인공신경망을 사용한 한글 OCR 모델의 구현

이진성

임베디드시스템공학과 / 202201673

Implementation of an Korean OCR Model Using ANN

요 약

본 연구는 인공신경망(Artificial Neural Network, ANN)을 활용하여 한국어 문자인식(Optical Character Recognition, OCR) 모델을 설계하고, 레이어 구성, 가중치 초기화 방식, 활성화 함수, 학습 데이터셋 비율 등 다양한 요소가 모델 성능에 미치는 영향을 분석하였다. 실험은 Python을 사용하여 생성된 3388개의 1비트 비트맵 이미지를 활용하였으며, 각 이미지에서 초성, 중성, 종성 인덱스를 출력으로 예측하는 ANN 구조를 개발하였다. 레이어 수와 노드 수가 성능에 중요한 영향을 미쳤으며, 레이어 12개, 노드 256개 구성에서 MSE가 0.00007로 최적의 성능을 나타냈다. 활성화 함수로 ReLU를 사용했을 때 Sigmoid와 Tanh에 비해 성능이 우수했으며, 가중치 초기화 방식(He와 Xavier)의 차이는 크지 않았다. 학습 중 나오는 NaN 값을 방지하기 위해 클리핑을 적용하였으며, 학습 초기 높은 학습률을 점진적으로 낮추는 방식으로 안정성을 확보했다. 실험 결과 MSE는 신뢰할 만한 수준으로 낮았으나, 정확도 계산 로직의 한계로 인해 정확도 결과는 미흡했다. 향후 정확도 개선과 학습 속도 최적화를 위한 추가 연구가 필요하며, 본 연구는 ANN 기반 OCR 시스템 설계와 구현에 중요한 통찰을 제공하였다.

Abstract

This study presents the design and performance analysis of a Korean Optical Character Recognition (OCR) model using Artificial Neural Networks (ANN). Various factors, including layer structure, weight initialization methods, activation functions, and dataset ratios, were evaluated for their impact on model performance. A total of 3388 1-bit bitmap images were generated using Python, and an ANN was implemented to predict initial, medial, and final consonant indices from these images. The number of layers and nodes significantly affected performance, with the optimal structure (12 layers and 256 nodes per layer) achieving a Mean Squared Error (MSE) of 0.00007. ReLU activation function outperformed Sigmoid and Tanh, while the difference between He and Xavier weight initialization methods was negligible. Clipping was introduced to prevent NaN values during training, and a gradually decreasing learning rate ensured stable learning. While MSE results indicated a highly reliable model, accuracy measurements revealed discrepancies due to limitations in the calculation logic. Further research is required to enhance accuracy and optimize training speed. This study provides valuable insights into ANN-based OCR system design and implementation.

Key words

OCR, ANN, Korean Unicode Index, Expansion, Downsampling, Bitmap Image Classification

Ⅰ. 서 론

문자인식(Optical Character Recognition, OCR)은 이미지 또는 기타 비정형 데이터로부터 텍스트를 추출하는 기술로, 오래전부터 다양한 응용 분야에서 핵심적인 역할을 수행해 왔다. 초창기 OCR 기술[1]은 주로 규칙 기반 알고리즘과 템플릿 매칭 방식을 활용했으며, 이는 문자 구조를 명시적으로 정의하고 일치 여부를 검사하는 방식으로 작동했다. 이러한 초기 방식은 특정 환경에서 높은 성능을 보이지만, 복잡한 배경, 다양한 글꼴 스타일, 또는 비정형적인 문서에서 성능이 급격히 저하되는 한계를 보인다.

이러한 한계를 극복하기 위해서 1980년대 후반부터 인공신경망 (Artiicial Neural Network, ANN)이 OCR 기술에 도입되기 시작했다. 인공신경망은 데이터를 통해 패턴을 학습하고 복잡한 비선형 문제를 해결하는 데 강점을 보인다. 초기의 ANN 기반 OCR 모델은 단층 퍼셉트론(Single Layer Perceptron) 또는 다층 퍼셉트론(Multi-Layer Perceptron, MLP)을 사용했으며, 기본적인 문자 분류 작업에서 점진적으로 기존 방식보다 우수한 성능을 보였다[2].

1990년대 중반, 심층신경망(Deep Neural Network, DNN)이 본격적으로 연구되면서 OCR 기술은 또 다른 도약을 이루었다[3]. 특히, 시간적 종속성을 고려한 순환 신경망(Recurrent Neural Network, RNN)과 이미지의 공간적 특성을 효과적으로 학습할 수 있는 합성곱 신경망(Convolutional Neural Network, CNN)의 발전은 OCR 성능을 획기적으로 향상시켰다. 이러한 신경망은 글자의 세부적 특징과 문맥적 정보를 동시에 학습할 수 있어 다양한 문자 세트와 복잡한 이미지 배경에서도 높은 정확도를 기록했다[4][5].

본 실험에서는 기존의 복잡한 프레임워크와 외부 라이브러리를 사용하지 않고, 인공신경망과 단순한 전처리를 사용하여 특정 한국어 문자인식을 수행하는 모델을 설계하고 실험한다. 레이어와 노드의 수, 가중치 초기화 방식과 활성화 함수에 따른 최적 모델을 도출하는 것을 목적으로 한다. 이를 통해 인공신경망의 동작 원리를 명확히 이해하고 최적 하이퍼 파라미터를 도출하는 실험을 통해 인공지능 교과목을 수강함에 있어 전공역량을 증대시키는 것을 목표하고자 한다.

