인공지능 프로젝트 논문

임베디드시스템공학과 202201658 박소윤

초록

한글 문자 인식은 다양한 애플리케이션에서 중요한 역할을 하며, 본 연구에서는 한글 이미지 인식을 위한 신경망 모델의 성능을 최적화하기 위한 방법을 제시한다. 연구는 64x64 픽셀 크기의 흑백 한글 이미지를 사용하여, 픽셀 위치 변형, 원핫인코딩 라벨링, 노이즈 이미지 평가 등 다양한 실험을 수행하였다. 특히, 데이터셋의 확장과 다양한 변형을 통해 모델의 일반화 성능을 향상시키고, 최적의 가중치 파일 저장과 같은 최적화 기법을 통해 모델의 효율성을 극대화하였다. 또한, 실험을 통해 폰트 크기, 노드 개수, 학습률, 학습 환경에 따른 성능 변화를 비교하여, 모델이 다양한 환경에서 높은 정확도를 기록할 수 있음을 확인하였다. 본 연구는 노이즈가 포함된 이미지에서도 높은 정확도를 기록한 결과를 통해, 실제 환경에서의 적용 가능성을 높였으며, 향후 더 많은 폰트와 크기, 노이즈를 포함한 데이터셋을 사용하여 모델 성능을 더욱 개선할 수 있을 것이다. 이 연구는 한글 문자 인식 모델을 실시간 환경에서도 우수한 성능을 발휘할 수 있도록 최적화하고, 다양한 변형된 데이터를 처리할 수 있는 강력한 모델을 구축하는 데 기여할 것이다.

1. 서론

한글 문자 인식은 다양한 응용 분야에서 중요한 문제로, 손글씨 인식, 자동화된 문서 처리 및 디지털 아카이빙 등에서 광범위하게 활용된다. 그러나 한글은 고유한 구조와 형태적 특성으로 인해 문자 인식 시스템에서 특히 도전적인 문제를 제시한다. 본 연구는 이러한 문제를 해결하고자, 고유한 한글 이미지 데이터셋을 생성하고, 이를 기반으로 신경망 모델을 학습하여 높은 정확도를 달성하는 방법을 제시한다. 기존의 연구들은 주로 고정된 폰트와 이미지를 사용하여 모델을 학습시켜 왔으나, 본 연구에서는 픽셀 위치를 다양하게 변화시켜 데이터셋을 확장하고, 이를 통해 모델의 일반화 성능을 크게 향상시켰다. 또한, 다양한 폰트와 크기, 노이즈가 포함된 이미지를 사용하여 모델의 성능을 평가하고, 그 유의미한 결과를 도출하였다. 본 연구의 목표는 한글 문자 인식 모델의 정확도를 향상시키고, 향후 더 복잡한 실전 환경에서도 높은 성능을 발휘할 수 있도록 하는 것이다.

2. 연구방법

2.1. 연구 전체 흐름

본 연구는 한글 문자 인식을 위한 데이터셋 구축과 신경망 모델 학습 과정을 단계별로 설명한다. 첫 번째 단계에서는 다양한 폰트와 글씨 크기, 픽셀 위치 변형을 통해 데이터셋을 준비하였다. 두 번째 단계에서는 이 데이터를 이용해 신경망 모델을 학습시키고, 여러 하이퍼파라미터를 실험하여 최적의 모델을 도출하였다. 마지막으로, 학습된 모델을 다양한 환경에서 평가하여 성능을 분석하고, 실용적인 인식 시스템 개발을 위한 기초 자료를 제공하였다.

2.1.1. 한글 흑백 이미지 제작

한글 이미지는 64x64 픽셀 크기로 생성되었으며, 폰트 크기는 60으로 설정하여 흑백 BMP 이미지로 저장되었다. 각 이미지는 1비

트로 표현되어, 각 픽셀의 값은 흑(0)과 백(1)으로 구분된다. 이미지 데이터의 변형을 통해 모델의 일반화 성능을 높이기 위해, 픽셀 위치를 좌우 및 상하로 미세하게 변형하였다. 이를 통해 생성된 데이터셋은 원본 이미지의 다양한 버전을 포함하게 되며, 모델은 학습 과정에서 글자 위치와 변형에 대한 다양한 패턴을 인식하도록 학습한다. 이렇게 생성된 이미지는 그림 1처럼 115개의 고유한 폴더에 저장되며, 각 폴더 이름은 해당 글자의 원핫인코딩 라벨을 포함하고 있어, 각 이미지가 어떤 글자에 해당하는지 명확히 구분된다. 또한 그림 2와 같이 15개의 폰트를 적용했다. 이러한 과정들은 데이터셋의 다양성을 증가시키고, 모델이 특정 글자의 위치와 변형에 영향을 받지 않고 잘 일반화된 예측을 할 수 있도록 돕는다. 이를 통해 본 연구는 다양한 글자 위치와 변형을 포함한 데이터셋을 구성하여, 신경망 모델의 학습 효율성과 정확도를 향상시킬 수 있었다.

