# CNN 기반 손글씨 데이터 분류를 위한 신경망 모델 설계 및 분석

## (Design and Analysis of Neural Network Models for CNN-Based Handwriting Data Classification)

장 진 영, 강 성 욱, 백 현 우, 박 상 은
(Jinyeong Jang, Sungwook Kang, Hyunwoo Baek, Sangeun Park)

Soongsil University

**요 약** 최근 인공지능이 발전함에 따라 인공신경망을 활용한 데이터 분류 모델에 대한 많은 연구가 진행되고 있다. 그 중에서도 손글씨를 인식하고 분류하는 모델은 여러 분야에 활용가능하므로 정확하고 강건한 모델에 대한 중요성이 높아지고 있다. 본 연구는 Extended MNIST(EMNIST) 데이터셋을 이용한 합성곱 신경망(Convolutional neural network; CNN) 모델 최적화를 목표로 신경망 모델을 제안한다. 결과적으로 Baseline 모델과 Pre-trained 모델에 비해 명확하지 않은 데이터에도 강건한 분류가 가능함을 확인하였다.

**키워드 :** Extended MNIST, 합성곱신경망, 분류, 심층학습, 최적화

***Abstract*** With the recent development of artificial intelligence, many studies have been conducted on data classification models using artificial neural networks. Among them, the model for recognizing and classifying handwriting can be used in many fields, so the importance of accurate and robust models is increasing. This study proposes a neural network model with the aim of optimizing a convolutional neural network (CNN) model using the Extended MNIST (EMNIST) dataset. As a result, it was confirmed that robust classification is possible even for unclear data compared to baseline and pre-trained models.

**Key words :** Extended MNIST, Convolutional neural network , Classification, Deep learning, Optimization

장진영 (20202995), 강성욱 (20192562), 백현우 (20192596), 박상은 (20211522)

## I. INTRODUCTION

손글씨를 분류하는 모델의 설계는 인공지능 분야에서 기초적인 연구로 거론된다[1]. 손글씨 분류 모델은 문자 인식, 자동 번역 등에 활용되기 때문에 명확하지 않은 데이터도 분류가능한 강건한 모델이 요구된다. 본 연구는 손글씨 분류를 위한 CNN 기반 인공신경망 모델을 제안한다. 학습은 알파벳과 숫자로 구성된 EMNIST[2]가 사용되었으며, 모델 선정 과정 및 최종 모델 설계 과정을 설명한다. 특히 제안하는 모델에 대해 이전 모델 및 기본 모델과의 성능을 비교하고자 한다.

구성은 다음과 같다. 먼저 II장에서 ResNet-50 과 LeNet5 을 포함한 Baseline 모델과 Pre-trained 모델 연구를 살펴본다. 이후 III장에서 최종 모델 설계 과정과 구조를 소개하며 IV장에서는 최종 모델의 성능 결과를 확인한다. 마지막으로, V장에서는 본 연구의 결론을 맺는다.

## II. BACKGROUND

EMNIST 데이터셋은 6개의 카테고리로 분류가능하며 각 카테고리 별로 클래스와 데이터의 수가 상이하다. 그 중에서도 Bymerge 데이터셋과 Balanced 데이터셋은 각각 학습 데이터가 많고 클래스 수가 적다는 특징과 클래스 별 데이터 양이 균일하고 얇은 모델로 높은 성능을 낼 수 있는 특징을 가진다. 그러므로 Baseline 모델의 결과에 따라 이후 진행되는 연구에 사용할 데이터셋을 선정한다.

### 2.1 Baseline Model: ResNet-50 and LeNet5

ResNet-50[3]은 Short-cut 구조로 Residual Learning을 구현한 모델이다. LeNet5[4]는 CNN과 딥러닝의 발전을 가속화시킨 모델로, 기본적인 CNN 구조로 이루어져 있다. 두 모델에 대해 Bymerge 데이터셋과 Balanced 데이터셋을 학습한 후 결과를 비교한다. Fig. 1은 학습 진행에 따른 정확도를 나타낸 그래프이고 각 그래프에서 주황색 실선은 검증 정확도를 의미하고 파란색 실선은 학습 정확도를 의미한다. 이때 (a)와 (b)는 ResNet-50 모델이고 (c)와 (d)는 LeNet5 모델에 대한 학습 결과이다. 해당 그래프를 통해 Balanced 데이터셋을 사용할 때 비교적 안정적인 학습이 가능함을 확인할 수 있다.