Ⅱ. 선행 과제

선행 과제에서는 C언어를 사용하여 인공신경망을 구현하고 순전파 연산을 하는 것을 구현하였다. 실행 중 텍스트 입력을 통해 레이어의 수와 각 레이어 당 노드 수를 입력받으며, 이를 동적으로 할당하여 입력되는 값에 대응되도록 코드를 작성하였다.

활성화 함수는 Sigmoid로 고정되어 있었으며, 가중치의 초기화는 별도 조건이 없었다. 이에 랜덤으로 가중치를 만드는 코드와 레이어와 노드 수를 입력 받은 후 가중치도 입력받도록 하는 코드를 작성하였다.

Ⅲ. 실험 방법

3.1. 실험 데이터셋 텍스트 선별

본 실험에서 사용하는 데이터셋에 들어가는 텍스트의 종류는 다음과 같다.

- 주민등록상의 현주소에 포함된 모든 글자
- 인천대학교 도로명 주소 글자
- 제출자의 이름
- 부모와 조부모 6명의 성함
- 민태원의 정춘예찬에 나오는 첫 두 문단의 글자

주민등록상의 현주소는 기숙사이나 인천대학교 도로명 주소와 일치하기 때문에 본가를 기준으로 작성하였다. 이들 중 한글에 해당하는 텍스트는 아래와 같다.

- ~~개인정보삭제~~

- 인천광역시연수구송도동아카데미로인천대학교
- 이진성
- 개인정보삭제
- 청춘이는듣기만하여도가슴이설레는말이다 청춘너의두손을가슴에대고물방아같은심장의고동을들어보라청춘의피는끓는다끓는피에뛰노는심장은거선의기관같이힘있다이것이다인류의역사를꾸며내려온동력은바로이것이다이성은투명하되얼음과같으며지혜는날카로우나갑속에든칼이다청춘의끓는피가아니더면인간이얼마나쓸쓸하랴얼음에싸인만물은죽은이있을뿐이다

이들의 합집합은 아래와 같다.
- 개인정보삭제

본 실험에서는 위의 121자의 텍스트를 사용한다.

3.2. 사용 폰트의 수집

사용하는 폰트는 한글 프로그램에서 제공하는 폰트를 사용하였으며, 데이터셋 이미지를 만들기 위해 사용한 폰트는 아래와 같다.

- batang.tcc
- gulim.tcc
- H2GPRM.TTF
- H2GSRB.TTF
- H2GTRE.TTF
- H2GTRM.TTF
- H2HDRM.TTF
- H2MJRE.TTF
- H2MJSM.TTF
- H2MKPB.TTF
- H2PORL.TTF
- H2PORM.TTF
- H2SA1M.TTF
- HANBatangB.ttf
- HANBatangExtB.ttf
- HANBatangExtBB.ttf
- Hancom Gothic Bold.ttf
- Hancom Gothic Regular.ttf
- HANDotum.ttf
- HANDotumB.ttf
- HanSantteutDotum-Bold.ttf
- HanSantteutDotum-Regular.ttf
- HMFMMUEX.TTC
- HMFMOLD.TTF
- HMFMPYUN.TTF
- HMKMAMI.TTF
- HMKMMAG.TTF
- HMKMRHD.TTF
- NGULIM.TTF

위 30개의 폰트 중 후에 기술할 데이터셋 이미지를 만드는 과정에서 정상적인 이미지를 생성하지 못한 두 개의 폰트는 실험에서 제외되었다.

3.3. 실험 데이터셋 이미지 생성

실험 데이터셋 이미지를 만들기 위해 3.1.과 3.2.의 텍스트와 폰트를 사용하였다. 이미지 생성은 Python을 사용하여 만들었으며 Python Imaging Library인 Pillow를 사용하였다.
먼저 중복 텍스트가 존재하는 문자열을 변수에 할당하고 set을 사용하여 중복되는 텍스트를 제거하였다.
각 텍스트와 각 폰트에 대해서 64x64 사이즈의 1비트 비트맵 이미지를 생성하였으며, 배경은 0, 글자가 있는 부분의 값은 1을 가지도록 하여 검은색 배경의 흰 글씨를 갖는 이미지를 갖도록 하였다.

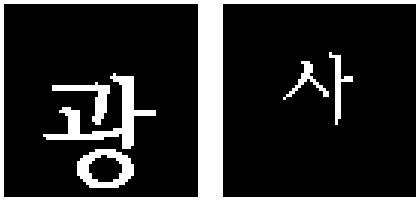

그림 1. 비트맵 이미지

각 이미지에 대한 정보를 저장하기 위해

text{05d}.bmp

형태로 이미지를 저장하도록 한다. 동시에 텍스트 파일에

text{05d}_{02d}_{02d}_{02d}

형태의 텍스트를 저장하는데, 순서대로 현재 생성하는 이미지의 순서 인덱스, 초성 인덱스(19개, 0~18), 중성 인덱스(21개, 0~20), 종성 인덱스(28개, 0~27)이다. 187번째로 생성되는 이미지의 글자가 '최'라면 'text00187.bmp'와 같은 형식으로 이미지를 저장하고, 'text00187_14_11_00'와 같은 형식으로 텍스트파일에 텍스트를 저장한다. 이때 'ㅊ'의 인덱스는 14, 'ㅚ'의 인덱스는 11, 종성은 없으므로 종성의 인덱스는 0이다. 인덱스는 유니코드에서 한글을 정의하는데 사용하는 순서를 사용한다.

