 인식에 어려움이 있다. 하이브리드 러닝을 통한 한글 필기체 인식에서는 lenet을 기반으로 하여 낮은 레이어를 가진 모델을 사용하여 한글 필기체 데이터베이스 PE92에서 96.34%의 정확도를 보여주었다. 본 논문에서는 하이브리드 러닝에서 사용하였던 데이터 확장 기법(data augmentation)이나 multitasking을 사용하지 않고도 GoogLenet 네트워크를 기본으로 한글 필기체 데이터에 적합한 더 깊고 더 넓은 CNN(Convolution Neural Network) 네트워크를 도입하여 PE92 데이터베이스에서 98.64%의 정확도를 얻었다.

■ 중심어 : | 한글 필기체 인식 | CNN | GoogLenet | PE92 데이터베이스 |

Abstract

The advent of deep learning technology has made rapid progress in handwritten letter recognition in many languages. Handwritten Chinese recognition has improved to 97.2% accuracy while handwritten Japanese recognition approached 99.53% percent accuracy. Hanguel handwritten letters have many similar characters due to the characteristics of Hangeul, so it was difficult to recognize the letters because the number of data was small. In the handwritten Hanguel recognition using Hybrid Learning, it used a low layer model based on lenet and showed 96.34% accuracy in handwritten Hanguel database PE92. In this paper, 98.64% accuracy was obtained by organizing deep CNN (Convolution Neural Network) in handwritten Hangeul recognition. We designed a new network for handwritten Hangeul data based on GoogLenet without using the data augmentation or the multitasking techniques used in Hybrid learning.

■ keyword : | Handwritten Hangeul Recognition | CNN | GoogLenet | PE92 Database |

I. 서 론

최근에는 인공신경망의 발전으로 인해 이미지 인식에 있어 많은 발전이 있었고 이를 한글 필기체에 적용시킨 연구[1][2]들이 있었다. 이러한 연구의 결과로 한글 필기체 인식의 수준은 상당히 올라갔지만 아직 다른

* 이 연구는 한국외국어대학교 교내학술연구비와 한국연구재단의 이공분야 기초연구사업(NRF-2016R1A2B4015141)의 지원에 의하여 이루어진 것임.
접수일자 : 2018년 05월 28일
수정일자 : 2018년 06월 25일
심사완료일 : 2018년 07월 02일
교신저자 : 정유진, e-mail : chungyj@hufs.ac.kr

언어들(중국어[3], 일본어[4])의 정확도에 비해서는 높지 않다. 중국어 필기체 연구의 경우에는 상당히 큰 데이터베이스를 사용한다. 사용된 데이터베이스는 CASIA-OL HWDB1.1이며 클래스는 약 3,800개로 데이터는 110만개를 보유하고 있다. 중국어 데이터베이스의 클래스는 3800개로 매우 많지만 클래스별 데이터도 충분히 많아서 간단한 네트워크를 도입하였음에도 97.20%의 정확도를 얻었다. 일본 필기체의 데이터 셋은 ETL-1,8 (Electrotechnical Laboratory Character Database)를 사용하는데 이 데이터 셋은 크게 3가지로 나뉜다. 첫 번째는 ETL-1 Katakana 데이터 셋이고 클래스는 51개, 데이터의 수는 71,961개이다. 두 번째는 ETL-8 Hiragana로 클래스 75개, 데이터의 수는 12,000개 이다. 마지막으로 한자를 일본어에 맞도록 변형은 Kanji의 경우 클래스는 878개, 데이터 수는 14만개 이다. 일본어 데이터베이스는 클래스 개수가 적으며 클래스간의 유사성이 매우 적어서 99.53%의 정확도를 보여주었다.

