JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 28 No. 5, pp. 29-38, May 2023
https://doi.org/10.9708/jksci.2023.28.05.029

Prediction of Doodle Images Using Neural Networks

Hae-Chan Lee*, Kyu-Cheol Cho*

*Student, Dept. of Computer Science, Inha Technical College, Incheon, Korea
*Professor, Dept. of Computer Science, Inha Technical College, Incheon, Korea

[Abstract]

Doodles, often possess irregular shapes and patterns, making it challenging for artificial intelligence to mechanically recognize and predict patterns in random doodles. Unlike humans who can effortlessly recognize and predict doodles even when circles are imperfect or lines are not perfectly straight, artificial intelligence requires learning from given training data to recognize and predict doodles. In this paper, we leverage a diverse dataset of doodle images from individuals of various nationalities, cultures, left-handedness, and right-handedness. After training two neural networks, we determine which network offers higher accuracy and is more suitable for doodle image prediction. The motivation behind predicting doodle images using artificial intelligence lies in providing a unique perspective on human expression and intent through the utilization of neural networks. For instance, by using the various images generated by artificial intelligence based on human-drawn doodles, we expect to foster diversity in artistic expression and expand the creative domain.

▸ **Key words**: Deep Learning, Neural Network, Doodle

[요 약]

낙서는 대부분 불규칙한 형태와 패턴을 갖추고 있기에, 인공지능이 불규칙한 낙서를 기계적으로 패턴을 인식하고 예측하기란 매우 어렵다. 만약 그려진 원이 완벽한 동그라미가 아니거나, 직선도 완전히 일직선이 아닐 경우, 인간은 별도의 학습 과정 없이도 낙서를 인식하고 예측할 수 있다. 이에 반해, 인공지능은 주어진 학습 데이터로 패턴을 학습해야만 낙서를 인식하고 예측한다. 본 논문은 국적, 문화, 왼손잡이 또는 오른손잡이 등 관계없이 다양한 사람들의 낙서 이미지 데이터셋을 활용한다. 그리고 두 가지의 신경망 학습을 거친 뒤, 어느 신경망이 더 높은 정확도를 제공하는지, 낙서 이미지 예측에 더 적합한지 대한 여부를 결정한다. 인공지능을 통한 낙서 이미지 예측을 하는 이유는 신경망을 활용함으로써, 인간의 표현과 의도에 대한 독특한 관점을 제공할 수 있기 때문이다. 가령, 인간이 그린 낙서에 대해 인공지능이 제공하는 다양한 이미지를 활용하여 예술적인 표현의 다양성을 촉진하고 창작 영역을 넓히는 데 기여할 것으로 기대한다.

▸ **주제어**: 딥러닝, 신경망, 낙서

• First Author: Hae-Chan Lee, Corresponding Author: Kyu-Cheol Cho
*Hae-Chan Lee (devgisoun@gmail.com), Dept. of Computer Science, Inha Technical College
*Kyu-Cheol Cho (kccho@inhatc.ac.kr), Dept. of Computer Science, Inha Technical College
• Received: 2023. 03. 07, Revised: 2023. 05. 22, Accepted: 2023. 05. 22.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

인공지능의 연구 가치는 신경망의 충분한 학습을 통해 인간보다 빠르고 뛰어난 처리 능력을 지니고 있다는 점이다. 인공지능은 자율주행 자동차, 사람의 얼굴 표정, 행동 인식, 영상 및 이미지 분석 등 실생활에서 다양하게 사용되고 있다[1]. 예시로 iPhone의 Face ID는 얼굴 이미지 인식 딥러닝 기술이 접목되어있는 보안 기술[2]이다. 사용자가 수염을 기르거나 마스크를 착용한 상태의 얼굴도 학습하고 사용자가 계속 사용할수록 인식률이 향상되고 속도도 빨라지는 이점을 드러내고 있다.

기존 딥러닝 기반 영상 처리 표현 방법의 대부분은 사진과 비디오를 위해 설계되었다. 특히, 낙서 스케치를 인식하고 분류하는 능력은 인공지능의 발전에 중요한 영향[3]을 준다. 예를 들어, 광학 문자 인식(Optical Character Recognition, OCR) 같은 컴퓨터 비전 및 패턴 인식 연구에서 노이즈가 많은 데이터셋에 대한 분류기로 큰 이익을 얻을 수 있지만 쉽게 탐색할 수 있는 사진과 달리 스케치는 일일이 그리는 작업이 필요하다. 이러한 이유로 데이터 이슈가 스케치의 경우 특히 두드러진다. 무엇보다 일반적으로 낙서는 개인의 단순 스케치이면서 창작의 한 형태이기 때문이다.

낙서는 그리는 사람마다 국적, 문화, 성격 등 다양한 특성이 있어 각자의 개성이 담긴 다른 형태의 낙서 결과가 나타난다. 다양한 형태의 고양이 스케치가 제시되었을 때, 사람은 쉽게 고양이를 그렸다고 판단할 수 있지만, 컴퓨터는 별도의 학습 과정이 없다면 주어진 모든 고양이 그림을 다른 클래스로 인식할 것이다. 그렇기에 신경망을 통한 다양한 유형의 낙서를 학습하여 정확한 예측을 할 필요가 있다.

본 연구는 인공지능이 낙서 스케치를 얼마나 정확하게 예측할 수 있는지 분석한다. 딥러닝 기반의 신경망인 CNN(Convolutional Neural Network), RNN(Recurrent Neural Network) 기반 모델을 제작해, 학습 후 성능을 비교한다. 낙서 스케치를 픽셀 또는 시퀀스로 해석했을 때, 검증 정확도의 결과와 낙서 이미지 예측에 더 적합한 신경망의 여부를 결정한다.

제1장을 제외한 본 논문의 구성은 다음과 같이 이루어져 있다. 제2장은 실험에 사용할 신경망과 낙서 이미지 처리 방법에 대한 이론과 관련 연구 사례를 설명한다. 제3장은 낙서 이미지 데이터셋을 수집하고 신경망별 데이터 전처리 과정을 설명한다. 제4장은 학습된 모델의 검증 정확도를 비교하고 캔버스 UI에 직접 낙서를 스케치한 뒤, 낙서 이미지 예측을 실험하고 결론을 내린다.