Python 코드 상에서 텍스트를 정상적으로 작성하지 못하는 경우는 저장하는 이미지에서 제외하였다. 이때 2개 폰트에 대해 정상적인 이미지 작성이 불가하였으며, 최종적으로 28개 폰트에 대한 121자 텍스트의 이미지인 총 3388개의 이미지를 생성할 수 있었다. 텍스트의 초성, 중성, 종성에 대한 인덱스를 저장하는 하나의 텍스트 파일은 별도로 생성하였다.

3.4. 이미지 전처리1 - 확장

3.3.을 통해 만들어진 텍스트 이미지의 경우 텍스트가 위치한 영역이 일정하지 않으며, 크기 또한 일정하지 않다. 코드 상에서 가운데 정렬을 사용했지만 폰트에 따라 중앙의 정의가 달라 모든 이미지 파일들의 텍스트가 중앙에 위치하지 않는다. 또한 평가 지표로 사용할 텍스트 이미지의 크기와 이미지를 알지 못한다.

때문에 ANN의 입력층에 이미지를 넣기 전에 텍스트만 남도록 확장하는 전처리가 필요하다. 이를 위해 cropIndex 함수를 정의하여 이미지에서 텍스트가 위치한 영역을 판별하여 bounding box를 그릴 수 있는 좌표를 저장하여 반환시키고 이를 imageStretch 함수가 이미지 배열과 함께 입력으로 받아 확장, 이미지에 텍스트가 꽉 차도록 새로운 배열을 생성한다.

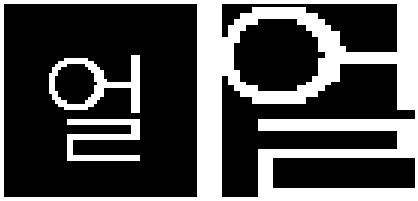

그림 2. 원본 이미지와 확장 이미지

이미지를 확장하기 위해서 확장된 배열의 인덱스를 원본 배열의 크롭되는 인덱스에 매칭하고 반올림하여 가장 인근에 위치한 인덱스의 값을 통해 확장된 배열에서의 값을 정한다.

3.5. 이미지 전처리2 - 다운샘플링

3.4.를 통해 이미지에서 텍스트가 전체 영역에 위치하도록 한 후, 이미지의 픽셀 수를 줄이기 위한 작업을 한다. 입력 이미지는 64x64 사이즈로 총 4096개의 픽셀 값을 가진다. 문제 상황에서 입력층과 출력층의 노드의 수는 자율적으로 처리하라는 지시가 있음과 별개로 다른 레이어와 동일하게 256개 이하의 노드를 사용하도록 처리하고자 하였다.

이를 위해 16x16의 배열을 만들고 64x64 사이즈의 이미지를 4x4 만큼 묶어 16개 픽

셀의 평균값을 하나의 픽셀에 할당한다.

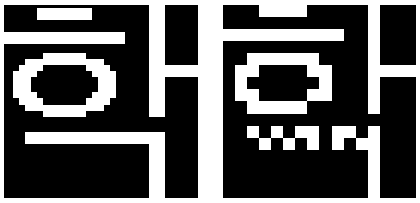

그림 3. 64x64 사이즈의 비트맵 확장 이미지와
16x16 사이즈의 gray scale 다운샘플링 이미지

이를 통해 64x64 이미지를 16x16 이미지로 다운샘플링하면서 ANN의 입력층에 넣는 값을 기존 4096개에서 256개로 축소시킨다.

3.6. 출력 레이어와 Target 구성

출력 레이어는 3개의 노드로 구성되며 각 노드의 값은 각각 초성, 중성, 종성의 값을 가진다. 이때 출력 레이어의 활성화 함수는 Sigmoid인데 0에서 1 사이의 값을 반환한다.

유니코드는 초성 19개, 중성 21개, 종성 28개로 구성되는데, 이는 3.3.에서 기술한 바와 같이 0부터 (각 글자 수 - 1)까지의 정수 값을 가지게 된다. 이 값이 1을 더하고 최대값보다 1만큼 더 큰 값이 1이 되게끔 값을 축소하여 Target 값을 정의하게 된다. 초성이 'ㅊ'인 경우 인덱스가 14이므로 $\frac{1+14}{1+19+1}$을 Target으로 정하게 된다.

반대로 순전파 후 출력층의 값이 나온 경우에는 역으로 연산한 후 반올림하여 인덱스를 추정한다. 출력층의 세 번째 노드의 출력값에 30(1+28+1)을 곱한 후 1을 뺀 값을 반올림 하였을 때 값이 16이라면 종성이 'ㅁ'인 것으로 추정한다.

