ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

CAPTCHA에 사용되는 숫자데이터를 자동으로 판독하기 위한 Autoencoder 모델들의 특성 연구☆

A Study on the Characteristics of a series of Autoencoder for Recognizing Numbers used in CAPTCHA

전 재 승[1] 문 종 섭[1*]
Jae-seung Jeon Jong-sub Moon

요 약

오토인코더(Autoencoder)는 입력 계층과 출력 계층이 동일한 딥러닝의 일종으로 은닉 계층의 제약 조건을 이용하여 입력 벡터의 특징을 효과적으로 추출하고 복원한다. 본 논문에서는 CAPTCHA 이미지 중 하나의 숫자와 자연배경이 혼재된 영역을 대상으로 일련의 다양한 오토인코더 모델들을 적용하여 잡음인 자연배경을 제거하고 숫자 이미지만을 복원하는 방법들을 제시한다. 제시하는 복원 이미지의 적합성은 오토인코더의 출력을 입력으로 하는 소프트맥스 함수를 활성화 함수로 사용하여 검증하고, CAPTCHA 정보를 자동으로 획득하는 다른 방법들과 비교하여, 본 논문에서 제시하는 방법의 우수함을 검증하였다.

☞ 주제어 : CAPTCHA, 오토인코더, 적층 오토인코더, 잡음 제거, SOFTMAX, 딥러닝

ABSTRACT

Autoencoder is a type of deep learning method where input layer and output layer are the same, and effectively extracts and restores characteristics of input vector using constraints of hidden layer. In this paper, we propose methods of Autoencoders to remove a natural background image which is a noise to the CAPTCHA and recover only a numerical images by applying various autoencoder models to a region where one number of CAPTCHA images and a natural background are mixed. The suitability of the reconstructed image is verified by using the softmax function with the output of the autoencoder as an input. And also, we compared the proposed methods with the other method and showed that our methods are superior than others.

☞ keyword : CAPTCHA, Autoencoder, Stacked autoencoder, Denoising, SOFTMAX, Deep learning

1. 서 론

Completely Automated Public Turing test to tell Computers and Humans Apart(CAPTCHA, 캡챠)는 사용자가 사람인지 아닌지를 구분하기 위해 이용하는 기술로 인간은 쉽게 풀 수 있지만, 컴퓨터는 풀기 어려운 문제를 제시하여 자동 프로그램을 이용한 자동 계정 생성, 게시물 등록 등을 방지하는데 적용되고 있다[1].

캡챠의 양식은 텍스트 기반 캡챠, 오디오 기반 캡챠, 이미지 기반 캡챠 등 여러 가지가 존재하며, 최근에는 텍스트나 이미지 등을 이용하는 문제가 아니라 방문자의 상태에 따라 봇을 구분하는 Invisible reCAPTCHA가 발표되기도 하였다[2]. 하지만 아직까지 대부분의 웹사이트에서는 텍스트 기반 캡챠가 주로 사용되고 있다[3].

텍스트 기반 캡챠는 기계 학습 알고리즘을 적용한 문자 인식 기술의 발전에 따라 개별 문자의 자동화된 판독이 쉬워졌으며[4], 이에 따라 개별 문자의 인식을 어렵게 만들거나, 문자와 문자 또는 문자와 배경 간 분리를 어렵게 하는 방법으로 캡챠의 자동화된 판독을 방지하고 있다[5]. 이중 개별 문자의 인식을 어렵게 하기 위한 방법으로는 폰트의 종류와 크기 조정, 문자 기울이기, 문자 변형·왜곡하기, 문자 흐리게 하기 등의 방법들이 있다. 분리를 방지하는 기술로는 문자와 문자 간 간격을 제거하여 개별 문자로의 분리를 어렵게 하는 방법 이외에도 캡챠 이미지에 복잡한 배경과 문자를 함께 삽입하거나, 문

1 Graduate School of Information Security, Korea university, Seoul, 02481, Korea
* Corresponding author (jsmoon@korea.ac.kr)
[Received 1 September 2017, Reviewed 3 September 2017, Accepted 2 October 2017]
☆ 이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-00427, 딥러닝을 이용한 시스템 독립적 악성코드 및 취약점 탐지 기술개발)

자가 아닌 선들을 삽입하는 등의 방법들이 있다.

본 논문에서는 캡챠에 들어있는 자연배경을 제거하고 숫자 데이터를 자동으로 판독하기 위해 오토인코더(Autoencoder, AE) 모델들을 사용한다. 즉, 오토인코더 모델들의 입력 데이터에 대한 복원 기능을 이용하여 자연배경을 제거하는 방법을 제시한다. 오토인코더의 모델은 하나의 은닉 계층으로 구성된 전통적인 오토인코더와 여러 개의 은닉 계층으로 구성된 적층 오토인코더(Stacked autoencoder)를 사용하고 각 구조의 복원 결과를 비교 분석하여 가장 효과적인 알고리즘을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로서 캡챠 공격에 사용된 대표적인 방법들에 대해 설명하였으며, 3장에서는 본 논문에서 제안하는 시스템의 전체적인 구조와 사용되는 딥러닝 모델에 대하여 설명하였다. 즉, 하나의 숫자와 자연배경, 잡음으로 구성된 이미지에서 숫자를 제외한 자연배경과 잡음을 제거하는 시스템의 구조를 살펴보고, 이때 사용하는 오토인코더와 적층 오토인코더에 대해 알아본다. 또한 적층 오토인코더를 효과적으로 학습시키는 Greedy layer-wise unsupervised training 기법에 대하여 살펴보고, 오토인코더에 잡음을 주는 방법과 이의 특성을 살펴본다. 4장에서는 실험의 세부과정을 설명하고 실험결과를 분석한다. 마지막으로 5장에서 결론을 맺으며 논문을 마무리한다.