II. Related Works

1. Image Analysis Using CNN

CNN은 이미지 내에서 특징을 인식하고 분류하여 빠르게 학습할 수 있는 것으로 컴퓨터 비전 분야에 적합하여 많이 사용되는 신경망이다. 일반적인 CNN은 그림1과 같이 합성곱과 신경망을 결합한 구조이다. 합성곱 레이어(Convolution Layer)가 필터를 통해 이미지의 특징을 추출하면 풀링 레이어(Pooling Layer)에서 특징을 강화하고 이미지 사이즈를 줄이는 과정을 반복하여 이미지의 특징을 강조한다. 추출된 이미지의 특징은 전결합층(Fully Connected Layer)에서 분류를 수행[4]한다.

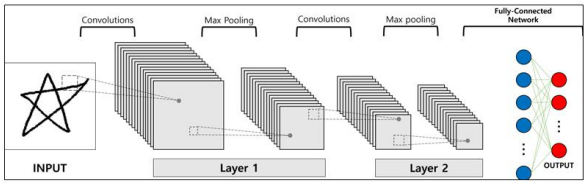

Fig. 1. Traditional CNN Structure [5]

최근 CNN은 텍스트 분류 작업에도 사용되고 있다. 이미지의 특징을 추출하는 것처럼 CNN 모델을 NLP(Natural Language Processing)[6]에 적용하면 텍스트의 특정 위치에서 특징을 추출할 수 있다.

2. Sketch-RNN

구글의 인공지능 연구팀은 RNN을 사용해 스케치 도면을 모델링 하는 방법인 Sketch-RNN을 제시[7]하였다. Sketch-RNN은 공통 객체의 스트로크 기반 스케치를 구성할 수 있는 RNN이다. 이 신경망은 수백 개의 종류를 표현하는 수천 개의 낙서 이미지에 대해 훈련된다. 그림2는 Sketch-RNN의 구조를 도식화한 것이다

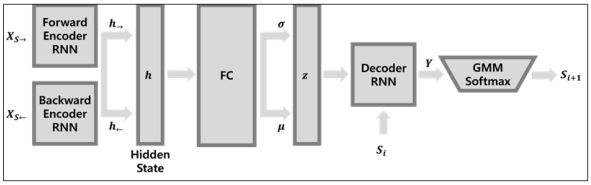

Fig. 2. Schematic Diagram of Sketch-RNN [7]

Sketch-RNN은 양방향 RNN(Bidirectional Recurrent Neural Networks)[8]의 VAE(Variational Auto-Encoder)를 이용하여 입력 스케치로부터 잠재 벡터를 출력하고 Decoder에 넣어 새로운 스케치를 얻는다. 이

렇게 얻은 스케치는 펜을 제어하는 일련의 동작으로 취급되며, $[\triangle x, \triangle y, p_1, p_2, p_3]$로 이루어진 벡터들의 시퀀스로 표현된다. $\triangle x$, $\triangle y$는 이전 point에서부터 펜의 x 및 y축의 오프셋 거리, p_1은 펜을 종이에 그대로 대고 있는지, p_2는 펜을 종이에서 들었는지, p_3은 스케치를 마쳤는지에 대한 여부를 나타낸다.

스케치를 시퀀스로 처리하여 사용자의 미완성 스케치 결과를 예측하고 사용자가 낙서를 그리는 도중에 사용자의 의도를 파악하여 모델이 완성될 스케치의 형태를 제안할 수 있음을 시사한다.

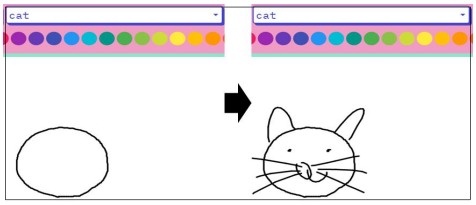

Fig. 3. Example of The Result of 'Magic Sketchpad Game' [9]

구글의 오픈소스 연구 프로젝트 콘텐츠인 'Magic Sketchpad Game'[9]는 Sketch-RNN 모델을 기반으로 제작되었다. 개체 주제를 선택한 뒤 스케치를 중단하면 Sketch-RNN이 낙서를 추측하고 중단한 부분부터 드로잉 작업을 진행하여 그림을 완성 시켜주는 과정을 보여준다. 그림3에서 나타나듯 'Cat'이라는 주제에서 사용자가 원 형태로 스케치 중 동작을 멈추면 Sketch-RNN은 작업을 인계받아 중단한 위치를 기반으로 나머지 낙서를 추측하고 낙서를 완성한다.

3. Thresholding

스레시홀딩(Thresholding)은 바이너리 이미지를 만드는 가장 대표적인 방법이다. 바이너리 이미지란 흑(0)과 백(255)으로만 표현하는 이미지를 의미한다. 즉 스레시홀딩은 여러 값을 어떤 임계점을 기준으로 하여 두 가지 분류로 나누는 방법을 의미한다. 스레시홀딩을 통해 바이너리 이미지를 만드는 목적은 배경과 객체를 구분하고 관심 영역과 비관심 영역을 구분하는 데 있다. 스레시홀딩은 전역 스레시홀딩(Global Thresholding)과 적응형 스레시홀딩(Adaptive Thresholding)으로 나뉜다.

(1) Global Thresholding

전역 스레시홀딩은 흑백 이미지를 간단히 이진화하는 데 적합한 방식[10]으로 이미지의 모든 픽셀에 동일한 임곗값이 적용된다. 픽셀값이 임곗값을 넘으면 흰색, 그렇지 않다면 검은색을 반환한다. 그림4는 전역 스레시홀딩의 결과이다. 우선, 픽셀값이 0에서 255로 변경되는 그라데이션 이미지를 준비하였다. 이미지의 임곗값을 픽셀 최댓값(255)의 $1/2$인 127로 지정한다. 만약 원본 이미지의 픽셀값이 지정한 임곗값 이상이면 255, 그렇지 않다면 0으로 픽셀값을 변환한다.