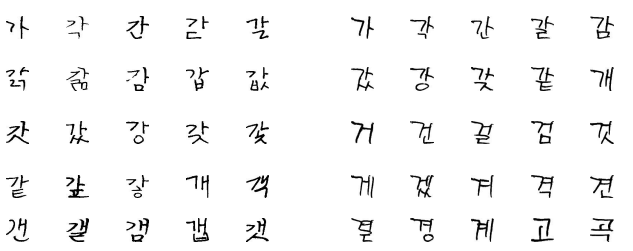
이전 한글 필기체 인식 관련 논문[1][2]에서는 한글 필기체 인식을 위해 LeCun Yann의 CNN(Convolutional Neural Network) 모델인 lenet[5]을 데이터에 맞게 변형된 모델을 사용 하였다. [1]에서는 이미지의 개수를 증가시키기 위한 Elastic distortion[1]을 사용한다. 이는 이미지의 각 픽셀 값을 +1 또는 -1 랜덤으로 이동시켜 이미지 량을 늘리는 방법이다. 이러한 데이터 늘리기 작업을 통해 PE92 데이터 셋에서 92.92%의 정확도를 얻었으며, [2]의 논문에서는 유사 클래스간의 구별을 하는 Multitasking 방법인 Hybrid learning[2] 을 적용하였다. Hybrid learning은 필기체 데이터를 분류하는 테스크와 10개의 유사클래스 간의 차이를 계산하는 테스크를 더한 값을 오차 함수로 사용한다. 이 방법을 통해 [2]논문에서는 PE92 데이터베이스에서 96.34% 정확도를 얻었다. [6]에서는 inception 모듈[7][8]을 사용하였으나 전체적 구조가 충분히 깊지 않았고 inception 모듈을 네트워크 전반에 위치시켰으나 본 논문에서는 convolution과 max_pool을 전반에 위치시켜 1차 특징을 뽑아낸 뒤 inception 모듈을 적용하였다. 그리고 [6]은 마지막에 2계층의 fully-connected 레이어를 두었으나 이는 너무 많은 파라미터를 사용해야 한다. 따라서 본 논문에서는 마지막 분류를 위한 하나의 fully-connected 레이어만 사용하며 대신 inception 레이어의 개수를 늘렸다. [6]에서는 낮은 레이어와 이미지에 바로 inception 모듈을 적용시켰고 SERI95a 데이터베이스에서 [1][2]보다 낮은 94.3%정확도를 보여주었다.

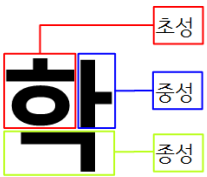
최근 이미지 인식 분야에서 깊은 CNN의 사용으로 좋은 결과들이 나오고 있는데 본 논문에서는 한글 필기체 인식을 위해 inception 모듈[7][8]을 사용한 좀 더 깊은 CNN을 구성하여 98.64%의 정확도를 얻었다. 2장에서는 한글의 구조, 3장에서는 본 연구의 한글 필기체 인식 모델의 구성과 사용된 기술들, 4장에서는 학습률의 변화 방식에 따른 정확도, 5장에서는 실험결과, 6장에서는 결론을 기술한다.

Ⅱ. 한글구조와 한글 필기체 데이터

한글은 자음과 모음을 결합시켜 하나의 완성형 글자를 만들고 글자에 코드를 부여하는 완성형 인코딩을 사용한다. 한 글자에는 크게 3가지 음절로 나뉘며 각각 초

표 1. PE92(left)와 SERI95a(right) 데이터베이스의 샘플





o 초성 : ㄱ ㄲ ㄴ ㄷ ㄸ ㄹ ㅁ ㅂ ㅃ ㅅ ㅆ ㅇ ㅈ ㅉ ㅊ ㅋ ㅌ ㅍ ㅎ(19개)

o 중성 : ㅏ ㅐ ㅑ ㅒ ㅓ ㅔ ㅕ ㅖ ㅗ ㅘ ㅙ ㅚ ㅛ ㅜ ㅝ ㅞ ㅟ ㅠ ㅡ ㅢ ㅣ (21개)

o 종성 : ㄱ ㄲ ㄳ ㄴ ㄵ ㄶ ㄷ ㄹ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅁ ㅂ ㅄ ㅅ ㅆ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ (27개)

그림 1. 한글의 구조

어가며 중성에는 모음이 들어가게 된다. 각 자리에 들어 갈수 있는 글자는 [그림 1]과 같으며 종성은 없을 수도 있으므로 초성(19) × 중성(21) × 종성(27+1)을 계산하면 총 11,172개의 글자를 표현 가능하다.