3.7. Batch 구성과 스레드를 통한 가속

본 실험에서는 총 3388개의 데이터셋이 사용되는데, 121개의 텍스트에 대해 28개의 폰트로 만든 이미지가 사용된다. 흔히 2의 배수로 batch를 구성하는 것에 반해 한 텍스트 조합인 121개를 batch의 사이즈로 정하였다. 총 28개의 batch를 사용한다.

기본적으로 28개의 batch를 21:7로 나누어 1 epoch에 21개의 batch를 training에, 7개의 batch를 testing에 사용하였다. 실험의 진행에 따라 24:4와 같은 다른 비율로 나눠서 실험을 추가적으로 진행하였다.

Batch에 대해서 batch의 사이즈 만큼의 스레드를 만들어 순전파와 역전파를 진행하도록 설계하였는데, 맥과 리눅스, 리눅스 가상환경에서 구동하도록 pthread를 사용하여 구현하였다. CUDA 가속의 경우 동작 시간과 개발시간, 코드 구동 환경을 고려하여 배제하였다. Training의 순전파와 역전파, Testing의 순전파를 모두 스레드를 통해 동시에 연산을 수행하도록 하였다.

3.8. Clipping

코드 작성 완료 후 초기 실험 단계에서 20 epoch 내에서 노드의 입력값과 출력값, 가중치, MSE에서 nan(Not a Number)이 반환되는 상황이 계속적으로 발생하였다. 이를 해결하기 위해 클리핑을 사용하였으며, 총 두 부분에 대해서 클리핑을 하였다.

먼저 활성화 함수에 대한 클리핑을 진행하였다. 별도의 클리핑 함수를 정의하여 활성화 함수에 값이 들어가기 이전에 범위를 제한한 후에 활성화 함수를 거칠 수 있도록 하였다.

또한 가중치에 대한 클리핑을 하였는데, 가중치를 업데이트한 후 범위 외의 가중치를 범위의 경계가 되게끔 수정하였다.

3.9. 학습률

인공신경망의 학습비율은 낮을 경우 학습의 속도가 저하되며, 높을 경우 값이 발산하는 등 불안정적인 학습이 진행될 가능성이 높다. 이에 적절한 학습률을 지정할 필요가 있다.

이때 학습 초기에는 높은 학습률을, 학습이 진행됨에 따라 학습률을 낮춰 모델의 학습속도를 높이면서 안정적인 학습을 진행하도록

할 수 있다.
이에 오차로 사용하는 MSE(Mean Squared Error)의 값과 비례하게 학습률을 낮추도록 하였다. $MSE \times 10$과 $MSE \times 100$를 학습과정에서 사용하였으며, 최종적으로 $MSE \times 100$를 학습에 사용하였다.
이때 학습 초기의 경우 0.1대의 MSE를 보이는데, 이때 MSE가 0.001 이하로 떨어지기 전까지는 학습률을 $MSE \times 100$가 아닌 0.1을 사용하도록 하였다. 이를 통해 과도한 학습에 따른 문제를 방지하고자 하였다.

3.10. 학습 중단

본 실험에서는 오차를 위해 MSE를 사용하는데, 통상적으로 MSE의 미분값이 0에 수렴함에 따라 학습을 중단시키는 것이 일반적이다. 하지만 본 실험에 있어서 학습 로그를 관찰한 결과 진동하는 경향이 강하고 학습이 진행됨에 따라 변화량이 미미하여 미분값에 따른 학습 중단은 어려웠다.
또한 대부분의 상황에서 epoch이 증가함에 따라 값이 감소하는 추세를 보였으며, epoch이 늘어나는 것과 별개로 MSE가 지속적으로 감소하는 것을 볼 수 있었다. 때문에 MSE에 따라 학습을 중단하는 것이 아닌 특정 epoch까지 학습한 후 학습을 중단하도록 하였다.
본 실험에서는 전체적으로 5000 epoch을 학습한 후 학습을 중단하도록 하였다. 성능평가 또한 5000 epoch 학습에 대한 성능평가로 진행된다.

3.11. 학습 및 테스트 비율

흔히 1 epoch에서 학습과 테스트 데이터 셋의 비율은 7:3 혹은 8:2, 9:1과 같이 나눈다. 본 실험에서는 통상적으로 나누는 비율이 아닌 데이터셋의 수에 따라 나누고자 하였다.
121개로 구성하는 하나의 batch에 따라 28개의 batch가 구성되며, 이를 이용하여 21:7, 24:4과 같은 비율로 학습과 테스트 데이이터 셋을 나누었다.

3.12. 가중치 초기화

가중치를 초기화 하는 방법은 다양하지만 본 실험에서는 두 가지의 초기화를 사용하였다. 기본적으로 He 초기화를 사용하였으며, Xavier 초기화를 사용하였다.

3.13. 활성화 함수

본 실험에서는 ReLU와 Sigmoid, Tanh를 활성화 함수로 사용하였는데, 출력층의 노드는 개별적으로 0 초과 1 미만의 범위의 수를 반환하여야 함으로 Sigmoid를 사용하였다.
기본적으로 은닉층에 대해서는 ReLU를 사용하였으며, Sigmoid와 Tanh에 대한 비교 실험을 진행하였다.