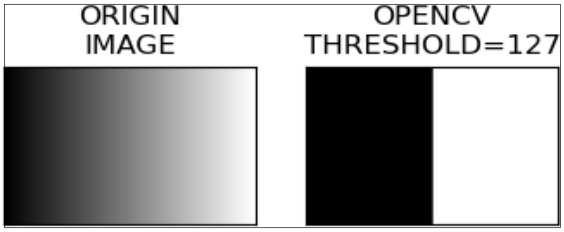

Fig. 4. As a Result of Global Thresholding of Gradient Image With OpenCV (Threshold : 127)

바이너리 이미지를 만들 때 가장 중요한 점은 적절한 임곗값 선택이다. 일반적으로는 가장 적합한 임곗값을 찾기 위해 여러 번의 적용과 직접 판단이 필요하다. 하지만 이러한 방식은 복잡한 형태의 이미지에서 정확하게 감지하는 것이 어렵기에, 오츠의 알고리즘(Otsu's method)[11]을 통해 한 번에 임계값을 찾을 수 있도록 적용하였다. 오츠의 알고리즘은 이미지를 흑과 백, 두 개의 클래스로 나누고, 각 클래스의 분산을 최소화하기 위해 이미지의 히스토그램을 사용하는 방법이다. 클래스의 분산은 아래 수식(1)을 통해 정의할 수 있다.

$$\sigma_w^2(t) = w_0(t)\sigma_0^2(t) + w_1(t)\sigma_1^2(t) \qquad \cdots (1)$$

수식 (1)은 클래스 내 분산이 최소가 되는 임의의 임곗값 $t(0 \sim 255)$을 찾아 선택한다. $\sigma_w^2(t)$는 t에서의 가중분산, w_0과 w_1은 각 클래스의 가중치이며, $\sigma_0^2(t)$와 $\sigma_1^2(t)$는 각 클래스의 분산이다. 그림5는 오츠의 알고리즘을, OpenCV를 통한 스레시홀딩 기법에서 적용[12]한 결과이다.

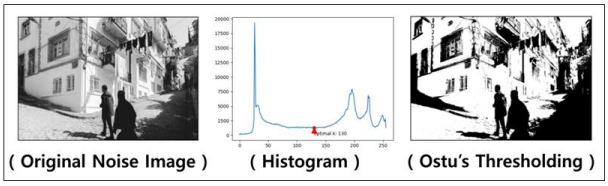

Fig. 5. Histogram and Result of Image Through Global Thresholding Using Otsu's Method.

오츠의 알고리즘을 적용하여, 입력 이미지의 히스토그램에 대해서 임곗값을 0부터 255까지 증가시킨다. 최적의 임곗값이 130으로 출력되었으므로 클래스 내 분산이 최소가 되는 임곗값은 130임을 알 수 있다.

(2) Adaptive Thresholding

원본 이미지에서 조명이 일치하지 않거나, 배경색이 다양한 상황에 전역 스레시홀딩에서 적용한 것처럼 하나의 임곗값으로 선명한 이미지를 만들어내기 힘든 경우, 적응형 스레시홀딩을 적용한다. 적응형 스레시홀딩은 조명에 따른 공간 변화를 고려하는 기법이다. 입력 적분 이미지를 이용한 실시간 적응형 이진화 기법[13]으로 이미지의 조명 변화에 강하다.

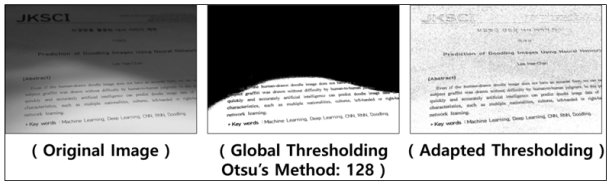

Fig. 6. Comparison of Image With Global Thresholding and Adaptive Thresholding

그림6은 OpenCV를 통한 스레시홀딩 적용 결과이다. 두 번째 이미지는 전역 스레시홀딩의 결과로, 임곗값이 128로 지정되어 픽셀값이 이보다 낮은 영역은 전부 0, 높은 영역은 255로 표현되어 조명이 어두운 부분을 확인하기 어렵다. 하지만 적응형 스레시홀딩의 경우 후술할 임곗값 결정 방법으로 인해, 그림6의 세 번째 이미지처럼 전역 스레시홀딩에 비하여 선명한 결과물을 얻을 수 있다.

적응형 스레시홀딩에서의 임곗값 결정 방법은 이웃 픽셀의 평균값 또는 가우시안 분포에 따른 가중치의 합으로 결정된다. 본 연구에 적용된 적응형 스레시홀딩의 임곗값 결정 알고리즘은 다음과 같다. 전체 이미지에 총 9개의 영역을 설정한 뒤, 영역별로 임곗값을 지정한다. 이때 이웃 픽셀의 평균값을 매개변수로 전달하게 되면 나누어진 각 영역의 이웃 픽셀의 평균값으로 임곗값을 정한다. 그리고 가우시안 분포에 따른 가중치의 합으로 임곗값을 정한다. 대부분의 이미지는 그림자가 있거나 배경 사이에 조명 차이가 있다. 따라서 더 선명하고 부드러운 결과를 얻기 위해서는 적응형 스레시홀딩을 사용한다.

III. Data Preprocessing

이번 장에서는 'Quick, Draw!' 데이터셋을 수집 후 사용하여 CNN, RNN 기반 모델에서 사용될 데이터 전처리 과정을 소개한다.

1. Collect Datasets

본 연구는 신경망 학습을 위한 구글 AI Experiments의 'Quick, Draw!' 데이터셋을 수집하였다. 'Quick, Draw!'는 인공지능이 단어를 제시하면 사용자가 스케치하고 인공지능은 사용자의 그림을 유추하여 단어를 맞히는 게임이다. 구글은 345개의 카테고리에 걸쳐 약 5000만 개 이상의 낙서 이미지를 포함하는 'Quick, Draw!' 샘플 데이터셋을 공개[14]하였다.