주목할 만한 점은 ResNet-50 모델이 복잡한 데이터셋에 높은 성능을 보이는 복잡한 구조로 이루어져 있다는 것이다. 본 연구에 사용하는 EMNIST 데이터셋은 기존 ImageNet에 비해 Size와 Channel 수 등 Input feature가 작으므로 ResNet-50 모델 구조보다 단순한 모델이 필요하다. 추가적으로 학습이 진행되는 동안 학습 정확도가 완만하게 증가하는 경향성을 보였고, 노이즈에도 강건한 모델을 설계해야 하므로 본 연구는 데이터 증강 기법을 도입함으로써 이를 개선한다.

### 2.2 Data Augmentation

데이터 증강 기법은 Fig. 2와 같이 기존 데이터를 변형

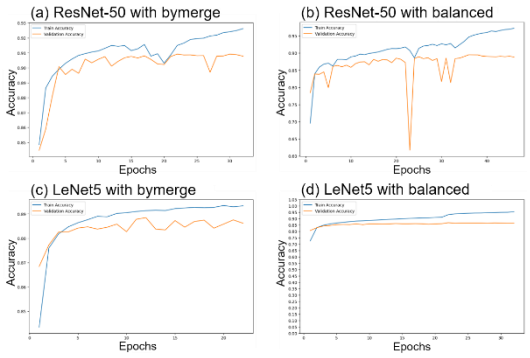


Fig. 1: Baseline model 실험 결과

한 데이터를 추가함으로써 학습 데이터의 양을 증가시킬 수 있다. 이를 통해 낮은 과적합 가능성과 높은 학습 안정성을 기대한다. Table. 1은 증강 데이터 생성 객체의 파라미터 구성을 나타낸 표이다. 각 파라미터는 Rotation, Width shift 그리고 Height shift를 의미한다. 따라서 본 연구는 기존 Balanced 데이터셋에서 각 모델을 이용한 데이터 증강 기법으로 네 배 증가한 데이터셋을 이용하여 제안하는 모델을 학습한다.

### 2.3 Pre-trained Model

본 연구는 EMNIST 데이터를 성공적으로 분류할 수 있는 CNN 모델 설계를 목표로 한다. 모델 구조 설계에 앞서 대규모 학습 데이터에 기반한 사전 훈련 모델의 성능을 비교해봄으로써 성능을 높일 수 있는 모델 구조에 대해 확인한다. 2.1에 이어서 Pre-trained 모델에 EMNIST Balanced 데이터셋으로 학습함으로써 성능을 비교한다. Pre-trained 모델의 경우 이미지 분류에 자주 사용되는 모델로 선정하였으며 각각은 (1)ResNet-34, (2)DenseNet, (3)EfficientNet, 그리고 (4)MobileNet 모델을 사용한다. 각 모델 별 실험 결과는 Table. 2에서 확인가능하다.

Pre-trained 모델을 통한 학습은 유사한 데이터셋으로 학습된 모델에 대해 Lower layer 를 그대로 이용하고, 기존 파라미터를 이용함으로써 학습 시간을 단축하는 것에 의의가 있다. 그러므로 기존 ImageNet 을 통해 학습한 Pre-trained 모델의 입력 Feature 크기에 맞게 학습 데이터를 일치시켜야 한다. 그러나 모든 학습 데이터를 확장해야 하므로 메모리 사용으로 인한 에러가 발생한다. 이에 대한 해결 방안으로 확장된 데이터를 저장하는 것이 아닌 Lower layer 를 추가하여 데이터 확장을 위한 레이어를 모델에 더하는 방법을 고안했다. 그럼에도 불구하고 확장을 위한 함수에서 Compile 함수를 지원하지 않아 문제가 발생한다. 결국 Pre-trained 에 대한 학습은 모델의 구조만을 불러와서 진행하게 되었다.