3.14. 레이어 수와 레이어 별 노드 수

본 실험에서 레이어 수와 레이어 별 노드의 수는 균일한 경우와 감소하는 경우로 나누어 실험을 진행하였다.
5개 이상 16개 이하의 레이어, 최대 256개의 노드를 갖는 모든 ANN을 구성하고 실험하는 것은 많은 자원을 필요로 함으로 본 실험에서는 특정 수의 레이어와 노드에 대한 실험만을 진행하였다.
레이어의 경우 6, 8, 10, 12, 16개에 대한 실험을 진행하였으며, 노드는 모든 레이어에 대해 4, 8, 16, 32, 64, 128, 256개에 대한 실험을 진행하였다.
전체 노드 수가 균일한 경우 외에도 노드의 수가 감소하는 형태의 ANN에 대한 실험을 진행하였는데, 8, 10개의 레이어를 갖는 신경망에 대해 은닉층의 노드 수가 순서대로 128, 64, 32, 16, 8, 4개인 ANN, 그리고 256, 128, 64, 32, 16, 8, 4, 3개인 ANN에 대한 학습을 실험하였다.

3.15. 평가 지표

실험을 통해 생성되는 모델의 성능에 대한 평가는 두 가지의 지표로 이루어진다. 오차와 정확도이다.

먼저 오차는 테스트 데이터 셋에 대한 MSE를 사용한다. 출력 층이 3개이기 때문에 MSE는 다음과 같이 나타낸다.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{3}\sum_{j=1}^{3}(y_{ij}-\hat{y}_{ij})^2$$

정확도는 총 세 가지로 측정한다. 3.6.에 따라 각 문자의 인덱스를 유리수로 변환하는 과정을 거치게 되는데, 이를 반대로 유리수를 정수로 변환하면 문자의 인덱스를 알 수 있게 된다. 여기에서 사전에 불러온 target 값과 비교하여 정확도를 측정한다.

이때 정확하게 일치하는 경우와 1개 인덱스 내의 오차, 2개 인덱스 내의 오차로 총 3개의 정확도를 측정한다. 정확하게 일치하는 경우를 제외한 두 경우에 대해 정확하게 일치하지는 않지만 근사치를 찾은 것으로 간주한다.

다만 인덱스를 기준으로 하는 정확도의 경우 로직상으로는 문제가 없어보이나 수치가 저조한 탓에 MSE를 기준으로 성능을 평가하고자 한다.

3.15. 실험 환경

소스코드는 C언어로 작성되었으며, 세 종류의 PC를 통해 학습을 진행하였다. 학습 진행에 사용된 PC의 시스템은 다음과 같다.

표 1. 실험 환경 1

Linux Server Desktop	
CPU	i9-10900K
RAM	32GB
OS	Ubuntu 20.04.06 LTS

표 2. 실험 환경 2

Apple Macbook Pro	
CPU	M2 Pro
RAM	32GB
OS	MacOS 15.1.1 (Sequoia)

표 3. 실험 환경 3

ASUS Vivobook Laptop	
CPU	i5-11300H
RAM	16GB
OS	WSL2.0 in Windows 11

GCC 컴파일러를 이용하여 컴파일 하였으며, 실험 환경 1에 대해서는 동시간대 최대 5건의 학습을 진행하였다.

한 건의 학습에 대해 실험 환경 1에서는 10~15%의 CPU 점유율을, 실험 환경 2에서는 20~25%, 실험 환경 3에서는 60~70%의 CPU 점유율을 보였다.

Ⅳ. 실험 결과 및 분석

4.1. 레이어와 노드 수에 따른 MSE

표 4는 레이어 수에 따라 균일한 노드 수를 가지는 ANN에 대한 5000 epoch에 대한 MSE이다. 편의에 따라 MSE가 상대적으로 높은 경우에 대해 반올림하였으며, 0.0001대 이하의 낮은 경우 더 정밀한 자릿수를 작성하였다. 또한 가중치의 초기화는 He 초기화를, 은닉층의 활성화 함수는 모두 ReLU를 사용하였다. 학습과 테스트 데이터셋의 비율은 21:7이다.

표 4. 레이어와 노드 수에 따른 MSE

Node	Layer				
	6	8	10	12	16
4	0.03	0.05	0.05	0.09	0.1
8	0.009	0.07	0.07	0.05	0.1
16	0.002	0.002	0.003	0.05	0.05
32	0.0005	0.0004	0.0004	0.0006	
64	0.0002	0.002	0.0002	0.0002	
128	0.000137	0.00012	0.000101		
256	0.000088	0.000080	0.000073	0.000070	0.00018

MSE가 기입되지 않은 경우는 학습을 진행하지 않은 경우이다. 기본적으로 노드의 수가 많을수록 MSE가 낮은 모습을 볼 수 있으며, 레이어가 많을수록 MSE가 높은 모습을 보인다. 이에 레이어 당 노드의 수가 균

일한 경우에 대해 노드의 수가 적은 경우는 학습에 적합하지 않으며, 레이어의 수가 많은 경우는 학습이 정상적으로 이루어지지 않음을 알 수 있다.