이미지에는 다양한 표현 방식이 있는데, CNN에서는 각각의 이미지를 28x28 사이즈인 Grayscale 매트릭스로 포함[3]한 데이터셋을, RNN에서는 각 이미지를 일련의 선 벡터로 나타낸 데이터셋을 사용하였다.

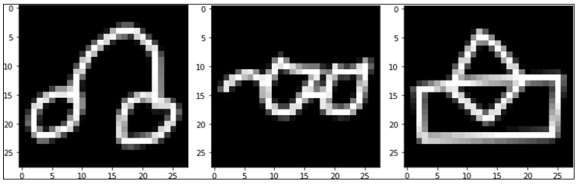

Fig. 7. Sample Data of a Headphone, Eyegalsses, Envelope (Left to Right) From The Training Dataset.

본 논문에서는 낙서의 분류를 위해 Numpy Bitmap 파일 형식의 데이터셋을 사용한다. 그림7은 'Quick, Draw!' 데이터셋의 샘플 중 일부로 본 연구에서는 표1과 같이클래스를 선정하였다.

Table 1. The Chosen Classes From The Datasets

Class		
Apple	Bowtie	Candle
Door	Envelope	Fish
Guitar	Ice Cream	Lightning
Moon	Mountain	Star

각 이미지 클래스별 10,000개의 데이터를 사용, 전체 120,000개의 데이터를 불러왔다. 모델별 학습 및 예측을 위해 파이선 언어 기반의 라이브러리, 패키지, 모듈들을 Anaconda 가상환경에서 사용하였다. 데이터셋을 읽어 특징과 라벨을 불러오기 위한 Numpy 패키지와 라벨을 One-Hot 인코딩 방법으로 전처리하고 모델에 레이어를 추가하는 Keras 라이브러리, 데이터를 혼합하거나 학습 데이터(Train Data)와 테스트 데이터(Test Data)를 분리하여 사용하기 위한 Scikit Learn 라이브러리를 사용하였다.

먼저, 추출한 데이터셋의 특징과 라벨을 혼합한 뒤 학습 데이터와 테스트 데이터를 분리한다. 혼합 과정은 데이터를 무작위 순서로 제시할 때, 알고리즘이 사이클에 갇히는 것을 방지[15]한다. 그리고 과적합을 방지하기 위해 학습 데이터와 테스트 데이터를 분리하였다. 모델을 활용하기 위해서는 학습하지 않은 데이터에 대한 예측이 되어야 하는데, 주어진 데이터만을 학습한 경우, 조금이라도 다른 패턴에 대해서는 모델의 성능이 떨어지게 된다. 그래서 테스트 데이터를 검증 데이터(Validation Data)로 사용하여 학습 중인 모델의 과적합을 확인하고 예측을 수행한다.

2. CNN (Convolution Neural Network)

CNN에서의 학습 결과를 얻기 위해 낙서 이미지를 28x28 크기로 재구성하고 0과 1사이의 픽셀 값을 정규화하여 전처리 된 데이터셋을 입력받았다. 두 개의 2차원 합성곱 레이어를 통해 이미지를 읽어 커널 사이즈가 2x2인 최대 풀링 레이어를 거친다. 전결합층에서는 1차원 데이터만 입력받을 수 있는데, 입력된 이미지는 가로, 세로, RGB 값으로 이루어진 3차원의 데이터이다. 그리고 평탄화 작업으로 이미지를 1차원 데이터로 변형하고 ReLu 활성화되어 있는 전결합층 두 개와 다중 클래스 분류를 위한 Softmax 함수, Drop-Out 레이어를 추가하였다. 모델 컴파일 과정에서는 가중치를 조정하여 손실 함수를 최소화하기 위한 확률적 경사 하강법 Adam Optimizer와 손실 함수로 교차 엔트로피 계열의 Category Crossentropy를 사용하였다. 아래 그림8은 CNN의 아키텍처 및 입력 데이터가 레이어를 거치는 과정을 도식화한 것이다.

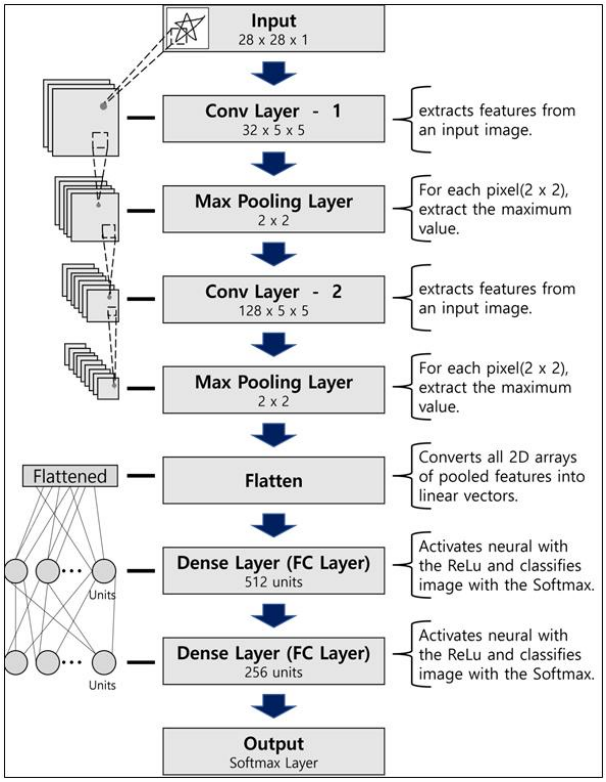

Fig. 8. CNN System Configuration Layout

3. RNN (Recurrent Neural Network)

RNN에서의 학습 결과를 얻기 위해 point가 x, y, n로 이루어진 Stroke의 시퀀스로 인코딩 전처리된 낙서를 입력받았다. 그림9와 같이 n은 해당 point가 새로운 Stroke의 첫 번째 point인지의 대한 여부를 나타낸다. 본 연구에서 사용된 모델은 입력 낙서 하나당 100개의 points를 저장한 후 사용되었다.