영어의 경우 알파벳을 기반으로 하여 256개의 코드로 표현이 가능하지만 한국어의 경우에는 그 수가 많아 ASCII 코드로는 처리가 불가능하다. 그래서 새로운 인코딩 방식이 필요하였는데 이전에는 많이 사용되는 2350개의 글자만을 모아 EUC-KR 인코딩 방식을 사용하였다. 그러나 최근에는 CP949나 유니코드인 UTF-8 인코딩 방식을 사용하며 이는 11,172개의 글자를 모두 지원한다.

한글 필기체 데이터베이스는 PE92와 SERI95a 그리고 전자정보연구센터에서 보유한 213만개의 필기체 데이터가 있다. PE92는 자주 사용되는 2350개의 클래스와 각 클래스 당 100개의 데이터를 가지고 있다. SERI95a는 PE92보다 적은 520개의 클래스와 클래스 당 1000개의 데이터를 가지고 있다. 전자정보연구센터에서 보유한 필기체 데이터는 9개의 세트가 존재하며 각 세트마다 서로 다른 클래스와 데이터 수를 보유하고 있다. [표 1]은 두 데이터베이스에서 첫 번째 이미지부터 순서대로 뽑아낸 샘플이며, PE92의 클래스의 수가 SERI95a보다 많아 더 다양한 문자들이 포함되어 있다. 본 논문에서는 좀 더 다양한 문자를 인식할 수 있는 PE92 데이터베이스를 사용한다.

Ⅲ. 한글 필기체 인식에 사용된 모델

딥 러닝 기법의 하나인 CNN(Convolutional Neural Network)을 사용할 때, 데이터가 적고 클래스가 많을수록 깊이가 낮은 네트워크에서는 학습이 잘 되지 않아 낮은 정확도를 보인다. 이에, GoogLenet[7][8], Resnet[9] 등의 깊은 네트워크가 개발되어 이미지 인식에 우수한 성능을 보였다. 본 논문에서는 GoogLenet을 한글 필기체 인식에 맞도록 개선하였다.

GoogLenet은 주로 사람이나 자동차와 같이 큰 이미지를 분류하는 네트워크로 299×299의 그리드와 색을 표현하기 위한 3(RGB)개의 채널을 사용한다. 그러나 한글 필기체 데이터베이스는 평균 그리드가 60×60이고, 흑백이미지인 1채널 데이터이므로 기존 네트워크를 그대로 사용하면 한글 필기체 이미지의 그리드나 채널에 적합하지 않으며 이 차이로 인해 네트워크에 무의미한 차원이 추가 되고 이는 네트워크를 무겁게 하여 자원의 낭비를 발생시켰다. 이에 다음과 같이 GoogLenet을 개선하였다.

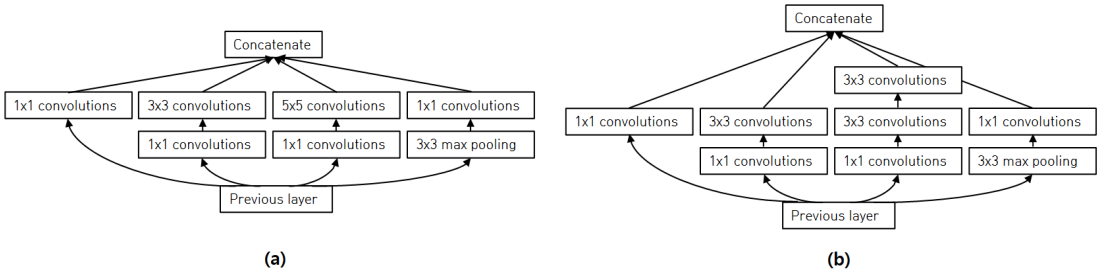


그림 2. (a) 기본 inception 모듈과 (b) Factorization 하여 연산량을 줄인 모델

GoogLenet은 크게 3가지 부분으로 이루어져 있다. 첫 번째 부분은 보조 분류기로 중간 단에 배치되며 이미지 분류를 돕는 역할을 하지만 추가로 필요한 레이어에 비해 높은 성능은 보여주지 않아 본 논문에서는 제거하였다.