표 5. 감소하는 노드를 갖는 ANN의 MSE

Layer	Hidden Layer's Node	MSE
8	128/64/32/16/8/4	0.000168
10	256/128/64/32/16/8/4/3	0.0665

레이어 당 노드의 수가 일정한 경우 뿐만 아니라 변하는 경우에 대한 학습 또한 진행하였는데, 입력 레이어의 노드가 256개, 출력 레이어의 노드가 3개로 고정됨에 따라 절반씩 감소하도록 구성하였다. 레이어의 수가 8개인 경우보다 레이어의 수가 10개인 경우 MSE가 큰 것을 볼 수 있다. 레이어 당 노드의 수가 동일한 경우와 같이 레이어의 수가 많을수록 학습이 정상적으로 되지 않음을 알 수 있다.

학습을 진행한 경우들에 대해 12개의 레이어에 대해 은닉층에 각 256개의 노드를 사용한 ANN의 MSE가 0.000070으로 가장 낮게 나타났다.

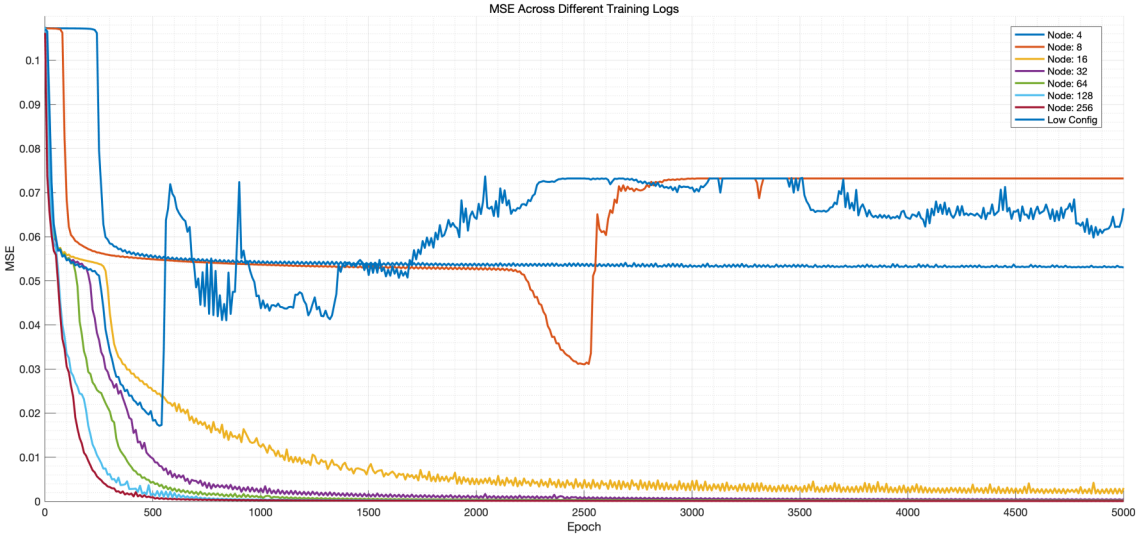

그림 4. Layer size = 10 에서의 레이어 당 노드 수에 따른 Epoch 당 MSE

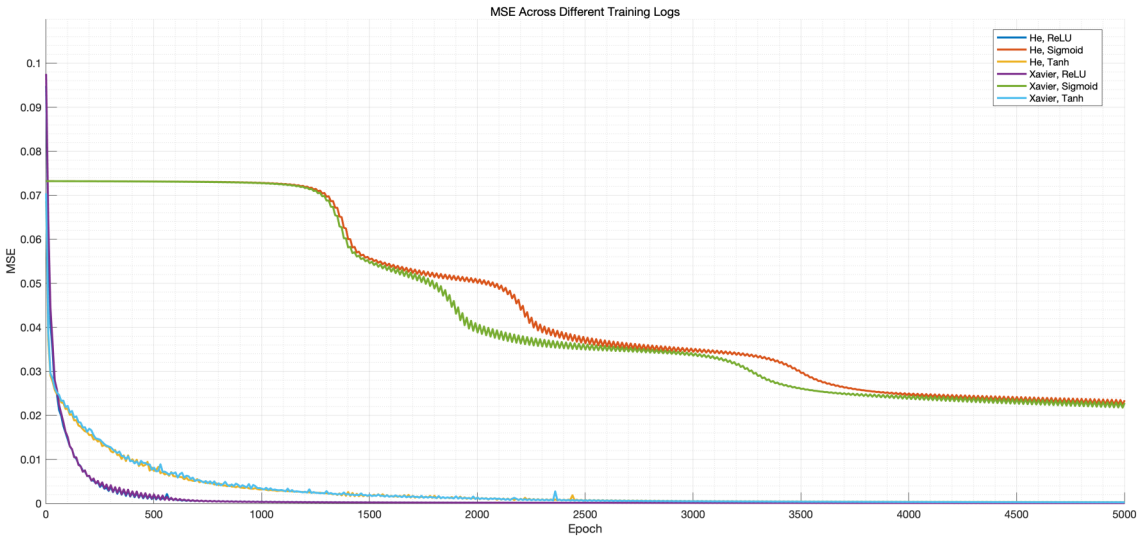

그림 5. Layer size = 6, Node per layer = 256 에서의 가중치 초기화와 활성화 함수에 따른 Epoch 당 MSE

4.2. 가중치와 활성화 함수에 따른 MSE

출력 레이어의 활성화 함수를 Sigmoid로 고정한 것을 제외하고 ReLU, Sigmoid, Tanh 세 개의 활성화 함수와 He, Xavier 두 개의 가중치 초기화 방식을 사용하였다. 이때 레이어는 6개, 각 레이어 당 노드는 256개로 고정하였다. 4.1.과 동일하게 데이터셋의 비율은 21:7이다.