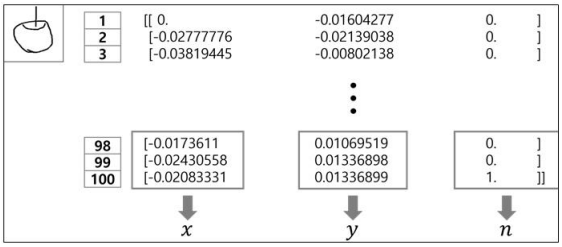

Fig. 9. Sequence of Strokes of Points

모델은 그림10과 같이 일련의 1차원 합성곱 레이어 세 개와 양방향 LSTM 레이어가 적용되었다. 양방향 LSTM 레이어는 합성곱 레이어의 출력을 전달한다. 또한, Softmax 활성화 기능으로 하는 전결합층 1개가 구성되고 Adam Optimizer를 통한 확률적 경사 하강법과 손실 함수로 Category Crossentory를 사용하였다.

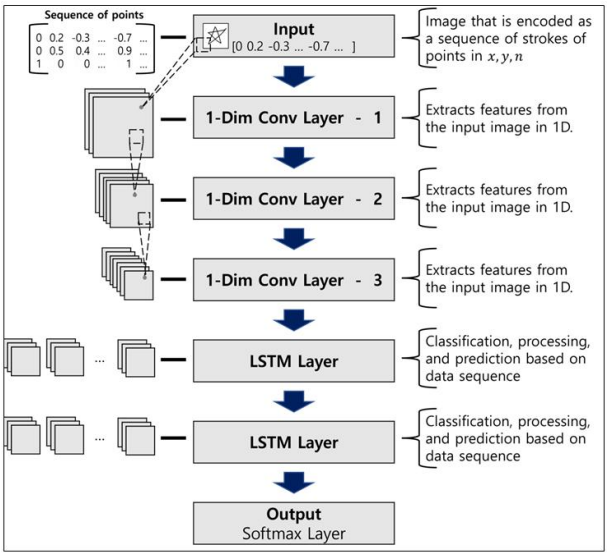

Fig. 10. RNN System Configuration Layout

IV. Experiment

4장에서는 데이터 전처리 과정을 통해 얻은 학습 데이터를 각 모델에 학습시킨다. 학습 Epoch는 총 100번으로, 10번마다 기록하였다. 또한, 각 모델의 배치 크기와 학습 및 테스트 데이터의 비율을 학습 데이터 90%, 테스트 데이터 10%로 나누었다. 배치 크기는 CNN에서 64, RNN에서 100으로 설정하였으며, 검증 데이터로서 테스트 데이터를 사용하여 모델 학습을 진행하였다. 그리고 캔버스 UI에 낙서를 스케치하여, 모델의 낙서 예측 성능을 비교한다. 낙서 예측 성능은 테스트 데이터를 검증 데이터로 사용하였을 때, 검증 정확도에 기반을 두어 평가하였다.

1. CNN Training

데이터 전처리 과정을 통해 얻은 낙서 이미지 데이터셋을 CNN에 학습시켰다. 그림11은 100번의 학습을 거친 CNN의 학습 결과이다. 80번 학습한 경우, 검증 데이터에 대한 검증 정확도는 약 96.06%로 나타났다. 한 번의 학습당 약 8초가 소요되었으므로 80번의 학습 시간은 약 640초, 즉 10분 40초가량 소요되었다.

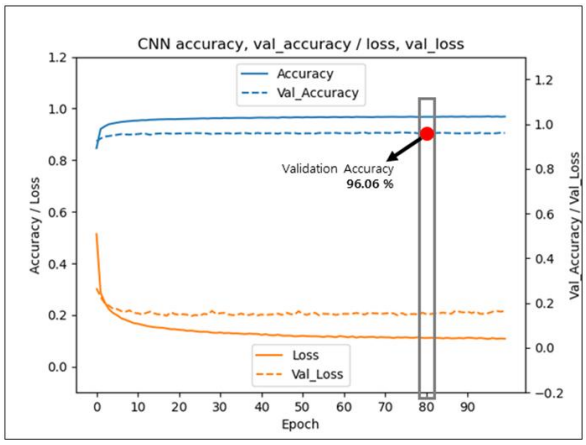

Fig. 11. Learning Results of CNN (Epoch 100)

2. RNN Training

낙서 이미지 데이터셋을 RNN에 학습시켰다. 그림12는 100번의 학습을 거친 RNN의 학습 결과이다. 40번 학습한 경우, 검증 데이터에 대한 검증 정확도는 약 95.76%로 나타났다. 실험 시간은 약 9,200초, 즉 2시간 33분 20초가량 소요되었다.

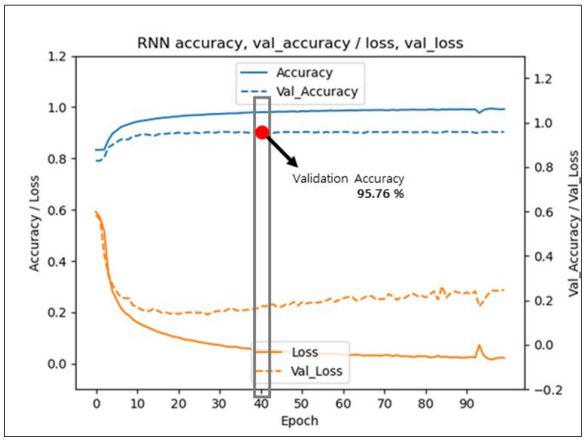

Fig. 12. Learning Results of RNN (Epoch 100)

3. Predict

예측을 위한 실험 낙서 클래스는 학습에 사용된 12가지 클래스 중 '사과(Apple)', '양초(Candle)', '물고기(Fish)', '기타(Guitar)', '아이스크림(Ice Cream)', '달(Moon)', 총 6가지 선정하였다. 다양한 상황에서의 결과를 얻기 위해 각자 다른 형태의 낙서를 세 번씩 스케치하여 진행하였다.