그림 1 115글자의 폴더

그림 2 15개의 폰트

2.1.2. 각 글자에 따른 원핫인코딩 실행

각 글자는 원핫인코딩 방식으로 라벨링되었으며, 이는 모델이 각 글자에 대한 고유한 출력 벡터를 예측하도록 돕는다. 그림 3처럼 115개의 글자는 고유한 폴더에 저장되며, 각 폴더에 속한 이미지는 해당 글자의 원핫인코딩된 라벨을 목표값으로 설정하여 순전파와 역전파 과정에 사용된다. 원핫인코딩된 라벨은 각 글자에 대해 정확한 클래스 정보를 제공하고, 이 값을 모델의 출력값과 비교하여 손실 함수를 계산하는 데 사용된다. 이 방식은 모델이 글자 인식 문제를 다중 클래스 분류 문제로 해결할 수 있도록 한다. 또한, 원핫인코딩된 라벨을 목표값으로 사용함으로써 모델은 학습을 진행할 때 각 글자에 대해 명확한 목표를 가질 수 있다.

본 연구에서는 기존의 이미지를 확장할 수 있도록 추가적인 폰트와 이미지 변형을 지원하는 시스템을 구축하였다. 이를 통해 향후 학습에 사용할 데이터셋의 다양성을 더욱 높일 수 있었으며, 데이터셋의 범위와 변형을 확장함으로써 모델의 일반화 능력을 강화할 수 있었다. 이러한 접근법은 모델이 다양한 폰트와 이미지 변형에 대해 잘 학습하고, 실험 환경에서의 성능을 극대화하는 데 중요한 역할을 하였다.

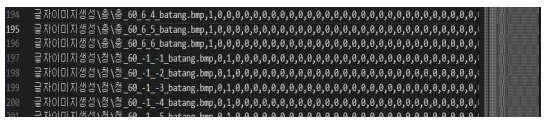

그림 3 115글자에 대해 원핫인코딩 결과

2.1.3. 순전파 및 역전파 수행

준비된 이미지 데이터셋을 로드하고 BMP 이미지 파일을 읽어들인 후, 데이터를 무작위로 훈련용(80%)과 테스트용(20%)으로 분리하여 순전파를 진행한다. 순전파 후 계산된 에러는 역전파를 통해 가중치를 업데이트하는 데 사용된다. 실험은 최적의 실험 방법을 적용하여 여러 에폭에 걸쳐 진행되며, 각 에포크마다 각 레이어의 출력값, 목표값, 오차,

최적의 가중치 등 다양한 정보를 파일에 저장한다. 또한, 학습 과정 동안 실시간으로 경과 시간, 현재 에포크, 배치 진행 상태 등을 터미널에서 확인할 수 있다.
모델 학습의 초기 단계에서 순전파 및 역전파가 제대로 작동하는지 확인하기 위해, 데이터셋의 일부분만 사용하였다. 이를 위해 단일 폰트에서 "충", "청", "남", "도" 네 글자의 픽셀 위치를 다양하게 변형한 이미지를 데이터셋으로 구성하였다. 표 2와 같은 환경에서 학습을 진행한 결과, 과적합이 발생한 것으로 나타났으나, 이는 모델이 학습을 제대로 수행하고 있음을 확인하는 단계로, 표 1에 나타난 결과를 통해 학습이 정상적으로 이루어졌음을 알 수 있다.

	에폭 1	에폭 10
훈련 정확도	5.31%	30.94%
테스트 정확도	66.25%	3.75%

표 1 4글자에 대한 학습 결과

2.1.4. 최적의 가중치 파일 전처리

순전파와 역전파를 통해 계산된 가중치는 학습 과정 중 일정 시점에서 수렴하게 된다. 이때, 훈련 정확도와 테스트 정확도가 가장 높은 지점에서 얻어진 가중치를 '최적의 가중치'로 정의한다. 최적의 가중치는 모델 학습의 성능을 결정짓는 중요한 요소로, 해당 가중치를 저장하고 이를 기반으로 평가를 진행할 수 있다.
최적의 가중치 파일에는 가중치 외에도 여러 레이어의 출력값, 목표값, 활성화 함수의 출력 등 다양한 정보가 포함되어 있어, 이들을 모두 포함하면 파일 크기가 매우 커질 수 있다. 따라서, 최적의 가중치를 포함한 파일에서 불필요한 데이터를 삭제하여 파일 크기를 최적화하고, 후속 분석 및 모델 배포에 필요한 공간을 절약할 수 있도록 하였다. 또한, 기본적으로 가중치는 float64 형식으로 저장되기 때문에, 이를 float 형식으로 변환하여 저장함으로써 파일 크기를 더욱 줄였다. 이러한 전처리 과정을 통해 저장 공간을 절약하고, 모델의 효율적인 관리와 빠른 로딩을 가능하게 하였다.
이 과정은 향후 모델을 실제 환경에 배포할 때, 더 빠르고 효율적인 추론을 지원하기 위해 필수적인 단계로, 모델의 최적화 및 성능 향상에 기여한다.

2.1.5. 평가

최적화된 가중치를 사용하여 순전파를 통해 평가를 진행한다. 평가 과정에서는 BMP 이미지 파일, 원핫인코딩으로 라벨링된 CSV 파일, 그리고 최적화된 가중치 파일이 필요하다. 먼저, CSV 파일을 통해 각 이미지 파일의 경로를 읽어들인 후, 해당 이미지를 모델에 입력하여 순전파를 진행한다. 순전파 후, 모델은 각 이미지에 대해 출력층에서 가장 높은 값을 도출하며, 이를 예측된 라벨로 사용한다.
이 평가 과정에서, 그림 4, 5를 살펴보면 예측된 라벨과 실제 라벨 간의 일치를 평가하여 모델의 성능을 검증한다. 예측된 라벨과 실제 라벨이 일치하는지 여부를 비교하여 정확도를 계산하고, 평가 결과를 기록하여 비교를 위한 파일에 저장한다. 이 파일에는 예측된 라벨과 실제 라벨의 일치 여부, 예측 벡터, 타겟 벡터 등 다양한 정보가 포함되며, 이를 통해 모델의 성능을 심층적으로 분석할 수 있다.