2. 관련 연구

2.1 기존의 캡챠 공격 방법들

텍스트 기반 캡챠를 공격하는 연구는 이전부터 다양한 방법으로 이루어지고 있었다. 그 결과 일반적으로 다양한 방어 기법이 적용되어 있는 텍스트 기반 캡챠를 공격하는 단계는 크게 3단계로 분류되고 있다[4, 6]. 먼저 전단계는 다음 단계인 세그멘트 단계가 보다 원활하게 수행될 수 있도록 캡챠 이미지를 그레이 스케일링(Gray-scaling)하여 행렬화하거나, 캡챠 이미지 내 배경이나 잡음 등을 제거한다. 다음으로 세그멘트 단계는 다음 단계인 분류 단계에서 개별 문자 인식을 가능하게 하기 위해서 문자 뭉치들을 각각의 문자로 분리한다. 이후 분류 단계에서는 세그멘트 단계에서 분리된 각각의 문자들을 Support Vector Machine(SVM)이나 신경망(Neural network) 등의 분류기(Classifier)를 학습하여 캡챠 이미지를 자동으로 인식한다.

Elie Bursztein[5] 등은 실제 캡챠 이미지에 적용되고 있는 분리방지 기법 중 캡챠의 문자열과 백그라운드 간 분리를 어렵게 만드는 기법인 Background confusion[5]을 3가지로 구분하고, 이를 공격하는 방법을 소개하였다. 이중 복잡한 배경을 이용한 방법은 선이나 다른 형태의 모양을 삽입하여 캡챠 이미지로부터 글자의 분리를 어렵게 하는 방법이다. 여기서는 게임 스크린샷들을 랜덤 배경으로 사용하고, 특정 색상을 글자에 적용한 Blizzard 캡챠를 예시로 들고 있으며, 이러한 경우에는 글자에 해당하는 색상을 제외한 모든 색상을 없앰으로서 글자를 추출할 수 있음을 보였다. 다음으로 색상의 유사성을 이용한 방법은 사람의 눈에는 구분되지만, RGB 색 공간에서는 유사한 색상을 백그라운드에 사용하여 배경과 글자의 분리를 어렵게 만든 기법이다. 여기서는 이러한 특성을 가지고 있는 Skyrock 캡챠[5]가 인간의 지각 능력과 유사한 HSV색 공간을 이용하는 경우 공격이 가능함을 보였다. 마지막으로 잡음을 이용하는 방법은 캡챠 이미지의 임의 영역에 잡음을 삽입하여 글자의 분리를 어렵게 하는 기법이다. 여기서는 Gibbs 알고리즘[5]을 사용하여 각 픽셀의 에너지를 주변 픽셀로부터 구하고, 특정 문턱치(Threshold) 이하의 에너지를 가지는 픽셀을 제거하는 방식으로 잡음을 제거할 수 있음을 설명하였다.

자연배경이 포함된 캡챠 이미지를 공격하기 위해 김근영 등의 논문[7]에서는 콘볼루션 필터링과 색상 반전을 이용하여 네이버 캡챠의 배경을 제거하였다. 콘볼루션 필터링은 입력 이미지의 각 픽셀과 해당 픽셀에 이웃한 영역의 값에 대해 특정 값을 가진 커널과의 행렬 곱셈 연산으로 이루어지며, 커널 값에 따라 이미지의 초점을 흐리게 만드는 블러링(Blurring)과 이미지의 윤곽을 또렷하게 만드는 샤프닝(Sharpening) 등을 원본 이미지에 적용할 수 있다. 여기서는 캡챠 이미지에 블러링-샤프닝을 두 번 반복 적용하여 자연배경을 제거하였고, 이후 세그멘트 단계에서는 숫자들이 모두 떨어져 위치하기 때문에 히스토그램을 이용하여 글자를 분리하였다.

또한, 김재환[8] 등은 자연배경이 포함된 캡챠 이미지를 공격하기 위해 특징 추출 단계에서 글자에 해당하는 포그라운드 정보와 자연배경에 해당하는 백그라운드 정보를 확보한 후 SVM을 사용하여 두 부류를 분리하였다. 이때 포그라운드 정보의 경우 백그라운드인 자연배경의 종류에 따라서 각기 상이한 밝기값을 가지기 때문에 캡챠 이미지별로 적절한 문턱치를 정하여 추출하였고, 백그라운드 정보는 글자가 중앙에 위치하는 특성을 이용해 캡챠 이미지의 가장자리에서 추출하였다. SVM를 이용한 특징 분리시 파라미터로는 앞서 [5]에서 언급한 HSV 색공간 중 S(채도)와 V(명도), 그리고 영상의 Texture를 나타내는 Local Binary Pattern(LBP) 정보를 사용하였다.