그림13은 각 모델의 예측 결과이다. 표2에 나타나듯, CNN은 18개의 스케치 중 17개를 예측하였으며, 이는 전체 스케치의 약 94.44%에 해당한다. 반면, RNN은 2개의 스케치만을 예측하였으며, 이는 전체 스케치의 약 11.11%에 해당한다. 모델별 예측한 모든 클래스에 대한 평균 예

Apple			Candle			Fish		
Input	Result		Input	Result		Input	Result	
	CNN	**Apple(95%)**, Candle(3%), Guitar(1%)		CNN	**Candle(100%)**		CNN	**Fish(61%)**, Lightning(34%), Moon(4%), Star(1%)
	RNN	**Apple(100%)**		RNN	Bowtie(100%)		RNN	Bowtie(100%)
	CNN	**Apple(98%)**, Moon(1%)		CNN	**Candle(100%)**		CNN	Envelope(1%), **Fish(79%)**, Lightning(6%), Moon(13%), Mountain(1%)
	RNN	**Apple(47%)**, Bowtie(52%), Star(1%)		RNN	Bowtie(100%)		RNN	Bowtie(82%), Star(18%)
	CNN	**Apple(100%)**		CNN	Bowtie(1%), **Candle(71%)**, Lightning(24%), Moon(1%), Start(1%)		CNN	Bowtie(1%), Envelope(1%), **Fish(1%)**, Lightning(93%), Moon(2%), Star(1%)
	RNN	**Apple(100%)**		RNN	Apple(1%), Bowtie(100%), Star(4%)		RNN	Bowtie(11%), Star(89%)
Guitar			**Ice Cream**			**Moon**		
Input	Result		Input	Result		Input	Result	
	CNN	**Guitar(100%)**		CNN	Apple(1%), Bowtie(16%), Candle(7%), Door(17%), Envelope(2%), Fish(1%), Guitar(1%), **Ice Cream(31%)**, Lightning(19%), Moon(2%), Mountain(1%), Star(1%)		CNN	**Moon(100%)**
	RNN	Bowtie(100%)		RNN	Bowtie(89%), Star(11%)		RNN	Bowtie(100%)
	CNN	Candle(1%), **Guitar(98%)**, Lightning(1%)		CNN	Bowtie(2%), Candle(2%), **Ice Cream(59%)**, Lightning(37%)		CNN	Apple(3%), Bowtie(7%), Candle(2%), Door(2%), Envelope(4%), Fish(8%), Guitar(3%), Ice Cream(3%), Lightning(7%), **Moon(56%)**, Mountain(3%), Star(2%)
	RNN	Bowtie(100%)		RNN	Bowtie(100%)		RNN	Bowtie(100%)
	CNN	Apple(4%), Bowtie(3%), Candle(7%), Door(1%), Envelope(2%), Fish(2%), **Guitar(46%)**, Ice Cream(3%), Lightning(15%), Moon(14%), Mountain(1%), Star(2%)		CNN	Apple(2%), Bowtie(26%), Candle(6%), Door(3%), Envelope(4%), Fish(4%), Guitar(3%), **Ice Cream(27%)**, Lightning(17%), Moon(3%), Mountain(3%), Star(2%)		CNN	Apple(4%), Bowtie(1%), **Moon(95%)**
	RNN	Bowtie(100%)		RNN	Apple(11%), Bowtie(88%), Star(1%)		RNN	Apple(80%), Bowtie(20%)

Fig. 13. Probabilities of Predicting Class-Specific Doodle Images in CNN and RNN Models.

측 정확도의 경우, CNN은 약 73.17%, RNN은 약 13.72%이다.

Table 2. Model-Specific Class Prediction Success Rate and Average Prediction Accuracy

	CNN	RNN
Prediction success rate	94.44% (17/18)	11.11% (2/18)
Average prediction accuracy	73.17%	13.72%

각 모델의 검증 데이터에 대한 검증 정확도 차이는 불과 약 0.3%이다. 그러나, 예측 성공 확률이 큰 차이를 보이는 이유는 캔버스 UI에 스케치한 낙서 이미지를 전처리한 과정이 RNN에 적합하지 않았던 것으로 판단된다. 기존 예측을 위한 이미지 전처리 과정은 스케치한 낙서의 데이터를 정규화한 뒤, 크기를 28x28 형태로 재구성하여 각 모델에 입력한 결과이다. 이는 낙서 이미지의 특징을 강조하고 크기를 일정하게 조정하였기에, 입력 데이터의 픽셀과 같은 공간적인 정보를 처리하는 것에 특화된 CNN에 적합하여 좋은 성능을 보일 수 있다. 그러나, 순차적인 시퀀스 데이터의 특성을 학습하기 위해 설계된 모델인 RNN은 낙서 이미지가 가지는 픽셀 정보, 그려진 형태 등 공간적인 정보를 처리하는 데 적합하지 않다. 기존 예측을 위한 이미지 전처리 과정은 RNN의 성질과 맞지 않았기에, 검증 데이터에 대한 검증 정확도 대비 예측 결과가 현저히 낮게 나타난 것이다. 이를 해결하기 위해 그림14와 같이 RNN에서의 예측에 적합한 낙서 이미지 전처리 과정을 추가로 구성하였다.

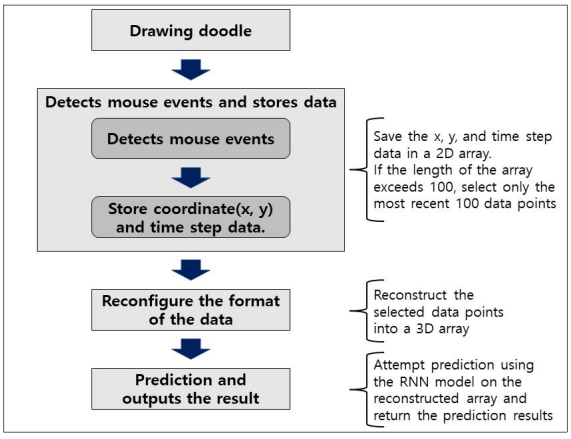

Fig. 14. The Preprocessing Steps for RNN Model Prediction of Doodle Images.