은닉층 활성화함수	출력층 활성화함수	초기화 방법	학습률	배치 크기	에폭 크기	레이어 노드
sigmoid	sigmoid	random	0.01	10	10	[4096, 128, 64, 4]

표 2 4글자에 대한 학습 환경

또한, 평가 결과는 향후 모델 개선을 위한 중요한 기준이 되며, 모델이 실제 데이터를 얼마나 잘 처리하는지에 대한 중요한 통찰을 제공한다. 이 과정을 통해 모델의 정확도뿐만 아니라, 다양한 데이터셋에 대한 모델의 일반화 능력도 평가할 수 있다.

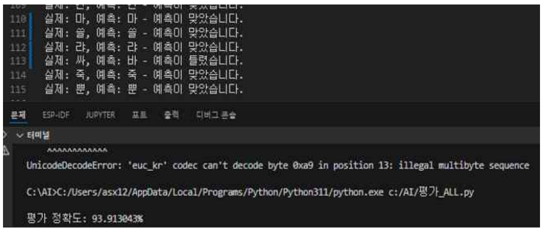

그림 4

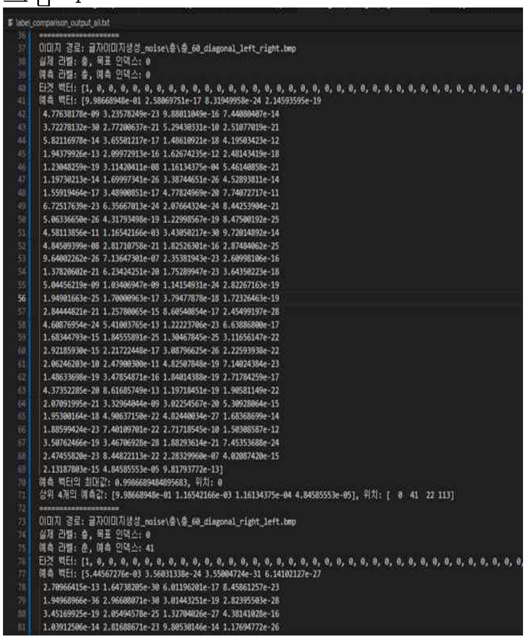

그림 5

2.2 최적의 가중치 찾기

최상의 한글 인식 결과를 얻기 위해서는 역전파를 통해 최적의 가중치를 찾아야 한다. 신경망 학습 과정에서 가중치는 모델의 성능을 결정짓는 중요한 요소로, 훈련 데이터에 맞는 최적의 가중치를 찾아내는 것이 필수적이다. 이를 위해 여러 비교 실험을 통해 다양한 환경과 설정에서 최적의 가중치를 구하는 방법을 적용하였다. 각 실험에서는 다양한 활성화 함수, 초기화 방법, 학습률 등의 하이퍼파라미터를 변경하여 최적의 성능을 도출할 수 있는 가중치 값을 찾아갔다. 이러한 과정을 통해, 모델의 정확도를 극대화하고, 한글 이미지 인식에서 뛰어난 성능을 발휘할 수 있는 최적의 가중치를 확보하였다.

2.2.1. 단일 폰트에 대해 최상의 한글 인식 환경 구하기

최적의 가중치를 구할 수 있는 환경을 한 폰트에 대해 우선 설정하고, 이후 폰트를 추가하여 최상의 결과를 얻는 것을 목표로 하였다. 이를 위해 단일 폰트의 115글자에 대해 픽셀이 다양한 이미지 약 15,000장을 학습시켰다. 실험 결과, 훈련 정확도 및 테스트 정확도는 표 3과 4에서 나타난 바와 같이 각각 0%, 1.086957%로 매우 낮았다. 처음에는 정확도가 낮다는 점에서 의심하지 않았으나, 여러 실험을 통해 학습 환경에 문제가 있음을 알게 되었다.

다양한 환경에서 실험을 반복했으나, 정확도는 항상 동일하게 낮은 값을 기록했다. 이를 해결하기 위해 실험 환경을 재검토했으며, 에폭마다 저장된 정보를 분석한 결과, 문제의 원인을 찾아낼 수 있었다. 표3, 4는 실험에 사용한 환경 설정을 나타낸다.

실험 결과 배열을 살펴본 결과, MSE(Mean Squared Error)가 가장 낮은 값으로 항상 인덱스 92에서 발생하는 것을 확인했다. 이에 대한 고찰 끝에, 데이터 분할 과정에서 문제가 발생했음을 알게 되었다. 데이터를 80%와 20%로 나누는 과정에서 데이터가 무작위로 섞이지 않고 순차적으로 학습이 진행된 것을 발견하였다. 순차적인 학습 방식으로 인해 과거의 학습 결과는 잊혀지고, 80번째 학습 결과만이 반영되어 평가가 이루어진 것이다.