표 6. 가중치 초기화와 활성화 함수에 따른 MSE

Activate Function	Weight Initialize	
	He	Xavier
ReLU	0.000088	0.000087
Sigmoid	0.02	0.022
Tanh	0.0003	0.0028

흥미롭게도 활성화 함수에 따라 MSE의 차이가 극명하게 나타났는데, Sigmoid와 Tanh를 사용한 것 보다 ReLU를 사용하였을 때 동일한 5000 epoch의 학습에 대해 MSE가 극명히 짧은 것을 볼 수 있었다.
Sigmoid와 Tanh에 대해서는 가중치 초기화 방식에 따라 MSE가 어느정도 차이가 있지만 ReLU를 사용하였을 때 He와 Xavier를 통한 가중치의 초기화에 대한 MSE의 값은 큰 의미가 있어보이지 않는다.

4.3. 학습, 테스트 셋의 비율에 따른 MSE

학습과 테스트 셋은 batch의 구성에 따라 å 이루어진다. 121을 batch의 사이즈로 구성함에 따라 28개의 batch가 만들어진다.
해당 batch를 기준으로 학습과 테스트 셋의 비율을 나눠 실험을 진행하고자 하였으나, 제출 마감 기한 내에 학습이 종료되지 않아 해당 실험에 대한 내용은 부재하다.
다만 학습 로그를 관찰함에 따라 결론적으로 셋 간의 큰 차이는 없을 것으로 보인다.

종합적으로 봤을 때 본 실험에서 구현한 모델의 경우 은닉층의 활성화 함수의 종류에 따라서는 MSE에 유의미한 영향이 있었지만 가중치를 초기화 하는 방식에서는 유의미한 차이가 발생하지 않았다. 또한 학습과 테스트 셋의 비율도 유의미한 차이를 발생시키지 않았다.
가장 크게 영향을 미친 것은 레이어의 수와 각 레이어 별 노드의 수이다. 기본적으로 각 레이어 당 노드의 수가 적을수록 MSE가 높은 값에서 유지되었으며, 노드의 수가 많을수록 값이 안정적으로 감소하는 것을 볼 수 있었다.
또한 레이어의 수에 있어서는 최대치로 제시한 16개의 레이어를 사용할 경우 정상적인 학습이 진행되지 않는 현상을 볼 수 있었으며, 본 실험에서 사용한 레이어의 수 중 12개의 레이어를 사용하였을 때 MSE가 가장 낮게 나왔다. 이에 12개의 레이어를 사용하고, 은닉층에는 256개의 노드를 둔 ANN이 가장 높은 성능을 보였다.
다만 각 인덱스 간의 유리수 상태의 간격이 $\frac{1}{30}$이라고 할 때 모든 값을 정확히 맞추는 상황이라고 한다면 각 출력값과 target 사이의 값은 $\frac{1}{60}$이내여야 하며, MSE를 사용하므로 모든 오차가 $\left(\frac{1}{60}\right)^2 = \frac{1}{3600} \approx 0.000277$ 이내의 값을 가질 때 정확한 모델이라고 할 수 있다. 이때 모든 값에 대해 일률적으로 0.00027 이내의 값을 가지는 것은 불가능하기 때문에 신뢰도가 높다고 하라면 MSE가 0.000277보다 작은 값을 가져야 한다.
현재 최적으로 도출된 모델의 MSE는 0.00007로 기준치의 $\frac{1}{4}$ 수준이다. 이를 통해 상당히 신뢰할 만한 모델임을 알 수 있다.
그러나 정확도가 대부분의 MSE가 신뢰도 있게 나오는 모델들에 대해서 11%대를 나타내는 것으로 보아 정확도를 판별하는 로직에 문제가 있는 것으로 보인다. 이에 대한 점검이 필요하다.

V. 결론

본 실험에서는 인공신경망(Artificial Neural Network, ANN)을 활용하여 특정 한국어 문자인식(OCR) 모델을 설계하고, 레이어와 노드의 구성, 가중치 초기화 방식, 활성화 함수, 학습 데이터셋 비율 등 다양한 요소가 성능에 미치는 영향을 분석하였다. 실험 결과는 다음과 같은 주요 결론을 도출할 수 있었다.

레이어와 노드의 수는 모델 성능에 가장 큰 영향을 미쳤다. 레이어의 수가 많아질수록 학습이 어려워지는 경향을 보였으며, 16개의 레이어를 사용한 경우 학습이 정상적으로 진행되지 않았다. 반면, 레이어가 12개이고, 각 레이어의 노드 수가 256개로 구성된 모델에서 가장 낮은 MSE(0.00007)를 기록하며 최적의 성능을 보였다. 이는 레이어의 수와 노드의 수가 적절히 균형을 이루는 것이 중요함을 시사한다.