표 2. GoogLenet stem 영역 수정표

GoogLenet의 stem 영역	수정된 모델
convolution(7×7/2) max_pool(3×3/2) convolution(3×3/1) max_pool(3×3/2)	convolution(5×5/1) max_pool(3×3/2)

GoogLenet의 두 번째 부분은 stem영역이라 하는데, 이는 [표 2]의 왼쪽 열과 같다. GoogLenet에서 stem 영역을 둔 이유는, 이미지 인식을 할 때 세 번째 부분인 inception 모듈을 바로 적용하면 클래스 간 분류효과가 떨어지기 때문에 stem 영역에서 convolution과 max_pool를 두어 특징 맵(feature map) 을 생성한다. 이때 7×7 convolution의 큰 필터를 사용하여 그리드를 줄이는데, 60×60 그리드를 가진 한글 필기체 데이터베이스의 경우에는 7×7 convolution의 큰 필터를 사용할 필요가 없어서 [표 2]의 오른쪽 열과 같이 개선하였다.

GoogLenet의 세 번째 부분은 inception 모듈로 [그림 2(a)]와 같다. 이 모듈은 1×1, 3×3, 5×5 3개의 필터를 사용하여 데이터의 인접 정보들을 모은다. 이렇게 얻은 특징 맵(feature map)들을 합친 뒤 다음 레이어의 입력으로 전달한다. 이러한 작업은 inception모듈에서 동일하게 수행하고 이를 통해 수많은 조합의 특징들을 얻는 것이 가능해 진다. 이러한 다양한 특징들의 조합을 통해 [가:갸][뭣:뭿]와 같은 유사 클래스 사이의 분류 성능이 하나의 3×3 필터를 사용해 네트워크를 쌓는 것 보다 더 좋은 성능을 발휘한다. max_pool 또한 병렬적으로 inception 모듈에 들어가게 되는데 이는 기존 convolution, max_pool 구조의 효과가 좋았기 때문에 추가되었다. 하지만 이처럼 convolution 레이어를 병렬로만 구성하여 합치게 되면 각 레이어의 부피가 늘어나기 때문에 3×3과 5×5필터 앞에 1×1 필터를 붙임으로써 채널의 크기를 조절한다. max_pool의 경우에는 채널을 컨트롤 하는 특성을 가지지 않았기 때문에 max_pool을 한 뒤 1×1 필터를 붙여 채널 사이즈를 조절한다. 본 연구에서는 파라미터의 수를 줄이기 위하여 네트워크 5×5필터를 두 개의 3×3 필터로 변형한 factorization inception[8] 모듈을 사용하여 파라미터를 30%를 줄였다. 이 모듈에서 입력되는 채널의 개수와 출력되는 채널의 개수를 바꾸어 가며 채널의 크기를 변형하고 pooling을 통하여 이미지의 그리드 사이즈를 바꾸어 주며 최적의 값을 찾았다. 본 논문에서 개발한 한글 필기

표 3. 한글 필기체 인식을 위해 사용된 깊은 신경망 모델

type	filter size/stride	input size	output size	branch 1×1/filter	branch 3×3/filter	branch 5×5/filter	avg_pool 3×3/filter
convolution	5×5/1	60×60×1	56×56×64				
max_pool	3×3/2	56×56×64	28×28×64				
inception_(a)		28×28×64	28×28×128	32	24/32	32/48/48	16
inception_(b)		28×28×128	28×28×256	64	48/64	64/96/96	32
max_pool	3×3/2	24×24×256	14×14×256				
inception_(b)		14×14×256	14×14×256	64	48/64	64/96/96	32
inception_(b)		14×14×256	14×14×256	64	48/64	64/96/96	32
inception_(b)		14×14×256	14×14×256	64	48/64	64/96/96	32
inception_(b)		14×14×256	14×14×256	64	48/64	64/96/96	32
inception_(c)		14×14×256	14×14×512	128	96/128	128/192/192	64
max_pool	3×3/2	14×14×512	7×7×512				
inception_(c)		7×7×512	7×7×512	128	96/128	128/192/192	64
inception_(c)		7×7×512	7×7×512	128	96/128	128/192/192	64
average_pool	7×7/1	7×7×512	1×1×512				
dropout(0.4)							
fully connected		512(flatten)	2350				