이 문제를 해결하기 위해서는 데이터를 무작위로 섞어 학습해야 한다는 결론에 도달했다. 또한, 출력층 활성화 함수에 대한 선택도 중요한 영향을 미쳤다. sigmoid 함수를 사용했을 때는 정확도가 0%로 나타났고, softmax를 사용했을 때는 정확도가 1.08%

로 나타났다. 이로 인해 학습 방식이 잘못 설정되었음을 깨닫게 되었으며, 향후 실험에서는 이러한 오류를 수정하여 정확한 학습을 수행할 수 있었다.
결론적으로, 표5에서 제시된 환경이 최적의 가중치를 갖는 환경임을 확인할 수 있었다. 이 결과에 대한 근거와 비교 실험은 다음 단계에서 자세히 설명할 예정이다.

2.2.2. 최상의 한글 인식 환경 비교 실험

2.2.2.1. 활성화 함수와 초기화 방식의 차이

초기 실험에서는 표 5와 같은 환경에서 단일

은닉층 활성화함수	출력층 활성화함수	초기화 방법	학습률	배치 크기	에폭 크기	레이어 노드
sigmoid	sigmoid	random	0.0001	32	100	[4096, 512, 256, 128, 64, 115]
sigmoid	sigmoid	xavier	0.001	32	100	[4096, 512, 256, 128, 64, 115]
sigmoid	sigmoid	random	0.001	32	100	[4096, 512, 256, 128, 64, 115]
sigmoid	sigmoid	random	0.01	32	100	[4096, 256, 128, 64, 115]
relu	sigmoid	random	0.01	32	100	[4096, 256, 128, 64, 115]
relu	sigmoid	he	0.05	32	100	[4096, 256, 128, 64, 115]
tanh	sigmoid	random	0.001	32	100	[4096, 256, 128, 64, 115]
tanh	sigmoid	xavier	0.01	32	100	[4096, 256, 128, 64, 115]
tanh	sigmoid	xavier	0.05	32	100	[4096, 256, 128, 64, 115]

표 3 훈련 정확도 및 테스트 정확도가 0%인 환경

은닉층 활성화함수	출력층 활성화함수	초기화 방법	학습률	배치 크기	에폭 크기	레이어 노드	Cross-Entropy
sigmoid	softmax	xavier	0.05	32	100	[4096, 512, 256, 128, 64, 115]	X
tanh	softmax	xavier	0.05	32	100	[4096, 512, 256, 128, 64, 115]	X
tanh	softmax	xavier	0.0001	32	100	[4096, 512, 256, 128, 64, 115]	X
relu	softmax	he	0.05	32	100	[4096, 256, 128, 64, 115]	O
relu	softmax	he	0.01	32	100	[4096, 256, 128, 64, 115]	O
relu	softmax	he	0.05	32	100	[4096, 256, 128, 115]	O

표 4 훈련 정확도 및 테스트 정확도가 1.08%인 환경 설정

은닉층 활성화 함수	출력층 활성화함수	초기화 방법	학습률	배치 크기	에폭 크기	레이어 노드	훈련 정확도	테스트 정확도
relu	softmax	he	0.001	64	20	[4096, 512, 256, 126, 115]	99.88%	99.64%

표 5 단일 폰트에서 최적의 가중치를 갖는 환경

폰트를 사용하여 학습을 진행한 결과, 예상보다 우연히 높은 정확도를 보였다. 이후, 단일 폰트에서 15개의 폰트를 추가하여 학습했을 때, 가장 높은 정확도를 기록하였고, 이는 모델이 다양한 폰트를 효과적으로 학습할 수 있음을 시사했다. 이 실험은 비교 실험을 통해 해당 환경이 최적인 이유를 증명하는 과정으로 이어졌으며, 실험 결과에서 얻은 인사이트는 연구를 보다 직관적으로 이해하는 데 도움이 될 것이다.

그림 6, 7, 8은 15개의 폰트와 115개 글자, 다양한 픽셀 설정을 적용한 183,540장의 이미지 데이터에 대한 결과를 나타낸다. 표 6에서 제시된 공통 환경에서, 은닉층 활성화 함수와 초기화 방법에 따라 다른 결과를 보였다. 그림 6과 7는 높은 정확도를 보인 반면, 그림 8은 상대적으로 낮은 정확도를 기록했다. 이를 통해 ReLU 함수와 He 초기화, tanh 함수와 Xavier 초기화 방법에서 학습 성능이 높다는 것을 확인할 수 있었다.

출력층 활성화 함수	학습률	배치 크기	에폭 크기	레이어 노드
soft max	0.001	64	20	[4096, 512, 256, 126, 115]