Pre-trained 모델의 학습에서 유의미한 학습 결과 비교를 위해 동일한 하이퍼파라미터로 학습을 진행한다. Baseline LeNet 을 포함한 다섯 개 모델에 대해 비슷한 정확도를 나타낸다. 그러나 학습 시간 및 추론 시간의 경우 상대적으

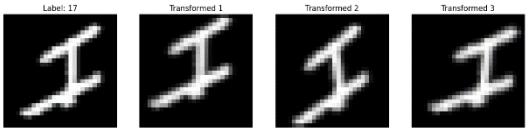


Fig. 2: 데이터 증강 기법 적용 예시

Table 1: 데이터 증강에 사용된 객체 별 파라미터

| Generator | Rotation | Width shift | Height shift |
|---|---|---|---|
| A | 10 | 0.2 | 0.2 |
| B | 15 | 0.1 | 0.1 |
| C | 20 | 0.05 | 0.05 |

로 간단한 모델 구조를 가진 LeNet에서 높은 성능을 보인다. Pre-trained 모델 학습 결과를 바탕으로 최종 모델 설계 시 LeNet과 같이 레이어의 복잡성을 줄이면서 정확도를 높일 수 있는 방안을 고안한다면 전반적인 성능을 향상할 수 있다고 예상한다.

## III. Proposed Model

본 절에서는 제안하는 CNN 모델 설계 과정과 모델 구조를 설명한다. 우선적으로 시간의 흐름에 따라 모델의 발전 과정을 나열하며 최종적인 제안 모델의 구성과 주요 설정 파라미터를 소개한다.

### 3.1 Model Design Process

Inception module은 Google에서 발표한 GoogleNet에 적용된 모듈로 하나의 Feature가 분산되어 여러 필터를 거친 뒤 각 결과를 합치는 특징을 가진다. Residual learning은 ResNet의 주요 특징이며 입력을 출력에 직접 더해줌으로써 기존 정보를 높게 반영한다. Model 1과 Model 2는 Convolution 외 추가 기법(e.g., Inception module, Residual learning)이 학습에 미치는 영향을 파악하기 위한 목적으로 설계되었다. Model 3과 Model 4는 LeNet과 같이 단순한 모델로 설계한다.

모델에 대한 정확도만을 성능으로 보는 것은 모델 성능에 대한 잘못된 해석을 야기하므로 정확도, 학습 시간, 추론 시간, 그리고 손실값을 함께 고려해야 한다[5]. 따라서 전반적으로 높은 성능을 보일 수 있는 최적의 모델을 위해 여러 모델을 설계하였다. 아래는 유의미한 결과를 보인 각 모델의 특징과 모델 별 결과에 대한 설명이다.

▸ Model 1

Model 1은 여러 필터를 통과함으로써 다양한 Feature에 대해 학습하고 기존 정보를 다 잃지 않고 학습하기 위한 목적으로 Inception module과 Residual network에 기반하여 설계한 모델이다. Table. 3에서 확인할 수 있듯이 전체 모델 중 가장 높은 정확도를 보이나, 학습 시간은 1674s로 다른 모델에 비해 낮은 성능을 나타낸다. 많은 레이어 수로 인해 다른 모델에 비해 복잡함이 증가하였고, 분산된 Feature와 입력 정보를 다시 더하는 추가적인 계산 과정이 필요하기에 발생한 것으로 추론할 수 있다. 그러므로 본 연구에 불필요한 레이어는 줄이고 핵심 레이어 및 모듈만을 선정할 필요성이 있다.

▸ Model 2

Model 2는 Inception module만을 추가하여 설계한 모델이다. Residual learning만을 적용한 결과는 2.1과 2.3의 ResNet-34와 ResNet-50으로 확인가능하므로 Inception module의 성능 확인을 위한 실험을 진행한다. Table. 3에서 확인할 수 있듯이 정확도는 약 89% 이고 학습 시간이 Model 1에 비해 $374s$ 줄어든 $1300s$을 보였다. 두 기법을 모두 추가한 결과보다 각 기법을 개별적으로 추가한 모델의 결과가 학습 시간 면에서 준수한 성능을 나타낸다. 그럼에도, Table 2의 LeNet에 비해 높은 학습 시간을 보이므로 경량화된 모델에 대한 학습을 추가적으로 진행하였다.