RNN에서의 적합한 데이터 전처리를 위해 마우스 이벤트가 발생할 때마다, 캔버스 UI 내 마우스의 좌표(x, y) 및 타임 스텝 데이터를 2차원 배열에 저장하였다. 2차원 배열의 길이가 100을 초과하기 시작하면, 가장 최근에 저장된 100개 데이터 points를 선택한다. points를 100개만 선택하는 이유는 그림9의 과정 중, 입력 낙서 하나당 100개의 points만을 저장하게끔 설계하였기 때문이다. 그리고 선택한 points를 3차원 배열로 재구성한다. RNN에 재구성한 배열을 입력하여 예측을 시도하고 예측된 결과를 얻는다. 추가로 구성된 전처리 과정을 바탕으로 CNN과 RNN의 클래스별 낙서 이미지 예측 확률을 재측정하였으며, 이는 그림15를 통해 확인할 수 있다. 그림13과 비교하면 RNN의 낙서 이미지 예측 횟수는 확연히 늘어났다. 표3에 나타나듯, CNN과 RNN 모두 18개의 낙서 이미지 중 15개를 예측하였으며, 이는 전체 스케치의 약 83.33%에 해당한다. 모델별 예측한 모든 클래스에 대한 평균 예측 정확도는, CNN은 약 62.28%, RNN은 약 62.83%로 RNN이 약 0.55% 근소하게 높은 것으로 측정된다.

Table 3. Model-Specific Class Prediction Success Rates and Average Prediction Accuracies After Improved Data Preprocessing

	CNN	RNN
Prediction success rate	83.33% (15/18)	
Average prediction accuracy	62.28%	62.83%

Apple			Candle			Fish		
Input	Result		Input	Result		Input	Result	
	CNN	**Apple(98%)**, Door(1%), Moon(1%)		CNN	**Candle(100%)**		CNN	Lightning(98%), Star(2%)
	RNN	**Apple(100%)**		RNN	**Candle(99%)**, Guitar(1%)		RNN	Bowtie(47%), **Fish(52%)**
	CNN	**Apple(78%)**, Bowtie(2%), Candle(8%), Door(1%), Guitar(2%), Ice Cream(1%), Lightning(3%), Moon(2%), Star(1%)		CNN	**Candle(100%)**		CNN	Apple(1%), Bowtie(5%), Candle(1%), Envelope(2%), **Fish(39%)**, Guitar(1%), Ice Cream(1%), Lightning(25%), Moon(20%), Mountain(2%), Star(3%)
	RNN	**Apple(98%)**, Guitar(1%)		RNN	Bowtie(4%), **Candle(37%)**, Door(1%), Envelope(1%), Guitar(6%), Ice Cream(15%), Lightning(25%), Moon(3%), Mountain(8%)		RNN	Guitar(100%)
	CNN	**Apple(87%)**, Bowtie(1%), Candle(4%), Door(1%), Guitar(1%), Lightning(1%), Moon(5%), Star(1%)		CNN	Apple(8%), **Candle(91%)**, Guitar(1%)		CNN	Apple(4%), Bowtie(11%), Candle(8%), Door(7%), Envelope(6%), **Fish(8%)**, Guitar(6%), Ice Cream(5%), Lightning(9%), Moon(27%), Mountain(5%), Star(4%)
	RNN	**Apple(89%)**, Guitar(10%), Lightning(1%)		RNN	Bowtie(1%), **Candle(34%)**, Guitar(13%), Ice Cream(39%), Lightning(13%)		RNN	Bowtie(1%), **Fish(69%)**, Guitar(23%), Moon(3%), Mountain(4%)
Guitar			**Ice Cream**			**Moon**		
Input	Result		Input	Result		Input	Result	
	CNN	**Guitar(100%)**		CNN	Apple(1%), Bowtie(11%), Candle(11%), Door(1%), Envelope(1%), Guitar(1%), Ice Cream(66%), Lightning(6%), Mountain(1%), Star(1%)		CNN	Apple(3%), Bowtie(4%), Candle(1%), Envelope(1%), Fish(1%), Ice Cream(1%), Lightning(25%), **Moon(54%)**, Mountain(2%), Star(6%)
	RNN	Candle(99%), Door(1%)		RNN	**Ice Cream(100%)**		RNN	Fish(1%), **Moon(99%)**
	CNN	Apple(8%), Bowtie(1%), Candle(12%), **Guitar(50%)**, Lightning(23%), Moon(3%), Mountain(1%), Star(2%)		CNN	Apple(1%), Bowtie(8%), Candle(24%), Door(3%), Envelope(2%), Fish(1%), Guitar(3%), **Ice Cream(36%)**, Lightning(17%), Moon(2%), Mountain(2%), Star(1%)		CNN	**Moon(100%)**
	RNN	Apple(1%), Bowtie(5%), Candle(13%), Door(9%), Envelope(1%), Fish(4%), **Guitar(22%)**, Ice Cream(17%), Lightning(21%), Moon(1%), Mountain(6%)		RNN	Bowtie(1%), Candle(4%), Guitar(44%), **Ice Cream(50%)**		RNN	Apple(7%), Fish(7%), **Moon(81%)**, Mountain(3%)
	CNN	Apple(1%), Bowtie(2%), Candle(4%), Envelope(1%), Fish(2%), **Guitar(47%)**, Ice Cream(1%), Lightning(6%), Moon(35%)		CNN	Bowtie(3%), Candle(13%), Door(3%), Lightning(76%), Moon(3%)		CNN	Apple(6%), Bowtie(5%), Candle(2%), Door(2%), Envelope(2%), Fish(3%), Guitar(2%), Ice Cream(2%), Lightning(6%), **Moon(67%)**, Mountain(2%), Star(1%)
	RNN	Apple(3%), Bowtie(24%), Candle(1%), Envelope(1%), Fish(12%), **Guitar(45%)**, Ice Cream(1%), Lightning(10%), Mountain(2%)		RNN	Apple(3%), Envelope(2%), **Ice Cream(92%)**, Moon(1%)		RNN	Apple(35%), Fish(1%), **Moon(64%)**

Fig. 15. Probabilistic Prediction of Class-Specific Doodle Images Using CNN and RNN Models with Enhanced Data Preprocessing.