표 6 15폰트, 공통 환경

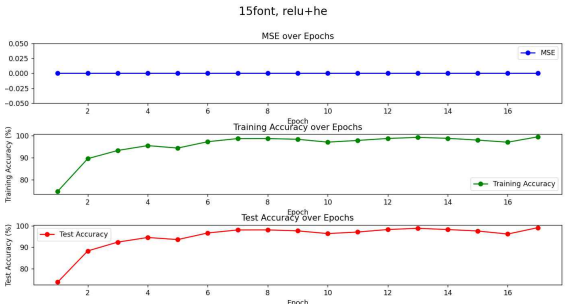

그림 6 15폰트, relu함수와 he초기화

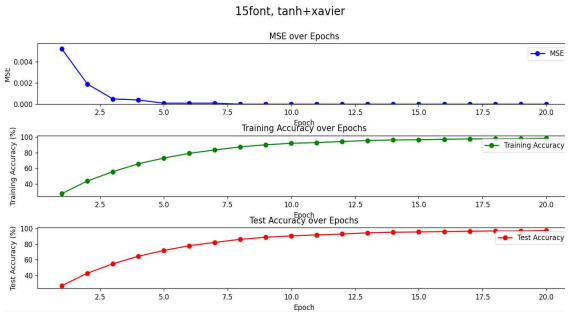

그림 7 15폰트, tanh함수와 xavier초기화

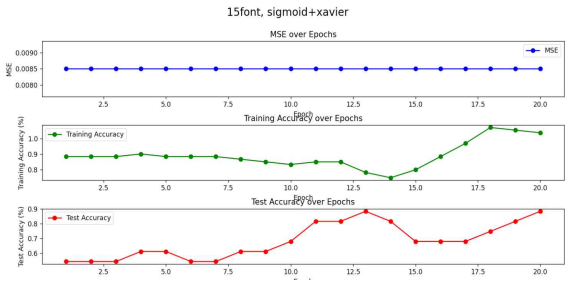

그림 8 15폰트, sigmoid함수와 xavier초기화

2.2.2.2. 픽셀의 다양성 비교

이미지 픽셀을 다르게 설정하여 학습시킴으로써 학습 데이터에 다양성을 부여하였다. 픽셀의 다양성이 유의미하다는 점을 설명하기 위해, 15개의 폰트에 대해 표 5와 같은 동일한 환경에서 픽셀 다양성의 포함 여부에 따라 학습을 진행하였다. 그림 9은 좌우 및 상하의 픽셀 움직임을 포함한 183,540장의 이미지로 학습한 결과를 나타내며, 그림 10은 픽셀의 움직임을 포함하지 않은 1,610장의 이미지로 학습한 결과를 보여준다. 그림 9은 높은 정확도를 기록한 반면, 그림 10은 상대적으로 낮은 정확도를 보였다. 참고로, 그림 11은 그림 10의 정확도가 낮아 약 200번의 에폭을 진행한 결과이다. 훈련 정확도가 테스트 정확도의 2배로 나타난 것을 확인할 수 있으며, 이를 통해 픽셀의 다양성이 학습 성능에 중요한 영향을 미친다는 것을 알 수 있었다. 따라서, 본 연구는 이 방식을 기반으로 학습을 이어나갔다.

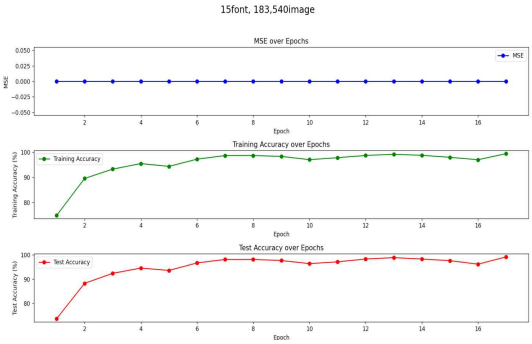

그림 9 15폰트, 픽셀의 다양성 포함

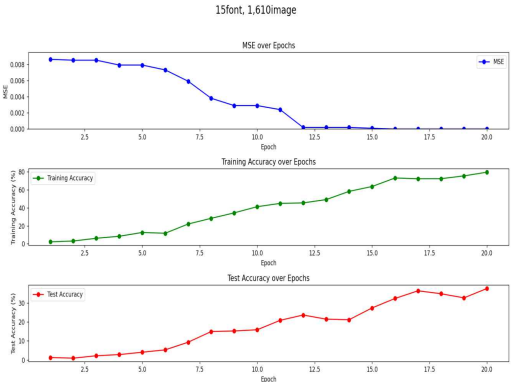

그림 10 15폰트, 픽셀의 다양성 미포함

2.2.2.3 노드의 개수 비교 실험

초기 연구 단계에서, 은닉층 노드의 개수를 256개로 제한해야 한다는 점을 정확히 인지하지 못하여, 입력층과 출력층을 제외한 은닉층 노드 수를 512로 설정하여 표 7의 조건으로 실험을 진행하였다. 이번 비교 실험은 은닉층 노드 수를 512에서 256으로 변경하여 수행한 결과를 다룬다. 표 5와 같은 동일한 환경에서 은닉층 노드 수만을 변경하여 실험을 진행하였다.

그림 12와 그림 13을 비교한 결과, 두 실험에서의 성능이 유사한 결과를 보였으며, 256 노드 제한에서도 학습 성능이 충분히 잘 드러난다는 것을 확인할 수 있었다.

또한, 은닉층의 256, 126, 64 노드 설정에서도 높은 정확도를 기록하여, 모델이 여전히 인공신경망으로서 충분히 효과적인 역할을 수행하고 있음을 알 수 있었다. 이는 네트워크 구조의 최적화와 노드 수가 학습 성능에 미치는 영향을 재확인할 수 있는 중요한 실험 결과로, 향후 모델의 구조를 결정하는 데 중요한 기준이 될 것이다.