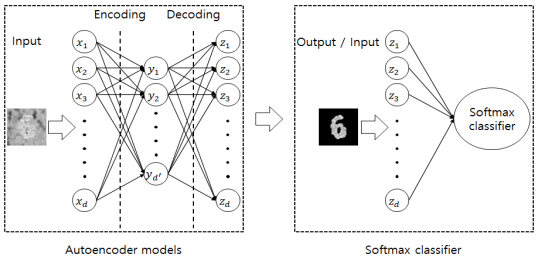
2.2 오토인코더를 이용한 유사 사례

인공신경망의 일종인 오토인코더는 주로 차원 축소를 통해 입력 데이터의 특징을 효과적으로 학습하거나, 깊은 신경망 구조를 비지도학습 방식으로 가중치를 초기화하여 효과적인 학습을 가능하게 하는 역할을 수행한다. 이러한 오토인코더 구조를 응용하여, Junyuan Xie[9] 등은 잡음제거 오토인코더와 희소 표현(Sparse coding)을 합친 Stacked Sparse Denoising Auto-encoders(SSDA)를 사용하여 배경이미지와 혼재된 문자 이미지를 제거하고 훼손된 이미지 영역을 복원(Inpainting)하였다. 배경이미지 복원을 위해 잡음제거 오토인코더(Denoising autoencoder)를 사용하였으며, 입력 데이터인 글자가 포함된 배경이미지로부터 목표 값인 원본 배경이미지를 학습하도록 하였다. 그 결과 학습 데이터와 유사한 패턴의 문자 이미지에 한해 문자 이미지를 제거하고 원본 배경 이미지에 가깝게 복원할 수 있었다. 여기서는 오토인코더 모델을 사용하여 제한적이지만 문자 이미지를 제거하고 파손된 부분을 복원할 수 있음을 보여주었으며, 이러한 이미지 복원 능력은 배경이미지를 제거하고 문자로만 이루어진 이미지 정보 추출로 응용 가능하다.

3. 제안하는 방법

3.1 개요

제안하는 오토인코더 모델을 적용한 자연배경 제거 및 숫자 분류 시스템의 구조는 그림 1과 같다.



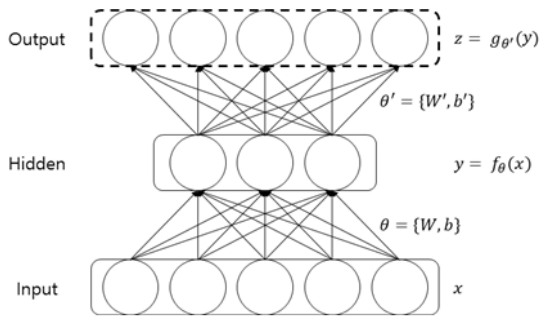
(그림 1) 제안하는 시스템의 구조
(Figure 1) A structure of the proposed system

제안하는 전체적인 시스템은 오토인코더 모델의 입력을 복원하는 단계와 오토인코더 출력을 입력으로 사용하여 1에서 9까지의 숫자를 예측하는 단계로 구성되어있으며, 각 단계는 구분하여 독립적으로 학습하였다.

이중 오토인코더 단계는 잡음인 자연배경을 제거하고 숫자 이미지를 복원하는 단계이다. 오토인코더 모델들을 학습 시 입력은 숫자와 자연배경, 잡음이 혼재된 이미지를, 목표 값은 숫자만 존재하는 이미지를 사용하였으며, 학습 과정을 통해 숫자 이미지를 제외한 자연배경을 제거하였다.

예측(Prediction) 단계는 오토인코더 모델의 출력인 자연배경이 제거된 숫자 이미지를 입력으로 사용하여 숫자 이미지를 분류하고 정확도를 측정하는 단계이다. 이때 파라미터를 학습하기 위한 교사 값은 One-hot encoding이 적용된 라벨을 사용하였고, 숫자 예측을 위한 활성화 함수는 Softmax classifier(SMC)를 사용하였다.

3.2 오토인코더



(그림 2) 오토인코더의 구조
(Figure 2) A structure of the autoencoder

오토인코더(Autoencoder, AE)는 인공신경망(Artificial Neural Network, ANN)의 일종으로 그림 2와 같이 인코딩-디코딩 과정을 하나의 은닉 계층(Hidden layer)을 이용하여 수행한다[10]. 오토인코더는 입력 계층(Input layer), 출력 계층(Output layer), 하나의 은닉 계층으로 구성되며, 입력 계층과 출력 계층은 일반적으로 동일하다. 학습방법은 일반적인 다중 계층 신경망(Multi-layer neural network)과 같이 역전파(Backpropagation) 알고리즘을 이용하여 학습한다. 이러한 오토인코더는 인코딩 과정에서 입력 벡터 $x \in [0,1]^d$를 식 $y = f_\theta(x) = s(Wx + b)$를 이용하여 은닉 계층의 표현인 $y \in [0,1]^{d'}$로 변환한다. 이때 $\theta = \{W, b\}$, W는 $d' \times d$ 차원의 가중치 행렬을, b는 바이어스 벡터를 의미한다. 이와 같이 인코딩된 은닉 계층의 표현인 y는 디코딩 과정에서 $y \in [0,1]^{d'}$를 식 $z = g_{\theta'}(y) = s(W'y + b')$를 이용하여 출력 벡터 $z \in [0,1]^d$

로 변환하며, 출력 z는 실제로는 입력에 해당되므로 복원(Reconstruction)된 벡터라고 표현한다. 여기서 $\theta' = \{W', b'\}$이고, 출력층과 입력층이 동일한 경우에는 가중치 행렬인 W'는 W^T가 된다.

오토인코더는 주어진 입력과 복원된 출력 간 손실(Loss)이 최소화되도록 학습되며, 최적화된 파라미터 $\theta^*, \theta^{*'}$를 유도하는 식 (1)은 다음과 같다[11].

$$\theta^*, \theta^{*'} = \arg min_{\theta,\theta'} \frac{1}{n}\sum_{i=1}^{n} L(x^{(i)}, z^{(i)}) = \arg min_{\theta,\theta'} \frac{1}{n}\sum_{i=1}^{n} L(x^{(i)}, g_{\theta'}(f_\theta(x^{(i)}))) \quad (1)$$

이때 손실 함수(Loss function)는 오토인코더의 입력 x와 출력 z의 값에 따라 사용가능하다. Mean Square Error (MSE, 평균제곱오차)함수인 L_{MSE}는 출력 z가 연속적인 값을 가지는 경우 사용하며, Cross Entropy(CE)함수인 L_{CE}는 출력이 이진 분류 시에 주로 사용한다[12]. 이러한 손실 함수 L_{MSE}, L_{CE}는 하나의 데이터에 대하여 다음 식 (2)와 식 (3)과 같이 각각 정의되며[11], 여기에서 k는 벡터 데이터의 k번째 요소를 의미한다.