Table 2: Pre-trained 모델 별 학습결과

| Model | Accuracy ↑ | Training Time(sec) ↓ | Inference Time(sec) ↓ |
|---|---|---|---|
| LeNet(base) | 0.87 | **106** | **0.00003** |
| ResNet-34 | **0.89** | 1067 | 0.0002 |
| DenseNet | **0.89** | 3352 | 0.0005 |
| EfficientNet | **0.89** | 5942 | 0.0004 |
| MobileNet | **0.89** | 1099 | 0.0008 |

Table 3: 설계 모델 후보군 별 학습결과

| Model | Accuracy ↑ | Training Time(sec) ↓ | Inference Time(sec) ↓ |
|---|---|---|---|
| Model 1 | **0.90** | 1674 | 0.0002 |
| Model 2 | 0.89 | 1300 | 0.0002 |
| Model 3 | 0.89 | **360** | **0.00003** |
| Model 4 | 0.89 | 422 | 0.0001 |

▸ Model 3

Model 3은 앞선 두 모델과 달리 추가적인 기법을 사용하지 않고 LeNet과 같이 단순한 형태를 가진 모델이다. 본 모델은 드롭아웃과 배치 정규화와 같은 정규화 기법을 추가하였으며 Convolution 레이어를 세 개로 구성한다. Table 3에서 확인할 수 있듯이, 정확도에 대한 절충없이도 약 89%의 높은 정확도와 $360s$의 낮은 학습 시간으로 뛰어난 성능을 보였으며 추론 시간 또한 $0s$ 대로 가장 적은 시간이 걸렸다. 현재까지 소개한 모델 중 가장 본 연구에 적합한 모델이며, Balanced 데이터셋이 레이어를 여러 층 쌓기에 충분히 복잡하지 않다는 것을 알 수 있다.

▸ Model 4

학습에 사용되는 EMNIST 데이터셋은 단순하므로 Feature가 제한적일 것이기에 레이어를 최소한으로 줄여 Model 4에 대한 실험을 진행하였다. 본 모델은 이전 뉴런과의 전체적인 연관성을 충분히 파악하도록 출력 레이어 전 Dense layer의 수를 네 배 증가하여 학습을 진행한다. 결과는 감소한 레이어보다 Dense layer의 증가가 크기 때문에 Model 3에 비해 학습 시간이 증가한 것을 확인할 수 있다. 최종적으로 모델의 레이어 수 및 구성에 대한 모델 구조는 Model 3을 기반으로 설계하며 모델 구조를 고정한 후 하이퍼파라미터에 대한 최적화를 진행한다.

### 3.2 Proposed Model Structure for Classification of EMNIST

본 연구는 3.1에서 결정된 Model 3을 기반으로 최종 모델을 설계하고 최적화를 진행하여 최적의 모델을 설계하는 것을 목표로 한다. Fig. 3은 최종 모델의 구조 및 레이어를 Convolution을 중심으로 나타낸 그림이다. 입력 크기를 비롯한 중간 레이어의 Feature map 사이즈는 그림

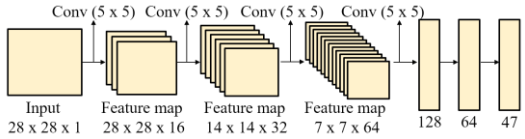


Fig. 3: 최종 모델 Architecture

Table 4: 최종 모델에 대한 하이퍼파라미터 설정 값

| Hyperparameter | Value or Type |
|---|---|
| Batch Size | 32 |
| LR Scheduling | Performance |
| Intial learning rate | 0.0001 |
| Min learning rate | 0.00001 |
| Dropout | 0.5 |
| Activation Function | ELU |
| Pooling type | AveragePooling |
| Optimizer Function | Adam |

에서 확인가능하다. 그림에는 Convolution 레이어와 Dense 레이어만을 명시하였고, 배치 정규화 레이어나 풀링 레이어 를 포함한 나머지 레이어는 나타내지 않았다. 또한 레이어를 전부 나타내지 않고 간소화하여 나타내었다. 최종 모델 구조의 전체 레이어에 대한 정보(e.g., Feature, Kernel size, Activation function)는 Appendix A에서 확인가능하다.

## IV. Model Optimization

모델 최적화를 통한 최종 하이퍼파라미터는 Table 4를 통해 확인할 수 있다. 모델의 정확도는 약 89% 이고 학습 시간은 477$s$을 보였으며 아래의 세 가지 변경으로 나누어 각 종류에 따라 비교 최적화를 진행한 결과이다.