활성화 함수는 모델 성능에 중요한 요소로 작용했다. ReLU를 사용한 경우 Sigmoid나 Tanh를 사용한 경우보다 유의미하게 낮은 MSE를 기록하였으며, 이는 ReLU가 비선형성을 잘 처리하고 그래디언트 소멸 문제를 효과적으로 완화했기 때문이다. 반면, He와 Xavier 초기화 방식 간의 성능 차이는 미미한 수준으로 나타나, 활성화 함수의 선택이 가중치 초기화보다 더 중요한 변수임을 알 수 있었다.

학습 초기에는 높은 학습률을 사용하고, 학습이 진행됨에 따라 학습률을 점진적으로 줄이는 방식이 안정적인 학습을 가능하게 했다. 활성화 함수와 가중치의 값을 클리핑하여 nan(Not a Number) 값이 발생하는 문제를 해결함으로써 모델 학습의 안정성을 확보할 수 있었다.

본 실험에서 최적 모델의 MSE는 0.00007로 기준치보다 월등히 낮아 신뢰할 만한 결과를 보였다. 그러나 정확도 측정에서는 상대적으로 저조한 11%대를 기록하여 정확도 계산 로직에 문제가 있을 가능성을 시사하였다. 이는 MSE와 정확도가 일치하지 않는 현상을 설명하며, 정확도 로직에 대한 추가 점검과 개선이 필요하다.

학습과 테스트 데이터셋의 비율에 따른 성능 차이는 크지 않을 것으로 예상되었으나, 제출 마감 기한으로 인해 충분한 실험 결과를 확보하지 못했다. 향후 이 비율에 따른 세부적인 분석을 통해 학습 안정성을 개선할 여지가 있다.

본 실험에서 구현한 ANN 모델은 초성, 중성, 종성 인식이라는 복잡한 문제를 다루면서도 낮은 MSE를 달성하며 상당히 신뢰할 만한 성능을 보였다. 특히, 최적화된 모델의 경우 MSE 기준으로 높은 신뢰도를 나타내었으며, 이는 제안된 ANN 구조가 주어진 문제에 적합함을 보인다.

본 실험에서 MSE와 정확도 간의 괴리를 해소하기 위해 정확도 로직을 점검하고 개선할 필요가 있다. 또한 학습과 테스트 데이터셋의 비율 변화가 성능에 미치는 영향을 추가적으로 분석하여 더 나은 학습 전략을 제안할 수 있을 것이다. 마지막으로, ANN의 구조를 최적화하고 CUDA와 같은 병렬 처리 기술을 도입하여 학습 속도를 향상시키는 것도 고려할 수 있다.

본 연구는 ANN의 동작 원리를 이해하고 최적 하이퍼파라미터를 도출하는 데 기여하며, 결과적으로 OCR 모델 설계와 구현에 대한 실질적인 통찰을 제공하였다.

Acknowledgement

본 보고서의 일부는 OpenAI의 Ghat GPT를 활용하여 작성되었음

본 보고서와 함께 제출되는 파일들 중 데이터 셋으로 사용된 이미지는

makeImage/exportImage

에 위치하며, C언어를 통해 작성된 학습 및 추론 모델의 소스 코드는

training_c

에 위치하며, Python을 통해 작성된 추론 모델의 소스 코드는

training_python

에 각종 함수를 비롯한 소스 코드들과 함께

main.py

에 작성되었다. C언어를 통해 작성된 학습 모델과 추론 모델은

training.c, inference.c

로 작성되었다. 학습을 통해 만들어진 가중치 파일은

training_init_activate

training_layer_node

두 개의 폴더에 저장되어 있으며, training_layer_node에는 4.1.의 가중치 파일로 파일 명 뒤에 붙는 두 개의 숫자는 레이어의 수와 각 레이어 당 노드의 수를 나타낸다. toLow가 붙은 경우는 표 5와 같이 레이어 별 노드의 수가 상이한 경우이다.

최적 모델로 도출한 가중치는

training_layer_node/weight_12_256.csv

이다.

모델의 성능 평가를 위한 추론 모델의 사용은 Python의 로직 문제로 인해 가중치를 불러와 글자를 추론하는 과정의 신뢰성이 저하됨에 따라 C언어로 작성된 inference.c를 사용하는 것을 권장한다.

컴퓨팅 환경은 3.15.에서 제시한 것과 같이 리눅스, 맥, WSL과 같은 환경에서 구동하여야 한다.

참 고 문 헌

[1] Yann LeCun, Leon Bottou, Yoshua Bengio, Patrick Haffner, "Gradient-based Learning Applied to Document Recognition," IEEE Proceedings, Vol.86, No.11, 2278-2324, Nov. 1998.

[2] Alex Graves, Jürgen Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," Advances in Neural Information Processing Systems, Vol.22, 545-552, 2009.

[3] Kai Wang, Boris Babenko, Serge Belongie, "End-to-End Scene Text Recognition," IEEE International Conference on Computer Vision, 1457-1464, Nov. 2011.

[4] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, "Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition," arXiv, arXiv:1406.2227, Jun. 2014.

[5] Baoguang Shi, Xiang Bai, Cong Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and its Application to Scene Text Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.39, No.11, 2298-2304, Nov. 2017.