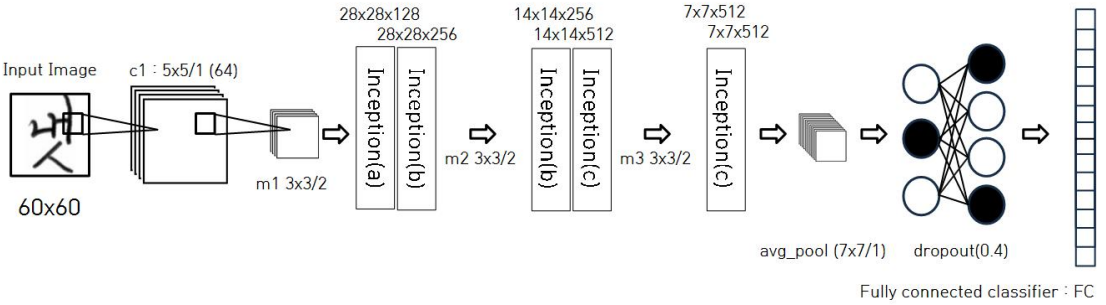


그림 3. 한글 필기체 인식을 위한 전체 네트워크 구성도

체 인식을 위한 깊은 신경망 모델의 자세한 내용은 [표 3]와 같다.

[그림 3]은 [표 3]을 그림으로 표현한 것으로 한글 필기체 네트워크의 전체 구성도이다. 한글 필기체 데이터는 평균적으로 60×60의 그리드 크기를 가지고 있지만 네트워크에 입력받기 위해서는 고정된 이미지 그리드가 필요하다. 따라서 모든 이미지 데이터를 평균 사이즈인 60×60으로 고정시키기 위하여 선형보간법을 통해 이미지 전처리를 한다. 그 다음 CNN의 가장 기본적인 형태인 convolution, max_pool을 사용하는데 이 사이에 학습률의 증가를 통한 빠른 학습과 이로 인해 생기는 미분 값 발산(exploding gradient problem)과 미분 값 소멸 문제(vanishing gradient problem)를 막기 위하여 batch_normalization을 넣고 비선형 이미지를 분류하기 위한 활성화 함수인 ReLU[10]를 넣는다. 즉 전체적 구조는 [convolution, batch_normalization, ReLU, max_pool]이다. 그 다음, 앞서 기술한 개선된 inception 모듈을 적용하는데 이 모듈에서는 기본적으로는 같은 채널을 유지한다. 그리고 max_pool 이전에 inception 모듈이 위치하게 되면 pooling 으로 인한 정보의 소실을 완화하기 위하여 pooling 레이어의 채널수를 2배로 늘려준다. inception 모듈의 연산을 통해 7×7 그리드의 512개의 채널이 만들어 지며 7×7의 데이터의 평균을 구한 뒤 fully connected를 통해 분류 할 수 있도록 1×1×512의 노드로 변형한다. 그리고 2350의 fully connected를 통해 분류를 하게 된다.

성능을 측정하기 위해 이전 연구[1]에서는 MSE (Mean Square Error)[11]를 사용하며 오차 함수는

$$E_{MSE} = \frac{1}{N}\sum_{i}(X_i - Y_i)^2 \quad (1)$$

와 같이 주어진다. 여기서 X는 모델을 통한 예측 값, Y는 실제 정답이다. MSE는 예측 값과 실제 값의 L2 distance를 오류 값으로 사용한다. 그러나 MSE 형태의 오차 함수는 일반적으로 미분 시 기울기 값이 0에 가까운 값이 나와 수렴이 느려지는 단점이 있다. 이에 본 연구에서는 CE(Cross Entropy)[12]를 사용하며

$$E_{CE} = -\sum_{i} Y_i \log(X_i + \epsilon) \quad (2)$$

와 같다. X는 모델을 통한 예측 값이고, Y는 실제 정답이며 log(X)가 무한대로 발산하는 것을 막기 위해 ϵ라는 아주 작은 값을 더해준다.