- 비율 변경: 학습률, 드롭 아웃
- 크기 변경: 배치 사이즈
- 종류 변경: 풀링 유형, 최적화 함수, 학습률 스케줄링, 활성화 함수

### 4.1 Learning Rate and Dropout

학습률을 0.0005로 변경한 경우 정확도 약 89%와 학습 시간 215$s$를 보인다. Fig. 4는 학습률에 따른 학습 곡선 결과 그래프를 나타내며 좌측 그래프는 0.0005 로 학습률을 설정한 경우에 대한 결과를 의미한다. 해당 그래프는 Performance Scheduling의 설정을 초기 학습률 0.0005과 최소 학습률 0.0001으로 학습하는 경우 검증 데이터에 대한 성능이 학습 데이터에 대한 결과보다 높음으로써 비정상적인 학습이 진행되었음을 알 수 있다. 이러한 불안정한 학습은 제안하는 모델의 구조가 단순함에도 불구하고 초기 학습률이 상대적으로 커서 발생한 문제라고 판단하였다. 따라서 초기 학습률과 최소 학습률에 대해 0.0001 , 0.00001 로 변경하였다. Fig. 4 (b)에서 보여주는 경향성에 따라 학습되었고, 결과적으로 학습률의 수정으로 학습 곡선에 대한 오류를 수정하였다.

드롭 아웃의 경우 CNN에서 0.5 일 때 가장 최적의 성능을 나타내므로 본 실험에서는 드롭아웃 비율을 고정하여 사용한다[6].

### 4.2 Batch Size

배치 사이즈와 학습률에 따른 영향을 확인하기 위해 최적의 조합을 찾기 위한 실험을 진행하였다. 배치 사이즈를

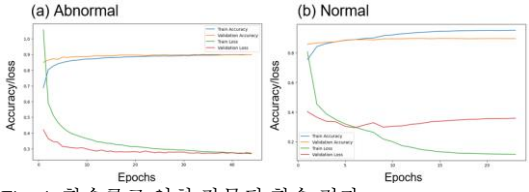


Fig. 4: 학습률로 인한 잘못된 학습 결과

64로 변경한 경우 정확도 약 87%와 학습 시간 401$s$를 보였다. 배치 사이즈를 32 로 진행한 모델의 결과와 비교하는 경우 학습시간은 줄어들었으나 정확도는 감소하였으므로 배치 사이즈를 32로 고정한다.

### 4.3 Remain

풀링 유형의 경우 Max Pooling로 변경하여 학습한 모델의 결과에서 학습 시간은 단축되었지만 학습률이 감소하기 때문에 Average Pooling을 사용하여 모델을 구성한다.

최적화 함수는 SGD와 Adam을 고려한다. SGD의 결과는 정확도 88.3% 와 672$s$ 의 학습 시간을 보였다. Adam의 경우 정확도 88.6%와 215$s$의 학습 시간이 나오며 SGD에 비해 학습 시간 면에서 높은 성능 보였으며 정확도 또한 미세한 상승이 있었기에 Adam을 본 모델의 최적화 함수로 고려한다.

추가적으로 동일한 모델에 대해 데이터 증강 기법을 사용한 데이터셋으로 진행하였고, 그 결과 정확도가 89.6%로 상승함을 확인할 수 있다. 또한 학습 곡선에서 후반 Epoch에도 학습 정확도와 검증 정확도의 차이가 증가하지 않고 검증 손실 그래프가 지속적으로 감소하는 모습을 확인할 수 있다. 따라서 데이터 증강 기법과 학습률 스케줄링을 통해 학습 안정성을 높이고 과적합 문제를 해결한다.

마지막으로 활성화 함수에 대한 최적화를 진행한다. 활성화 함수를 ReLU로 변경하였을 때 학습 시간은 줄어들지만 정확도가 미세하게 감소하여 88.3%를 보였고 데이터 증강 기법 적용 시 정확도 89.5% 임을 확인하였다. 추가적으로 기본 활성화 함수인 Tanh 함수로 학습을 진행하였을 때 88% 의 정확도와 286$s$ 의 학습 시간을 보였다. 따라서 세 가지 활성화 함수 중 학습 시간이 양호하고 정확도 면에서 높은 성능을 보인 ELU를 활성화 함수로 결정하였다.

### 4.4 Problem situation during model optimization process

모델의 최적화 과정에서 특정 케이스에 대한 부적절한 결과가 도출되었고 이에 대한 문제 상황을 소개한다. 모델 선정 후 기준이 되는 하이퍼파라미터에 대한 학습을 진행하며 검증 데이터에 대한 정확도와 손실 그래프가 학습 데이터에 대한 정확도와 손실 그래프에 비해 더 좋은 결과를 보이는 문제를 확인할 수 있다. 이에 대한 원인으로 높은 학습률로 인해 모델이 학습 데이터에 대한 최적의 가중치를 찾지 못하여 불안정한 학습이 이루어졌다고 판단한 후, 3.3에서 진행한 하이퍼파라미터 최적화 과정에서 Performance Scheduling의 값을 조절함으로써 해결할 수 있다.