$$L_{MSE}(x, z) = \| x - z \|^2 \quad (2)$$

$$L_{CE}(x, z) = -\sum_{k=1}^{d} [x_k \log z_k + (1-x_k)\log(1-z_k)] \quad (3)$$

위와 같은 학습 과정을 통해서 오토인코더는 입력 x와 출력 z를 항등 관계(Identity mapping)로 학습이 가능하다. 다만 효과적인 학습을 위해서 오토인코더의 은닉 계층에는 Undercomplete와 Overcomplete라는 제약 조건이 존재한다[13]. 전통적인 오토인코더는 은닉 계층의 벡터를 $d' < d$와 같이 입·출력 계층의 벡터의 수보다 줄임으로써 뉴런의 병목현상(Bottleneck)을 이용하며, 이와 같은 제약 조건을 Undercomplete라고 한다. 이러한 제약 조건은 차원 축소(Dimensionality reduction)를 통하여 입력 데이터에서 중요한 특징의 학습을 가능하게 한다[14]. 그리고 은닉 계층의 벡터가 $d' > d$와 같이 입·출력 계층의 벡터의 수보다 큰 경우 이와 같은 제약 조건을 Overcomplete라고 하며, 이러한 경우 은닉 계층에 희소 제약(Sparsity constraint)을 적용하여 신경망을 효과적으로 학습 가능하다[11, 15].

3.3 잡음제거 오토인코더

잡음제거 오토인코더(Denoising autoencoder)는 오토인코더의 입력 계층에서 입력 벡터 x 대신에 부분적으로 손상된 $\tilde{x}$ 값을 입력으로 사용하고, 교사학습으로 손상되지 않은 원래의 입력 값 x를 사용하여 손상된 입력인 $\tilde{x}$로부터 출력 z가 원래의 입력 x를 복원하도록 인코딩-디코딩 과정을 수행하는 인공신경망이다[11]. 잡음제거 오토인코더의 학습 방법은 오토인코더와 동일하며, 손상된 입력 값을 입력으로 사용하는 학습 방법은 입력 데이터로부터 흥미로운 구조를 찾는데 유용하고, 데이터의 손상 또는 변형으로부터 신경망을 강인하게 하는 특징이 있다.

3.4 적층 오토인코더

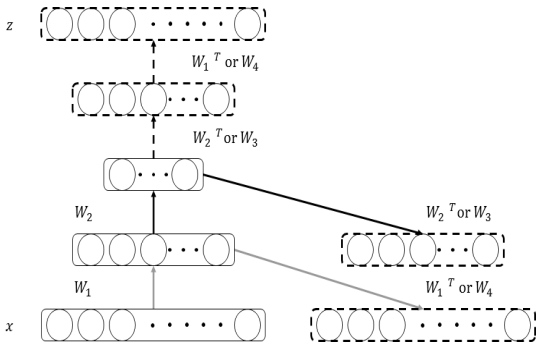
적층 오토인코더(Stacked autoencoder, SAE)는 다수의 은닉 계층을 이용하여 인코딩-디코딩 과정을 수행하는 인공신경망이다[13]. 적층 오토인코더 내 다수의 은닉 계층은 보다 복잡한 코딩을 학습하는 것을 가능하게하며, 그림 3과 같이 중앙의 은닉 계층을 기준으로 대칭인 구조를 가지고 있다.

그러나 적층 오토인코더와 같은 다중 계층 신경망은 은닉 계층이 깊어질수록 학습이 정체되는 Local minimum 이라는 문제가 발생한다. 이를 방지하기 위해 [14, 16]에서는 Deep Belief Network(DBN)의 학습에 Greedy layer-wise unsupervised training 알고리즘을 사용하여 깊이가 깊은 다중 계층 신경망에서도 학습이 가능함을 보였다. 이러한 학습 알고리즘은 적층 오토인코더에도 적용이 가능하며, [13]에서는 적층 오토인코더에서 Pre-training을 수행한 경우 수행하지 않은 경우보다 성능이 우수함을 보였다.

3.4.1 Greedy layer-wise unsupervised training

Greedy layer-wise unsupervised training은 DBN[14, 16]의 학습에 사용된 비지도학습(Unsupervised training) 알고리즘으로 깊은 다중 계층 신경망의 학습을 가능하게 하였다. 이러한 전략은 하나의 인접 계층씩 학습 과정을 반복하는 Pre-training과 Pre-training이 완료된 후 전체의 신경망에 대해 학습하는 Fine-tuning으로 구성된다. 이때 Pre-training은 각 층별 학습 단계를 통해 초기 입력 값의 정보를 보존한 가중치로 초기화하며, 그림 3과 같이 Pre-training을 거친 가중치는 다음 Pre-training의 입력으로

사용된다. 이러한 Greedy layer-wise unsupervised training은 역전파 알고리즘으로 학습하는 다층 오토인코더와 같은 깊은 신경망도 학습 가능하게 하는 효과가 있음이 경험적으로 증명되었다[10].



(그림 3) 적층 오토인코더와 Unsupervised pre-training
(Figure 3) Stacked autoencoder and unsupervised pre-training

3.5 Softmax classifier

Softmax classifier(SMC)는 클래스 분류를 위한 활성화 함수로 로지스틱 회귀(Logistic regression)에 대하여 정규화한 지도학습(Supervised training) 모델이다[17].