Ⅳ. 학습률의 변화 방식에 따른 정확도

본 논문에서 다양한 학습률을 사용해서 학습을 하였다. Adam optimizer[13]를 사용 시 일반적으로 학습률을 0.001로 잡고 학습을 하지만 학습시간이 길어지게 되면 어느 시점에서 학습을 하여도 시험 정확도가 제자리걸음을 하게 된다. 이와 같은 고정된 학습률은 세대가 길어질수록 고정되어 loss 그래프에 대해 최적의 위치를 찾기 어려워지기 때문인데 이를 해결하기 위해 너무 작은 학습률을 사용하게 되면 네트워크 오류 값이 지역 최소 오류 값(local optimization)에 수렴하여 모델의 전체 최소 오류 값(global optimization)을 찾을 수 없게 된다. 따라서 최적의 학습률을 찾기 위해 다양한 테스트를 해보았다. 테스트를 위해 학습률을 2개 고정

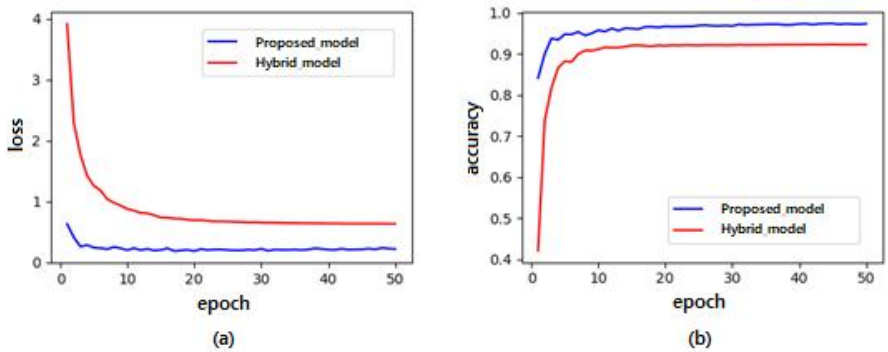


그림 4. 이전 모델과 제안된 모델의 (a) loss그래프와 (b) accuracy 그래프 비교

된 값과 3가지 천천히 감소하는 방법을 사용하였다. 학습률은 Adam optimizer에서 주로 사용되는 0.001과 0.0001두개의 고정 값에 대해 아래의 세 가지 방법으로 감소시켜 나가며 트레이닝을 한다.

첫 번째 방법은 학습률을 매 트레이닝 마다 지수 비율로 감소시키는 방법이다. 수식은

$$LR_{\exp} = \exp\left[\frac{(t-c)\log(lr_1)+(c-1)\log(lr_2)}{t-1}\right] \quad (3)$$

를 사용하고 t는 총 트레이닝 횟수, c는 현재 트레이닝 횟수, lr1은 초기 학습률, lr2는 마지막 학습률이다. 이러한 방식의 학습률 감소는 지수 형태를 가지며 초반에는 넓은 폭으로 학습률을 감소시키고 학습이 진행 될수록 그 폭이 작아진다. 두 번째 방법은 선형으로 학습률을 감소시킨다. 각 세대마다 초반 학습률과 마지막 학습률에 맞추어 일정하게 값을 줄인다. 수식은

$$LR_{linear} = \frac{lr_1(t-c)-lr_2 c}{t} \quad (4)$$

와 같다. 마지막 방법은 처음 30세대 동안은 0.001, 30~60세대 동안은 0.0005, 그리고 나머지 세대는 0.0001로 하는 방법(앞으로 이를 cliff drop 방법이라 부른다)을 사용하여 총 5가지 방법에 대하여 학습을 하였고 결과는 [표 4]와 같다.