## V. Model Performance

4 절에서 도출해 낸 하이퍼파라미터의 최적화 결과에 따라 최종적인 모델에 대한 학습을 진행하였고, 이에 대한 결과는 Fig. 5 의 학습곡선으로 확인할 수 있다. 그래프에서

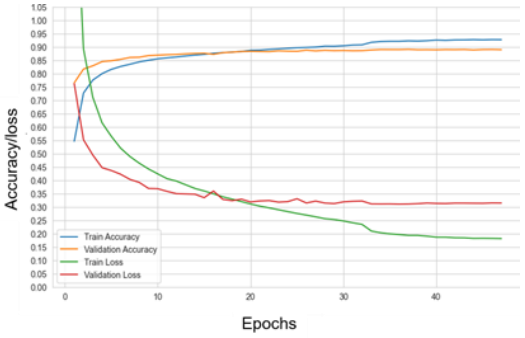


Fig. 5: 최종 모델에 대한 학습 곡선

Table 5: 최종 모델의 학습 결과

| Accuracy | Training Time(sec) | Inference Time(sec) |
|---|---|---|
| 0.89 | 477 | 0.00003 |

파란색 실선은 학습 정확도를 의미하고 주황색 실선은 검증 정확도를 의미한다. 추가적으로 초록색 실선은 학습 손실 값에 대한 학습 곡선이며 빨간색 실선은 검증 손실 값에 대한 학습 곡선이다. 학습 정확도와 검증 정확도는 큰 폭의 차이를 보이지 않으면서 학습이 진행된다. 추가적으로 손실 값에 대한 그래프가 과적합이 이루어지지 않았음을 확인하면서 본 모델에 대한 최종적인 학습 결과의 성능을 확인할 수 있다. 정량적인 수치는 Table 5 에서 확인가능하다. 결론적으로 정확도의 경우 89%를 보이며 학습 시간의 경우 477s, 추론 시간의 경우 0.00003s를 나타내었음을 확인할 수 있다.

최종 모델의 경우 데이터 증강 기법의 사용 없이도 준수한 성능과 학습 곡선을 보여주었다. 그럼에도 앞선 모델에 대해 진행한 데이터 증강 기법을 추가로 적용하여 결과를 분석하려 한다.

## VI. Additional Process

데이터 증강 기법에 사용할 객체에 대한 파라미터는 Table 6 과 같이 설정한다. 최종적으로 EMNIST Balanced 데이터셋에 변화를 주는 데 사용할 파라미터는 여러 파라미터 중 Rotation, Width shift 그리고 Height shift 만을 사용한다. EMNIST 데이터셋의 경우 채널 수가 한 개이고 사이즈도 $28 \times 28$로 ImageNet 데이터셋에 비해 작으므로 Keras 에서 제공하는 ImageDataGenerator 의 여러 파라미터를 적용한다면 단일 이미지에 가해 지는 손상이 급격하게 커질 수 있다고 판단했다. 이미 Balanced 데이터셋에는 학습하기에 부적절한 데이터가 많이 포함되어 있고 그에 더해져 데이터 증강 기법에 의해 손상된 데이터를 추가한다면 학습 성능에 직접적인 영향을 미친다고 생각하였다. 따라서 적용할 파라미터를 최소한으로 줄이되 의미 있는 파라미터만을 사용하여 데이터 증강을 진행하였다. 최종적으로 선정한 파라미터

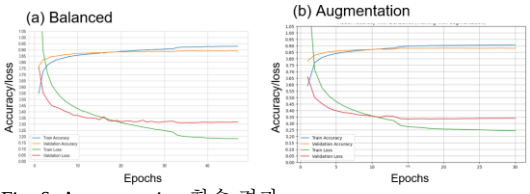


Fig. 6: Augmentation 학습 결과

Table 6: 데이터 증강에 사용된 객체 별 파라미터

| Generator | Rotation | Width shift | Height shift |
|---|---|---|---|
| A | 20 | 0.2 | 0.2 |
| B | 30 | 0.1 | 0.1 |
| C | 45 | 0.05 | 0.05 |