임의의 입력 x가 주어져있고, 클래스의 개수가 $k=1,\ldots,K$일 때, 식 $P(y=k|x)$에서 각각의 k에 대한 확률을 구하는 식 (4)는 다음과 같다.

$$softmax_\theta(x)=\frac{1}{\sum_{j=1}^{K}\exp(\theta^{(j)^T}x)}\begin{bmatrix}\exp(\theta^{(1)^T}x)\\ \exp(\theta^{(2)^T}x)\\ \vdots\\ \exp(\theta^{(K)^T}x)\end{bmatrix} \quad (4)$$

이때 출력 값은 K개의 클래스에 대한 각각의 확률이 K 차원의 벡터 형태로 나오며, 이들의 총합은 1이 된다. 그리고 학습은 손실 함수를 최소화시키는 가중치 θ를 구하는 방식으로 이루어지며, 손실함수로는 Cross Entropy를 사용한다.

4. 실 험

4.1 실험 데이터 특성

실험 데이터는 네이버에서 사용하고 있는 캡챠 이미지 중에서 그림 4와 같이 하나의 숫자와 자연배경이 혼재되어 있는 특정 영역을 대상으로 하였으며, 테두리(Edge)만을 이용하여 숫자를 표현한 글씨체는 제외하였다. 실험을 위해 해당 사이트의 캡챠 이미지[18]를 다운받아 숫자와 자연배경이 혼재된 영역을 추출하였다. 다운받은 캡챠 이미지 중 실험에 사용한 숫자 이미지는 1에서 9까지의 숫자로 구성되어있고, 다수의 폰트가 존재하며, 숫자의 기울기와 크기가 각각 다른 특성을 가진다.

(그림 4) 숫자와 자연배경이 혼재된 이미지
(Figure 4) Mixed numbers and natural backgrounds

4.1.1 학습 데이터

학습 데이터는 입력 데이터인 숫자와 자연배경이 혼재되어 있는 이미지, 출력 데이터인 숫자만 존재하는 이미지, 교사 값인 해당 숫자의 라벨로 구성되어있다. 학습 이미지를 제작하기 위해 자연배경 이미지는 네이버 캡챠에서 500개 추출하였으며, 숫자 이미지는 네이버 캡챠에서 HSV 색공간의 채도(Saturation)와 밝기(Value)값으로 자연배경과 글자 분리가 가능한 특성[8]을 이용하여 숫자와 자연배경이 혼재되어 있는 원본 이미지에 문턱치 기법을 적용하여 숫자의 형태를 구성하고 있는 픽셀의 위치를 찾아내고, 해당 픽셀의 채도 값과 밝기 값을 1300개 추출하였다.

하나의 캡챠 데이터는 앞서 생성한 자연배경 이미지와 숫자 이미지를 기준 데이터로 하여서, 임의로 각각 하나씩 선택하여 입력 데이터를 생성하였다. 생성방법은 선택된 자연배경 이미지에서 각 픽셀의 HSV 값을 각각 (H, S, V), 선택된 숫자 이미지에서 각 픽셀의 HSV 값을 각각 (H', S', V')라고 했을 때, 조합된 캡챠 이미지는 해당 픽셀이 배경일 경우 (H, S, V), 해당 픽셀이 숫자의 일부분일 경우 (H, S', V')가 되도록 하였다. 이때 S'와 V'값은 해당 숫자 이미지 전체의 최대 값과 최소 값 사이에 존재하는 임의의 값을 가지도록 하였다. 그 다음으로, Canny Edge Detection[19]을 사용하여 숫자 이미지의 테두리를 찾아서, 이중 1 ~ 10%의 테두리 픽셀과 테두리 인접 픽셀의 채도와 밝기 값을 해당 숫자 이미지의 최대 값과 최소 값 사이의 임의의 값을 가지게 하여 앞서 추출

했던 숫자 이미지에 다양한 형태의 잡음을 추가하였다. 그리고 숫자 이미지에 대해 기울기는 -20° ~ 20°를 적용하여서, 원래의 숫자 이미지를 변형하였다. 생성된 HSV 형태의 이미지는 그레이 스케일링을 적용한 후 신경망에서 학습 가능한 형태인 0에서 1사이의 값으로 일반화하였다. 출력 데이터는 입력 데이터 생성 시 선정된 숫자 이미지를 사용하였으며, 해당 숫자를 의미하는 교사 값은 임의로 선택된 숫자와 일치하도록 생성하였다.

4.1.2 테스트 데이터

테스트 데이터는 다운받은 캡챠 이미지에서 숫자와 자연배경이 혼재된 영역을 변형 없이 그대로 사용하였으며, 캡챠 이미지에 존재하는 1에서 9까지의 숫자를 각각 100개씩 임의로 추출하여 총 900개의 이미지를 테스트에 사용하였다.

4.2 실험 환경

실험을 위해 사용한 하드웨어와 운영체제의 환경은 다음 표 1과 같다.

(표 1) 개발 환경
(Table 1) Development environment

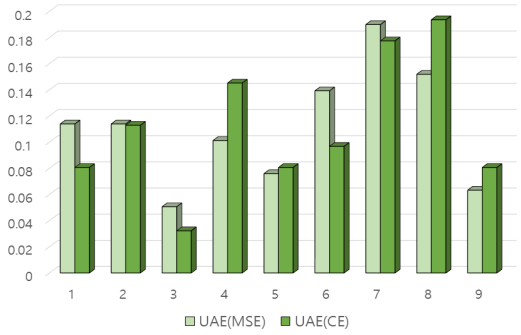
CPU	AMD Ryzen 7 1800X Eight-Core Processor
RAM	64GB
GPU	NVIDIA GeForce GTX 1080 Ti
OS	Ubuntu 16.04

실험은 Python 기반의 Tensorflow를 사용하여 진행하였으며, 1.0 버전을 이용하였다. 학습 데이터 생성 시 HSV 색 공간 변환 및 Canny edge detection을 적용하기 위해 OpenCV를 사용하였으며, 오토인코더의 구현을 위해서 [20]을 참고하였다.