표 4. 학습률 변화 방식에 따른 정확도

학습률 설정 방법	정확도
fixed_0.001	98.41%
fixed_0.0001	98.08%
hybrid model's equation	98.60%
linearly reduction equation	98.63%
cliff drop	98.64%

V. 실험 결과

본 논문에서는 이전 논문 [1][2]에서 사용된 PE92 한글 필기체 데이터베이스를 사용하여 실험을 진행하였다. 해당 데이터베이스는 2350개의 클래스로 이루어져 있으며 각 클래스 당 100개의 데이터를 가지며 총 235,000개의 데이터를 가지고 있다. 학습은 총 100 세대 동안 진행 되었으며 배치사이즈로는 128로 설정하여 약 183,500 번의 학습을 진행하였다. 학습 데이터와 시험 데이터의 비율은 이전 논문과의 비교를 위해 동일하게 9:1로 나누어 진행하였다. 네트워크가 무거워 짐에 따라 미분 값 소멸 문제와 학습 속도 저하의 문제가 있었고 이는 batch normalization[14]을 사용함으로 약간의 성능 향상이 있었다. 그러나 이전 모델이 세대 당 4분이 걸렸던 것에 비하면 약 4배정도인 15분이 소요되었다. 학습률은 4장에서와 같이 실험을 해본 결과 학습률을 일정 세대마다 급격하게 떨어뜨리는 방법이 가장 효과가 좋았으며 학습률을 고정 하는 방식에 비해 0.001보다 0.21% 향상되었고 0.0001보다는 0.56%향상된 결과를 보여주었다. 하드웨어는 그래픽 카드로 GTX 1080ti, CPU로는 intel i7-7700를 사용하였으며 소프트웨어로는 python3.5, Ubuntu 16.04.1 그리고 Tensorflow gpu(1.3.0)을 사용하여 연구를 진행하였다. 본 논문에서는 논문[1][2]의 elastic distortion, hybrid learning을 제외한 순수 네트워크만의 성능만을 비교하였으며 [1][2]에서 사용된 네트워크를 Hybrid_model이라 표기한다. 네트워크의 성능은 평균 오차률(Cross Entropy loss), 그리고 모델을 통해 나온 가정 값과 실

제 정답과의 정확도(accuracy)을 사용해 비교한다. [그림 4(a)]는 본 논문의 개발 된 네트워크와 Hybrid_model의 오차율(loss) 그리고 [그림 4(b)]는 정확도를 비교한 그래프이고 본 논문에서 개발된 네트워크를 사용 시 기존 논문[1][2]에서의 96.34%의 정확도보다 2.3% 높은 98.64%를 얻을 수 있었다. 이를 통해 깊은 네트워크의 사용함으로 한글 필기체 분류에 있어서 더 높은 정확도를 얻는 것이 가능함을 보여 주었다.

VI. 결론

우리는 기존 논문에서 사용되었던 데이터 확장 기법(data augmentation)이나 multitasking을 사용하지 않는 깊은 네트워크의 구축과 다양한 학습률을 변화시키며 실험하였고 인접 데이터들의 연관 정보를 모으는 개선된 inception 모듈을 사용하여 정확도를 향상시킨 한글 필기체 인식 네트워크를 개발하였다. 파라미터 수를 줄이기 위하여 기존 5×5 필터를 두 개의 3×3필터로 표현한 factorization을 사용하였다. 그리고 1×1 convolution을 각 convolution 앞에 위치시킴으로 학습의 안전성과 깊은 네트워크 구축이 가능하였다. 각 convolution 레이어 사이의 미분 값 소멸(vanishing gradient problem) 현상을 줄이고 빠른 학습을 위해 batch normalization 및 dropout[15]을 적용하여 모델 구성을 하였고 학습 시 기존 논문[1][2]에 사용된 모델과 비교를 위해 한글 필기체 PE92 데이터베이스를 사용 하였다. 새로운 모델의 도입을 통해 기존 논문[1][2]보다 높은 98.64%의 정확도를 얻을 수 있었다. 실생활에서도 좀 더 잘 동작하는 네트워크를 만들기 위해서는 먼저 더 많고 다양한 한글 데이터베이스가 구축되어야 할 것이고 그 글자들을 인식할 수 있도록 CNN에 대한 다양한 연구 또한 계속 되어야 할 것이다.