는 Table 6 에서 보여지듯이 Rotation, Width shift 그리고 Height shift 이고 각 개별적인 값을 다르게 하여 총 세 가지 ImageDataGenerator 를 구성하였다. 따라서 데이터의 수는 데이터 증강을 적용하지 않은 Balanced 데이터셋에 ImageDataGenerator 를 세 번 적용한 데이터를 더하여 약 네 배의 데이터를 사용할 수 있다. Bymerge 데이터셋에 비해 여전히 데이터의 수가 적지만 데이터 증강 기법이 진행된 데이터셋의 경우 각 클래스 별 데이터의 수가 균일하다는 장점이 존재한다. 학습은 제안하는 모델을 사용하고 Fig. 6 으로 결과를 확인할 수 있다. Fig. 6 (a)는 기존 Balanced 데이터셋만을 이용한 학습 결과이고 Fig. 6 (b)는 증강된 데이터셋을 이용한 학습 결과이다. Fig. 6 (b)의 학습 곡선에서 학습 데이터와 검증 데이터에 대한 정확도와 손실값의 간극이 줄었으나 89% 의 정확도를 유지하지만 데이터 수가 많아짐에 따라 학습 시간이 2399$s$ 로 늘어났다. 최종 모델의 학습 결과에 대한 성능에서 정량적인 개선은 미미하더라도 학습 곡선에서 과적합이 덜 발생했음을 알 수 있다.

## VII. Conclusion

본 연구에서의 수행 과정을 통해 적절한 데이터셋의 선정, 기존 CNN 모델에 대한 분석, CNN 모델의 설계, 학습 및 최적화 과정을 진행하며 인공신경망을 이용한 학습의 전체적인 진행에 대해 탐구하였다. 먼저 데이터셋의 선정 과정에서 분포에 따른 학습 안정성 예측, 모델 설계 시 반영해야 하는 데이터의 특성 등을 고려하고 전처리 과정, 학습/검증/테스트로 나뉘는 데이터 비율의 중요성에 대해 이해할 수 있었다. Baseline 모델과 Pre-trained 모델의 실행 및 분석 과정에서 주어진 데이터셋의 특성에 따른 적절한 모델의 구조, 기법과 복잡도를 확인하였고 모델에 대해 이해할 수 있었다. 마지막으로 학습 및 최적화 과정에서 전체적인 인공신경망의 학습 시 우선적으로 고려되는 하이퍼파라미터의 특징과, 과적합이 발생한 경우 해결방법 등에 대한 이론적 공부의 실습을 진행할 수 있었다.

# APPENDIX A

# PROPOSED MODEL ARCHITECTURE

| Layer | Feature | Kernel size | Stride | Activation function |
|---|---|---|---|---|
| Conv2D (Input) | 16 | 5 × 5 | 1 | ELU |
| AveragePooling2D | 16 | 2 × 2 | 2 | – |
| Conv2D | 32 | 5 × 5 | 1 | ELU |
| BatchNormalization | | | | |
| AveragePooling2D | 32 | 2 × 2 | 2 | – |
| Conv2D | 64 | 5 × 5 | 1 | ELU |
| BatchNormalization | | | | |
| AveragePooling2D | 64 | 2 × 2 | 2 | – |
| Conv2D | 128 | 5 × 5 | 1 | ELU |
| BatchNormalization | | | | |
| AveragePooling2D | 128 | 1 × 1 | 1 | – |
| Flatten | | | | |
| FC | 64 | – | – | ELU |
| Dropout ( 0.5 ) | | | | |
| FC | 47 | – | – | Softmax |

## REFERENCES


[1] J. Y. Song, J. W. Si, et al., "Analysis and Comparison of Classification Performance on Handwritten Datasets using ResNet-50 Model," KSCI, 2023.

[2] G. Cohen, S. Afshar, et al., "EMNIST: Extending MNIST to handwritten letters", IJCNN, 2017.

[3] K. He and X. Zhang, et al., "Deep Residual Learning for Image Recognition," CVPR, 2015.

[4] Y. Lecun, L. Bottou, et al., "Gradient-Based Learning Applied to Document Recognition," IEEE, 1998.

[5] K. Choi, D. Cho, et.al, "Time and Accuracy Trade-Off analysis and Minimum learning amount for Deep Neural Network Quantization Learning," KIISE, 2020.

[6] N. Srivastava, G. Hinton, et al., "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", JMLR, 2014.