4.3 실험 결과와 분석

실험에 사용된 오토인코더의 모델은 입·출력보다 적은 개수의 은닉노드 한층을 가진 오토인코더(Undercomplete autoencoder, UAE), 입·출력보다 많은 개수의 은닉노드 한층을 가진 오토인코더(Overcomplete autoencoder, OAE), 은닉 계층의 개수가 많은 적층 오토인코더(SAE)이다. 오토인코더의 손실 함수로는 [21]을 참고하여 Mean Square Error(MSE)와 Cross Entropy(CE)를 사용하였으며, 이를 통해 오토인코더 모델과 사용된 손실 함수에 따른 복원된 이미지의 형태와 정확도를 확인하였다. 모든 모델은 학습 횟수가 동일하며, 이때 적층 오토인코더의 학습 횟수는 초기 가중치를 결정하는 Pre-training 단계를 제외한 Fine-tuning 단계의 학습 횟수를 말한다.

4.3.1 Undercomplete autoencoder(UAE)



(그림 5) 테스트 데이터에서 UAE 모델의 숫자별 오분류율
(Figure 5) Probability of misclassifications by each number in UAE model in test data

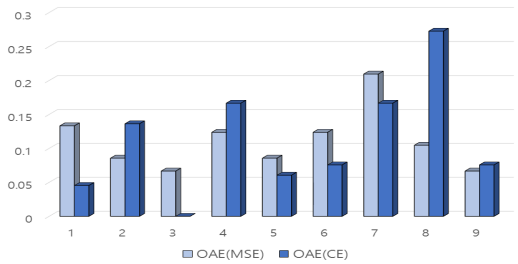
(그림 6) UAE로 정상적으로 복원·분류된 이미지 샘플
(Figure 6) Samples of images normally restored and classified by UAE

(그림 7) UAE로 비정상적으로 복원·분류된 이미지 샘플
(Figure 7) Samples of images abnormally restored and classified by UAE

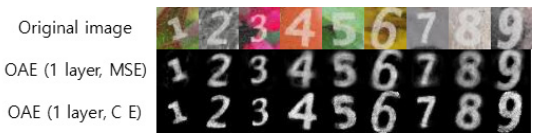
UAE 모델에서 사용한 입·출력 계층의 노드(Node) 개수는 1600개, 은닉 계층의 노드 개수는 300개이다. 이러한 구조의 UAE로 복원된 이미지는 각각 그림 6, 7이며, 이중 그림 6은 숫자가 정상적으로 복원·분류된 이미지이고, 그림 7은 숫자가 비정상적으로 복원되거나 오분류된

이미지이다. 실험 결과 테스트 데이터에 대하여 각각 91.2%(MSE), 93.1%(CE)의 정확도를 보였으며, 그림 5와 같이 UAE는 손실 함수와 상관없이 숫자 '7'과 숫자 '8'에 대한 오분류 비율이 가장 컸다.

4.3.2 Overcomplete autoencoder(OAE)



(그림 8) 테스트 데이터에서 OAE 모델의 숫자별 오분류율
(Figure 8) Probability of misclassifications by each number in OAE model in test data



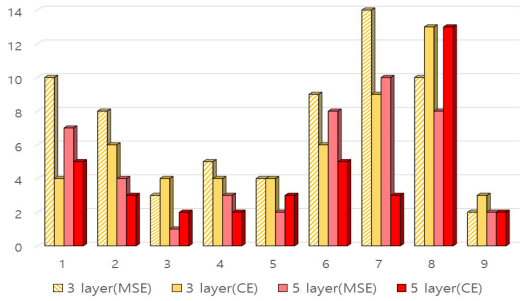
(그림 9) OAE로 정상적으로 복원·분류된 이미지 샘플
(Figure 9) Samples of images normally restored and classified by OAE

(그림 10) OAE로 비정상적으로 복원·분류된 이미지 샘플
(Figure 10) Samples of images abnormally restored and classified by OAE

OAE 모델에서 사용한 입·출력 계층의 노드 개수는 1600개, 은닉 계층의 노드 개수는 5000개이며, 여기서는 손실 함수에 희소 조건을 적용하지 않았다. 이러한 구조의 OAE로 복원된 이미지는 각각 그림 9, 10이며, 이중 그림 9는 숫자가 정상적으로 복원·분류된 이미지이고, 그림 10은 숫자가 비정상적으로 복원되거나 오분류된 이미지이다. 그림 10에서 숫자 '3'의 경우 모두 정상 복원·분류되었기 때문에 생략되었다. 실험 결과 테스트 데이터에 대하여 각각 88.3%(MSE), 92.6%(CE)의 정확도를 보였으며, 그림 8과 같이 OAE는 손실 함수가 MSE인 경우 숫자 '7'과 숫자 '1', 손실함수가 CE인 경우 숫자 '8'과 숫자 '4'에 대한 오분류 비율이 가장 컸다.

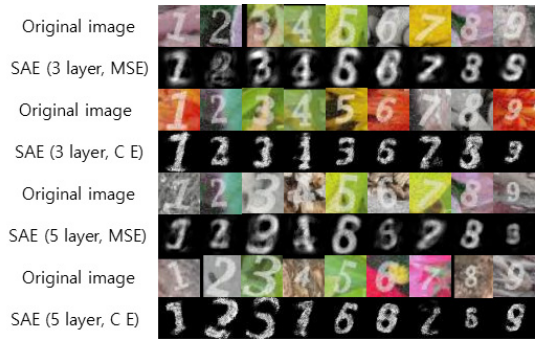
4.3.3 Stacked autoencoder(SAE)

SAE 모델에서 사용한 입·출력 계층의 노드 개수는 1600개이다. 은닉 계층의 노드 개수는 3개의 은닉 계층의 경우 1000-700-300이며, 5개의 은닉 계층의 경우 1300-1000-700-500-300이다. 이러한 구조의 SAE로 복원된 이미지는 각각 그림 12, 13이며, 이중 그림 12는 숫자가 정상적으로 복원·분류된 이미지이고, 그림 13은 숫자가 비정상적으로 복원되거나 오분류된 이미지이다. 실험 결과 테스트 데이터에 대하여 3개의 은닉 계층을 가진 적층 오토인코더는 각각 92.8%(MSE), 94.1%(CE)의 정확도를 보였으며, 5개의 은닉 계층을 가진 적층 오토인코더는 각각 95%(MSE), 96.2%(CE)의 정확도를 보였다. 그림 11과 같이 SAE는 공통적으로 숫자 '7'과 숫자 '8'에 대한 오분류 비율이 가장 컸다.