V. Conclusion

본 논문은 인공지능이 낙서 스케치를 얼마나 정확하게 예측할 수 있는지 분석하기 위해, CNN, RNN 기반 모델의 성능을 비교하였다. 100번의 Epoch 조건에서 10번마다 기록했을 때, CNN에서의 검증 데이터에 대한 검증 정확도는 RNN보다 약 0.3% 높았다. 낙서 스케치 예측 결과의 경우, 최종 예측 실험에서 18개의 스케치 중 15개를 예측하는 것에 성공하였고, 예측 정확도는 RNN이 CNN보다 약 0.55% 높았다. 위 실험 결과를 종합하였을 때, 본 연구에 사용된 CNN과 RNN의 성능은 유사하지만, CNN은 입력된 낙서 스케치의 픽셀을 통해 예측하는 것에 강점을 갖기에, 스케치가 끝난 낙서 이미지를 분류하는 것에 적합하다. 반면에 RNN은 낙서 이미지를 순차적인 시퀀셜 데이터 형식으로 처리하여 예측하는 것에 강점이 있어, 실시간으로 낙서 이미지를 예측하는 것에 적합하다.

인공지능은 불규칙한 낙서를 기계적으로 패턴을 인식하고 예측하는 것이 인간에 비해 어렵지만, 학습 데이터를 통해 패턴을 학습함으로써 이를 가능하게 할 수 있다. 그렇기에 본 논문에서는 낙서의 다양한 특성이 있는 'Quick, Draw!' 데이터셋을 활용하여, 낙서 이미지 예측에 신경망을 적용하는 새로운 접근 방식을 제시하였다. 이로써 다양한 환경에서 효과적인 낙서 이미지 예측을 실현하는 인공지능을 얻었다. 또한, 데이터 전처리 과정에 대한 개선을 진행하여 낙서 클래스 예측의 성능을 높였다. 이렇게 인간의 표현과 의도를 이해하는 인공지능이 낙서 스케치를 예측함으로써 인간에게 독특한 관점을 제공할 수 있다. 이는 창작과 예술 분야의 다양성 확장을 유도하고 예술적 표현의 폭을 넓힐 수 있을 것으로 기대한다.

REFERENCES

[1] J. W. Kim, P. K. Rhee, "Image Recognition based on Adaptive Deep Learning", The Journal of The Institute of Internet, Broadcasting and Communication (IIBC), Vol. 18, No. 1, pp. 113-117, Feb, 2018. DOI: 10.7236/JIIBC.2018.18.1.113

[2] CVML Team, "An On-device Deep Neural Network for Face Detection", Apple Machine Learning Journal, Nov, 2017. https://machinelearning.apple.com/research/face-detection

[3] G. Kristine, J. WoMa, E. Xu, "Quick, Draw! Doodle Recognition", Stanford University, 2018.

[4] N. K. Lee, J. Y. Kim, J. H. Shim, "Empirical Study on Analyzing Training Data for CNN-based Product Classification Deep Learning Model", The Journal of Society for e-Business Studies, Vol. 26, No. 1, pp. 107-126, Feb, 2021. DOI: 10.7838/jsebs.2021.26.1.107

[5] X. Sun, L. Liu, C. Li, J. Yin, J. Zhao, W. Si, "Classification for remote sensing data with improved CNN-SVM method", IEEE Access, Vol. 7, pp. 164507-164516, Nov, 2019. DOI: 10.1109/ACCESS.2019.2952946

[6] G. Wang, S. Y. Shin, W. J. Lee, "A Text Sentiment Classification Method Based on LSTM-CNN", Journal of The Korea Society of Computer and Information, Vol. 24, No. 12, pp. 1-7, Dec, 2019. DOI: 10.9708/JKSCI.2019.24.12.001

[7] D. Ha, D. Eck, "A Neural Representation of Sketch Drawings", arXiv preprint arXiv:1704.03477, May, 2017. DOI: 10.48550/arXiv.1704.03477

[8] M. Schuster, K. K. Paliwal, "Bidirectional recurrent neural networks", IEEE Transactions on Signal Processing, Vol. 45, No. 11, pp. 2673-2681, Nov, 1997. DOI: 10.1109/78.650093

[9] L. Zhou, "Paper Dreams: an adaptive drawing canvas supported by machine learning", Diss. Massachusetts Institute of Technology, Jun, 2019. DOI: 1721.1/122990

[10] R. Firdousi, S. Parveen, "Local Thresholding Techniques in Image Binarization", International Journal Of Engineering And Computer Science, Vol. 3, No. 3, pp. 4062-4065, Mar, 2014. DOI: 10.18535/ijecs

[11] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, pp. 62-66, Jan, 1979. DOI: 10.1109/TSMC.1979.4310076

[12] K. Dawson-Howe, "A practical introduction to computer vision with opencv", John Wiley & Sons, pp. 54-56, Apr, 2014.

[13] D. Bradley, G. Roth, "Adaptive thresholding using the integral image", Journal of graphics tools, Vol. 12, No. 2, pp. 13-21, Jun, 2007. DOI: 10.1080/2151237X.2007.10129236

[14] R. F. Fernandez, J. G. Victores, D. Estevez, C. Balaguer, "Quick, Stat!: A Statistical Analysis of the Quick, Draw! Dataset", arXiv preprint arXiv:1907.06417, Oct, 2019. DOI: 10.48550/arXiv.1907.06417

[15] S. Raschka, "Python machine learning", Packt publishing ltd, pp. 43-44, Sep, 2015.

Authors

Hae-Chan Lee received the A.S., B.S. degrees in Computer Information Engineering from Inha Technical College, Korea, in 2022, 2023, respectively. Mr. Lee entered the Inha Technical College in 2017 and graduated in 2023. He has experience in IT practice and interested in IT and the fourth industrial revolution.

Kyu-Cheol Cho received the B.S., M.S. and Ph.D. degrees in Computer Science and Information Engineering from Inha University, Korea, in 2005, 2007 and 2013, respectively. Dr. Cho joined the faculty of the Department of Computer Science at Inha Technical College, Incheon, Korea, in 2016. He is currently a assistant professor in the Department of Computer Science, Inha Technical College. He is interested in cloud computing, green IT and web programming.