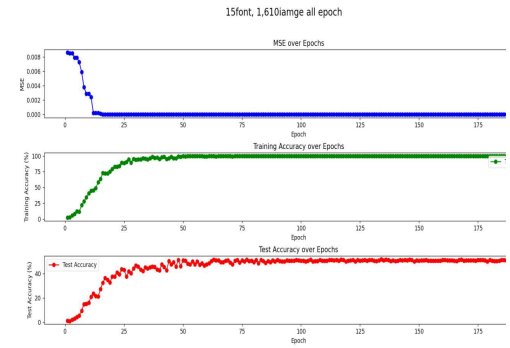

그림 11 그림 10의 모든 에폭 결과

	노드 수 변경
변경 전	[4096, 512, 256, 126, 115]
변경 후	[4096, 256, 256, 126, 115]
다른 경우	[4096, 256, 126, 64, 115]

표 7 노드 수 제한 규칙에 따른 변경

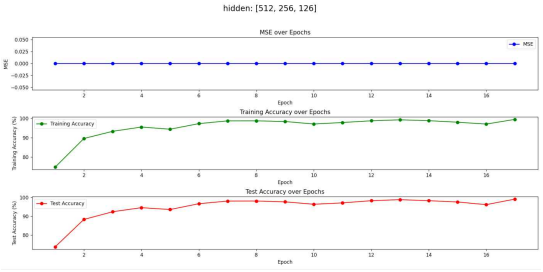

그림 12 노드:[4096, 512, 256, 126, 115]

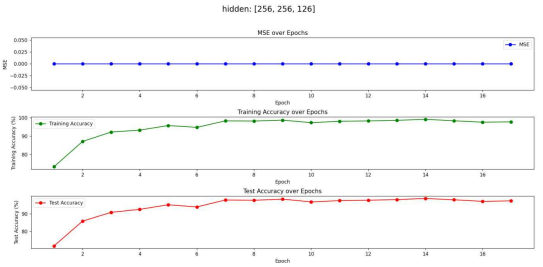

그림 13 노드:[4096, 256, 256, 126, 115]

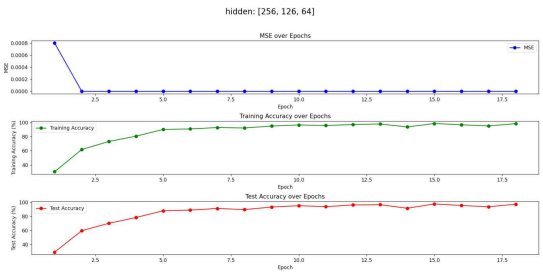

그림 14 노드:[4096, 256, 126, 64, 115]

2.2.2.4 학습률의 비교 실험

본 실험에서는 학습률이 모델 학습에 미치는 영향을 평가하고, 표 5의 실험 환경에서 설정된 학습률이 적절한지 확인하기 위해 학습률을 0.001을 기준으로, 0.005와 0.0001에 대해 실험을 진행하였다.

그림 15에서 학습률 0.001을 사용할 경우, 에폭 20이 되지 않아도 정확도가 빠르게 수렴하는 것을 확인할 수 있었다. 이는 학습률 0.001이 비교적 빠르게 높은 정확도를 달성하는 것을 의미한다. 반면, 그림 17에서 학습률 0.0001을 사용한 경우, 에폭 20에서는 정확도가 낮았으나, 이후 에폭에서도 정확도가 꾸준히 증가하는 것을 확인할 수 있었다. 그림 16에서는 학습률 0.005를 사용한 실험 결과를 보여준다. 학습 초기에는 정확도가 약 70% 정도로 정체되어 있으며, 이는 학습률이 너무 커서 모델이 최적의 가중치를 찾는 데 어려움을 겪는 것으로 해석될 수 있다. 특히, 학습률이 급격히 낮아지는 구간이 중간중간 발생하며, 이는 모델이 최적화 과정에서 과도한 조정을 반복하고 있다는 것을 시사한다. 이러한 현상은 다른 학습률에 비해 정확도가 상대적으로 낮은 상태에서 멈추는 경향을 나타내며, 이는 학습률 0.005가 모델 학습에 적합하지 않다는 것을 의미한다.

이 실험을 통해, 학습률 0.001이 다른 학습률에 비해 빠르게 높은 정확도를 도달할 수 있다는 것을 확인하였으며, 표 5에서 설정된 학습률 0.001이 최적의 가중치를 도출하는 데 가장 적합한 값임을 알 수 있었다.