(그림 11) 테스트 데이터에서 SAE 모델의 숫자별 오분류율
(Figure 11) Probability of misclassifications by each number in SAE model in test data

(그림 12) SAE로 정상적으로 복원·분류된 이미지 샘플
(Figure 12) Samples of images normally restored and classified by SAE



(그림 13) SAE로 비정상적으로 복원·분류된 이미지 샘플
(Figure 13) Samples of images abnormally restored and classified by SAE

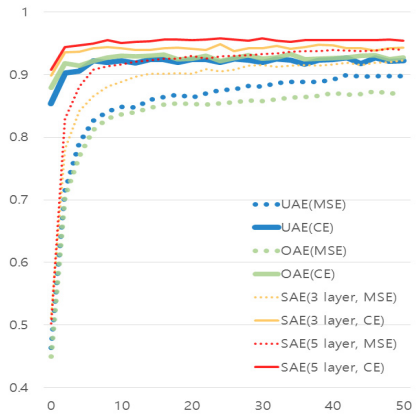
4.3.4 종합 분석

(표 2) 오토인코더 모델별 테스트 정확도
(Table 2) Test accuracy by autoencoder model

사용 알고리즘	손실 함수(AE)	정확도
UAE	MSE	91.2%
UAE	CE	93.1%
OAE	MSE	88.3%
OAE	CE	92.6%
SAE (3 layer)	MSE	92.8%
SAE (3 layer)	CE	94.1%
SAE (5 layer)	MSE	95.0%
SAE (5 layer)	CE	96.2%

앞에서 실시한 실험 결과를 바탕으로 다음과 같은 결과를 얻을 수 있었다. 먼저, 오토인코더 모델별 정확도를 나타내는 표 2와 학습 시간에 따른 정확도를 나타내는 그림 14에서 알 수 있듯이 1개의 은닉 계층보다는 3개의 은닉 계층에서, 3개의 은닉 계층보다는 5개의 은닉 계층을 가진 오토인코더에서 테스트 데이터에 대한 정확도가 높아졌다. 또한, 오토인코더의 손실 함수로는 Cross Entropy를 사용한 결과가 Mean Square Error를 사용한 결과보다 정확도가 더 높았다.

그 이유는 오토인코더 모델과 손실 함수에 따른 이미지 복원 결과를 나타내는 그림 15, 16과 같이 Cross Entropy를 사용한 경우 분류단계의 입력으로 사용되는 숫자 이미지가 보다 선명하게 복원되어 실제 분류 단계의 학습 시 영향을 끼쳤기 때문이다. 또한 희소 조건을 적용하지 않은 OAE는 UAE나 SAE에 비해 은닉 노드 개수는 많지만, 테스트 데이터에 대한 정확도가 떨어지는 것을 확인할 수 있었다.



(그림 14) 학습 횟수에 따른 테스트 정확도
(Figure 14) Test accuracy by number of learnings

(그림 15) 오토인코더 모델로부터 복원된 이미지
(Figure 15) Image restored with autoencoder model

(그림 16) 오토인코더 모델로부터 복원된 이미지
(Figure 16) Image restored with autoencoder model

4.3.5 다른 방법들과 비교

위 실험에서는 오토인코더 모델을 사용하여 하나의 숫자와 자연배경으로 이루어진 캡챠 이미지에서 자연배경을 제거하고 이를 입력으로 하여 정확도를 측정하였다.

선행 연구와의 성능 비교가 필요하지만, [7, 8] 모두 숫자열에 대한 정확도를 측정하고 있고, 특히 [7]의 경우 자연배경은 다르지만 같은 숫자열을 가진 이미지 30개를 한 세트로 분석하고 판독하여 정확도를 측정하고 있다. 또한, 현재의 네이버 캡챠는 문자의 종류와 폰트가 다양해지고, 문자의 위치가 랜덤으로 위치하며, 문자와 문자간 분리방지기법이 적용되고 있는 등 데이터의 특성이 과거와 많이 달라졌기 때문에 앞서 소개했던 네이버 캡챠를 공격한 사례와 단순 비교를 하기에는 무리가 따른다.

따라서, 객관적인 성능 비교는 어렵지만 본 실험의 의의는 네이버 캡챠에서 자연배경을 삽입하는 방식으로 적용하고 있는 문자와 배경 간 분리방지기술에 대해 오토인코더 모델을 이용하여 캡챠 이미지내 다양한 자연배경을 효과적으로 제거하였으며, 이를 입력으로 받는 단일계층 신경망(Single-layer neural network)과 소프트맥스 함수를 사용하여 96.2%의 높은 확률로 개별 숫자 인식을 성공했다는 점이다.

5. 결 론

본 논문은 오토인코더를 이용하여 숫자와 자연배경으로 구성되어 있는 이미지에서 자연배경을 제거함으로서 숫자를 인식하는 정확도를 높이는 방법을 연구하였다.