참 고 문 헌

[1] In-Jung Kim and Xiaohui Xie, "Handwritten Hangul recognition using deep convolutional neural networks," International Journal on Document Analysis and Recognition (IJDAR), Vol.18, No.1, pp.1-13, 2015.

[2] In-Jung Kim, Changbeom Choi, and Sang-Heon Lee, "Improving discrimination ability of convolutional neural networks by hybrid learning," International Journal on Document Analysis and Recognition (IJDAR), Vol.19, No.1, pp.1-9, 2016.

[3] Weixin Yang, Lianwen Jin, Zecheng Xie, and Ziyong Feng, "Improved deep convolutional neural network for online handwritten Chinese character recognition using domain-specific knowledge," Document Analysis and Recognition (ICDAR), pp.551-555, 2015.

[4] Charlie Tsai, *Recognizing Handwritten Japanese Characters Using Deep Convolutional Neural Networks*, Technical Report, Stanford University, pp.1-7, 2016.

[5] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner, “Gradient- based learning applied to document recognition,” Proceedings of the IEEE, Vol.86, No.11, pp.2278- 2324, 1998.

[6] 강우영, 김병희, 장병탁, "인셉션 모듈 기반의 보다 깊은 컨볼루션 신경망을 통한 한글 필기체 인식," 한국정보과학회 학술발표논문집, pp.883-885, 2016.

[7] Christian Szegedy, Wei Liu, Yangquing Jia, and Pierre Sermanet, Scott Reed, "Going deeper with convolutions," Proceedings of the IEEE conference on computer vision and pattern recognition, pp1-9, 2015(6).

[8] Christian Szegedy, Vincent Vanhoucke. Sergey Ioffe, and Jon Shlens, "Rethinking the inception architecture for computer vision," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.2818-2826, 2016.

[9] Kaiming He, Xiangyu, Zhang, Shaoquing Ren,

and Jian Sun, "Deep residual learning for image recognition," Proceedings of the IEEE conference on computer vision and pattern recognition, pp.770-778, 2016.

[10] Vinod Nair and Geoffrey E. Hinton, "Rectified linear units improve restricted boltzmann machines," Proceedings of the 27th international conference on machine learning (ICML-10), pp.807-814, 2010.

[11] Zhou Wang and Alan C. Bovik, "Mean squared error: Love it or leave it? A new look at signal fidelity measures," IEEE signal processing magazine, Vol.26, No.1, pp.98-117, 2009.

[12] Pieter-Tjerk De Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein, "A tutorial on the cross-entropy method," Annals of operations research, Vol.134, No.1, pp.19-67, 2005.

[13] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

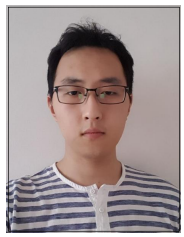
[14] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Rusian Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," Journal of machine learning research, Vol.15, No.1, pp.1929-1958, 2014.

[16] Ankit Sharma and Dipti R. Chaudhary, "Character recognition using neural network," International Journal of Engineering Trends and Technology (IJETT), Vol.4, pp.662-667, 2013.

저 자 소 개

김 현 우(Hyunwoo Kim) 준회원



- 2012년 3월 ~ 현재 : 한국외국어대학교 컴퓨터 공학과(학사)
- 2017년 6월 ~ 현재 : 한국외국어대학교 RTDCS 연구실(연구원)

<관심분야> : 머신러닝, 이미지 프로세싱

정 유 진(Yoojin Chung) 정회원

- 1989년 : 서울대학교 컴퓨터공학과 학사
- 1991년 : 서울대학교 컴퓨터공학과 석사
- 1997년 : 서울대학교 컴퓨터공학과 박사
- 1997년 ~ 1999년 : 서울대학교 컴퓨터신기술연구소 특별연구원
- 1999년 ~ 2001년 : 부산대학교 컴퓨터및정보통신연구소 기금조교수
- 2001년 ~ 현재 : 한국외국어대학교 컴퓨터·전자시스템공학부 교수

<관심분야> : 딥러닝, 알고리즘, 정보검색