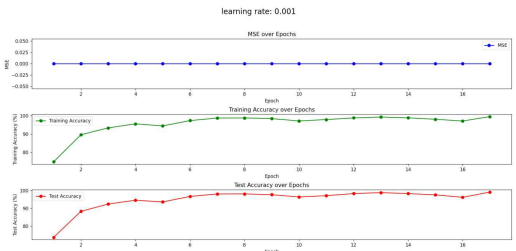

그림 15 학습률: 0.001

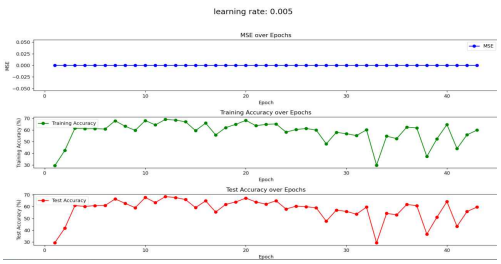

그림 16 학습률: 0.005

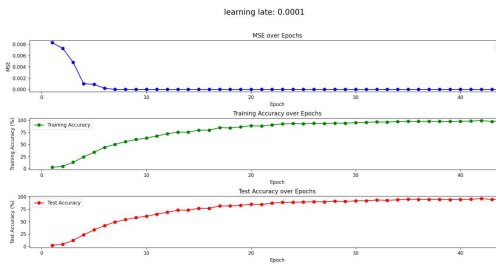

그림 17 학습률 0.0001

3. 결과

3.1. 새로운 폰트 학습에 대한 성능 향상

총 15개의 폰트와 183,540장의 흑백 이미지를 사용하여 표 5의 환경을 가지고 학습을 진행하였다. 최적의 가중치를 찾고, 이를 업데이트한 후 평가를 수행할 수 있는 환경을 구축하였다. 평가 결과는 꽤 높은 성능을 보였다. 표 7은 학습된 인공신경망에 새로운 폰트를 추가하여 평가한 결과를 보여주며, 새로운 폰트에 대한 학습 전후 결과를 비교한다. 첫 번째로 굴림 폰트에 대해 학습한 후, 해당 인공신경망에 새로운 폰트를 추가하여 학습시킨 내용과 정확도는 표 7에서 확인할 수 있다. 새로운 폰트에 대한 학습 전의 결과를 살펴보면, 정확도가 점차적으로 향상되고 있는 것을 확인할 수 있다. 이는 한글 이미지 데이터가 많아질수록, 새로운 폰트에서도 높은 정확도를 얻을 수 있다는 중요한 결과를 시사한다. 참고로, 표 7의 중간 부분에서는 학습 시간이 길어서 정확도가 점진적으로 증가하는 과정을 생략하고, 여러 폰트를 한 번에 추가하여 학습을 진행한 상태를 나타낸다. 한글 이미지 데이터의 양이 많아질수록 새로운 폰트에서 높은 정확도를 기록하는 또 다른 중요한 결과는 다음 단계에서 더 자세히 설명할 것이다.

3.2. 새로운 폰트 학습에 대한 성능 결과

3.2.1. 학습된 폰트의 노이즈 이미지 성능 평가

본 연구에서는 노이즈가 없는 순수 한글 이미지를 사용하여 학습을 진행하였다. 이는 모델이 한글 문자 인식에서 기본적인 성능을 발휘할 수 있도록 하기 위함이었다. 이후, 실험에서는 그림 11과 같이 대각선, 가로선, 세로선 형태의 노이즈를 이미지에 추가하여 평가를 진행하였다. 노이즈는 이미지의 다양한 부분에 고르게 분포하도록 적용되었으며, 이는 실제 환경에서 발생할 수 있는 다양한 유형의 노이즈를 모델이 처리할 수 있는지 확인하기 위한 의도로 진행되었다.

폰트 이름	학습 전 정확도	학습 후 정확도
굴림	-	99.24%
바탕	19.13%	99.38%
휴먼옛체	20.00%	99.80%
휴먼편지체	26.08%	99.76%
휴먼엑스포	27.82%	98.97%
휴먼아미체	28.37%	98.94%
HYGungSo 굵게	29.56%	98.98%
HYGothic 중간	32.01%	-
HYGraphic	-	-
휴먼매직체	-	-
휴먼둥근 헤드라인	-	-
휴먼모음T	-	-
HY견명조	-	-
HYShort Samul	-	-
HY견고딕	-	99.14%

표 8 새로운 폰트의 학습 전, 후 정확도 비교

115개의 글자에 대해 노이즈를 추가한 이미지를 표 5의 환경에서 평가한 결과, 그림 12에서 확인할 수 있듯이, 모델은 96.52%의 정확도를 기록하였다. 이는 본 연구에서 학습된 인공신경망이 노이즈가 없는 이미지뿐만 아니라, 노이즈가 포함된 이미지에 대해서도 우수한 성능을 보인다는 것을 시사한

다.

3.2.2. 15개의 폰트 학습에 새로운 폰트 학습 결과

15개의 폰트로 표 5의 환경에서 학습을 진행한 후, 새로운 여러 폰트에 대해 평가를 수행하였다. 표 9에서 보듯이, 새로운 폰트에 대한 정확도는 예상보다 낮은 값을 기록했지만, 표 7과 비교했을 때, 새로운 폰트에 대한 정확도가 상대적으로 높다는 것을 확인할 수 있었다. 이는 모델이 학습한 폰트의 범위가 확장되었음에도 불구하고 여전히 높은 수준의 성능을 유지하고 있다는 것을 의미한다. 이러한 결과는 모델이 새로운 폰트에 대해서도 일정 수준 이상의 일반화 능력을 가지고 있음을 시사한다.

따라서, 더 높은 정확도를 원한다면, 새로운 폰트에 대해서 픽셀의 위치 변화와 같은 데이터 변형을 통해 데이터셋을 다양화하는 방식이 효과적일 것이라는 결론을 얻을 수 있었다. 이러한 데이터 확장은 모델의 성능을 개선하고, 다양한 폰트에 대해서도 정확한 예측을 가능하게 할 것이다.