제안하는 방법은 다양한 오토인코더의 모델을 이용하여 자연배경이 제거된 숫자 이미지를 복구하고, 이를 입력으로 받는 Softmax Classifier를 사용하여 숫자별로 분류하는 형태이다. 실험 결과 오토인코더 모델에 층의 개수가 많고, 손실 함수로는 Cross Entropy를 사용하는 경우 자연배경을 효과적으로 제거하고, 이러한 결과가 숫자를 인식하는 정확도에 영향을 끼침을 확인하였다.

향후 연구로는 오토인코더와 같은 인공신경망을 사용하여 숫자가 아닌 다른 문자에 대해서도 자연배경과 같은 잡음이 제거된 형태의 이미지를 생성 및 분류하는 것과 이를 사용하여 실제로 사용되고 있는 다양한 캡챠 이미지 인식에 사용하는 연구가 가능하다.

참고문헌(Reference)

[1] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, “The End is Nigh: Generic Solving of Text-based CAPTCHAs,” Usenix Woot, 2014. https://www.usenix.org/node/185129

[2] https://www.google.com/recaptcha/intro/invisible.html

[3] B. M. Powell, E. Kalsy, G. Goswami, M. Vatsa, R. Singh, and A. Noore, “Attack-Resistant aiCAPTCHA using a Negative Selection Artificial Immune System,” urity and Privacy Workshops (SPW), IEEE, pp. 1-6, 2017. https://doi.org/10.1109/SPW.2017.22

[4] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski, “Computers beat humans at single character recognition in reading based human interaction proofs (HIPs),” in Proc. of Second Conf. Email Anti-Spam, 2005. https://www.microsoft.com/en-us/research/wp-content/uploads/2005/01/CEAS2005Final.doc

[5] E. Bursztein, M. Martin, and J. C. Mitchell, “Text-based CAPTCHA strengths and weaknesses,” in Proc. of 18th ACM Conf. Comput. Commun. Secur., ISBN: 978-1-4503-0948-6, pp. 125-138. 2011. https://doi.org/10.1145/2046707.2046724

[6] C. Cruz-Perez, O. Starostenko, F. Uceda-Ponga, V. Alarcon- Aquino, and L. Reyes-Cabrera, “Breaking reCAPTCHAs with unpredictable collapse: Heuristic character segmentation and recognition,” Pattern Recognition, vol. 7329, pp. 155-165, 2012. https://link.springer.com/chapter/10.1007/978-3-642-31149-9_16

[7] K. Kim, D. Shin, K. Lee and D. Nyang, “CAPTCHA Analysis using Convolution Filtering,” Journal of The Korea Institute of Information Security & Cryptology, Vol. 24, no. 6, pp. 1129-1138, 2014. http://dx.doi.org/10.13089/JKIISC.2014.24.6.1129

[8] J. Kim, S. Kim, and H. J. Kim, “Breaking character and natural image based CAPTCHA using feature classification,” Journal of The Korea Institute of Information Security & Cryptology, Vol. 25, no. 5, pp. 1011-1019, 2015. http://dx.doi.org/10.13089/JKIISC.2015.25.5.1011

[9] J. Xie, L. Xu, and E. Chen, “Image Denoising and Inpainting with Deep Neural Networks,” Nips, pp. 1-9, 2012. https://papers.nips.cc/paper/4686-image-denoising-and-inpainting-with-deep-neural-networks

[10] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy Layer-Wise Training of Deep Networks,” Adv. Neural Inf. Process. Syst., Vol. 19, no. 1, pp. 153-160, 2007.

[11] P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in Proc. of 25th Int. Conf. Mach. Learn. - ICML ’08, pp. 1096-1103, 2008. http://machinelearning.org/archive/icml2008/papers/592.pdf

[12] A. Ng, ″CS229 Lecture notes,″ CS229 Lecture notes, pp. 1-30, 2000.

[13] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and

P.A. Manzagol, “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion,” J. Mach. Learn. Res., Vol. 11, pp. 3371-3408, 2010. http://www.jmlr.org/papers/v11/vincent10a.html

[14] G. E. Hinton and R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” Science, Vol. 313, no. 5786, pp. 504-507, 2006. https://doi.org/10.1126/science.1127647

[15] A. Ng, “Sparse autoencoder,” CS294A Lect. notes, 2011, pp. 1-19.

[16] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets,” Neural Comput., Vol. 18, no. 7, pp. 1527-54, 2006. https://www.cs.toronto.edu/~hinton/absps/fastnc.pdf

[17] http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/

[18] https://nid.naver.com/login/image/captcha/nhncaptchav4.gif?key=

[19] J. Canny, “A Computational Approach to Edge Detection,” IEEE Trans. Pattern Anal. Mach. Intell., Vol. PAMI-8, no. 6, pp. 679-698, 1986. https://doi.org/10.1109/TPAMI.1986.4767851

[20] A. Géron, "Hands on Machine Learning with scikit-learn and Tensorflow," 2017

[21] T. Amaral, L. M. Silva, L. A. Alexandre, C. Kandaswamy, J. M. Santos, and J. M. De Sá, “Using different cost functions to train stacked auto-encoders,” Artificial Intelligence (MICAI), 2013 12th Mexican International Conference on, pp. 114-120, 2013. https://doi.org/10.1109/MICAI.2013.20

◐ 저 자 소 개 ◑

전 재 승(Jae-seung Jeon)
2012년 서경대학교 컴퓨터과학과(이학사)
2016년~현재 고려대학교 정보보호대학원 석사과정
관심분야 : 시스템 보안, 기계 학습
E-mail : damulart@korea.ac.kr

문 종 섭(Jong-sub Moon)
1981년~1985년 금성 통신 연구소 연구원
1991년 Illinois Institute of technology 전산학 박사
1993년~현재 고려대학교 전자 및 정보공학부 교수
관심분야 : 생체인식, 운영체제, 침입탐지
E-mail : jsmoon@korea.ac.kr