새로운 폰트	정확도
LG PC 보통	55.62%
HYPMokGak	76.52%
HYHeadLine	93.91%
HYSinMyeongJo	96.65%

표 9 새로운 폰트 평가 정확도 결과

3.2.3. 글씨 사이즈 변화의 성능 결과

기존 연구들보다 높은 정확도를 기록한 결과에 대해 그 원인을 고찰한 결과, 글씨 사이즈가 중요한 변수로 작용할 수 있다는 생각을 하게 되었다. 본 연구에서는 64x64 픽셀 크기에 맞춰 글씨 사이즈를 60으로 설정하여 모든 이미지 데이터를 생성하였다. 이를 통해 일관된 크기의 글씨가 학습에 영향을 미친 것으로 추측하고, 이에 대한 실험을 진행하였다. 실험에서는 15개의 폰트에 대해 60 크기의 글씨 이미지를 40 크기로 조정하여 평가하였다. 이때, 글씨 크기가 40인 이미지에 대해 표 5와 같은 환경에서 평가를 진행한 결과, 표 10에 나타난 바와 같이 12.54%의 정확도를 기록하였다. 이어서 글씨 크기 30, 50, 80에 대한 학습을 진행하고, 다시 글씨 크기 40인 이미지에 대해 평가한 결과, 상대적으로 높은 정확도를 보였다. 이를 통해 글씨 크기에 따라 학습을 진행하면 더 높은 정확도를 얻을 수 있다는 것을 확인할 수 있었다.

학습 전 정확도	학습 후 정확도
12.54%	72.42%

표 10 사이즈 학습에 대한 평가 결과

4. 결론

본 연구에서는 한글 문자 인식 성능을 최적화하기 위해, 다양한 실험을 통해 모델의 성능을 평가하고, 최적의 가중치와 구조를 찾기 위한 방법을 제시하였다. 실험을 통해 얻어진 주요 결과는 다음과 같다.

첫째, 데이터셋의 다양성은 모델의 성능에 중요한 영향을 미친다는 것을 확인하였다. 픽셀의 위치를 좌우 및 상하로 미세하게 변형하여 데이터셋을 확장함으로써, 모델은 다양한 글자 위치와 변형을 효과적으로 학습할 수 있었다. 이러한 데이터셋 확장은 모델의 일반화 성능을 크게 향상시켰으며, 실제 환경에서의 적용 가능성을 높였다.

둘째, 원핫인코딩을 통한 라벨링 방식은 모델의 목표값을 명확히 하고, 모델이 각 글자에 대한 정확한 출력을 생성하도록 돕는 중요한 요소였다. 이를 통해 모델은 각 글자에 대해 명확한 목표를 설정하고, 학습 과정에서 더욱 정확한 예측을 할 수 있었다.

셋째, 최적의 가중치 파일을 저장하는 과정에서 불필요한 데이터를 제거하고, 저장된 가중치를 최적화함으로써 모델의 효율성을 높일 수 있었다. 특히, 가중치를 float 형식

으로 변환하여 저장함으로써 파일 크기를 최적화하고, 추후 모델 배포 시 빠른 로딩 속도를 달성할 수 있었다.

넷째, 노이즈가 포함된 이미지에 대해서도 높은 정확도를 기록한 본 연구의 모델은, 노이즈가 없는 학습 데이터뿐만 아니라 다양한 변형된 데이터에 대해서도 우수한 성능을 보였다. 이는 모델이 실제 환경에서 발생할 수 있는 다양한 유형의 데이터를 잘 처리할 수 있다는 점에서 중요한 결과로 평가된다.

다섯째, 폰트 크기와 글씨 크기 변화를 실험한 결과, 글씨 크기에 따른 학습 성능의 차이를 확인할 수 있었다. 특히, 글씨 크기를 다르게 설정한 실험에서는 글씨 크기 40에서 높은 정확도를 기록했으며, 이는 모델이 다양한 크기의 글자에 대해서도 높은 인식 성능을 발휘할 수 있다는 가능성을 제시한다.

여섯째, 학습률에 대한 비교 실험을 통해 모델의 성능 최적화가 가능하다는 것을 확인하였다. 실험에서는 학습률 0.001이 가장 효과적임을 확인했으며, 이는 모델이 빠르게 높은 정확도를 기록하는 데 중요한 요소였다. 반면, 학습률 0.005는 정확도가 급격히 낮아지는 경향을 보였고, 0.0001은 학습 속도가 느리지만 시간이 지남에 따라 정확도가 꾸준히 증가하는 경향을 보였다. 이러한 실험 결과는 학습률이 모델 성능에 중요한 영향을 미친다는 것을 시사하며, 적절한 학습률 설정이 최적의 가중치를 찾는 데 중요한 역할을 한다는 점을 강조한다.

마지막으로, 은닉층 노드 수에 대한 실험에서는 256 노드 제한에서도 충분히 높은 정확도를 보였으며, 이는 네트워크 구조가 모델 성능에 미치는 영향을 잘 보여주는 결과였다. 모델의 은닉층 구조가 256, 126, 64 노드 설정에서도 높은 성능을 나타낸 것을 통해, 과도한 노드 수보다는 최적의 노드 수를 찾는 것이 중요함을 알 수 있었다.

종합적으로 본 연구는 한글 문자 인식 문제에서 다양한 실험과 결과를 통해 모델의 성능을 극대화할 수 있는 방법을 제시하였다. 향후 연구에서는 더 많은 폰트와 크기, 노이즈를 포함한 다양한 데이터를 사용하여 모델을 더욱 발전시킬 수 있을 것이다. 또한, 실시간 인식 시스템에 적용하기 위한 최적화 작업을 추가적으로 진행하여, 실제 환경에서도 우수한 성능을 발휘할 수 있도록 할 예정이다.