딥러닝을 활용한 필기체 인식 프로젝트

권병근[1,*], 성다솜[2]

[1]부산대학교 수학과, 산업수학 소프트웨어 연계전공, 산업수학센터 학부연구생
[2]부산대학교 수학과, 빅데이터 연계전공, 산업수학센터 학부연구생
이메일 : *house9895@pusan.ac.kr

1. Abstract

이 프로젝트에서는 손으로 쓴 알파벳 데이터셋을 사용하여 손글씨 인식을 위한 CNN 모델을 개발하였다. 데이터를 훈련 및 테스트 세트로 나누었고, 스케일링을 적용하여 데이터 정규화를 진행하였다. 데이터의 차원을 맞추기 위해 reshape 작업을 수행하였고, 이후에 입력받는 값이 정확하고 이쁘게 쓰이지 않은 입력값도 인식해서 예측을 할 수 있게 회전하고 가로, 세로로 이동하고, 전단 변형과 확대, 축도 등을 진행하면서 데이터를 증강한 뒤 학습을 진행하여 더 정교한 모델을 만들었다. CNN 모델을 구축한 뒤, 합성곱 층과 풀링 층을 번갈아 가며 추가하였다. 배치 정규화를 적용하여 모델의 안정성을 향상했고 Dropout을 적용하여 과적합을 방지하였다. 모델의 하이퍼파라미터를 튜닝하고, Adam 옵티마이저를 사용하여 모델을 컴파일하였다. 이 연구에서 개발한 손글씨 인식 모델은 알파벳 데이터셋에 대해 높은 성능을 보였으며, 데이터 증강과 배치 정규화를 통해 모델의 성능과 안장성을 향상했다. 이 모델은 실제 활용할 수 있는 손글씨 인식 시스템의 기반이 될 수 있을 것이다.

2. Introduction

머신러닝을 공부하면서 교수님의 추천으로 MNIST 숫자 손글씨 데이터를 학습하여 분류하는 모델을 만들기 시작했다. 숫자 데이터에서 더 나아가 영어 손글씨 데이터를 예측하는 모델을 학습하는 것에 흥미와 도전 의식을 느껴 프로젝트 주제로 선택하게 되었다.

손글씨를 스캔하거나 사진으로 찍으면 보통 이미지 형태로 저장되어 수정하기 어려운 경우가 많다. 그러나 손글씨를 바로 텍스트로 변환해주면 필요할 때마다 문서 작업으로도 쉽게 수정할 수 있는 장점이 있다.

영어 손글씨 데이터를 CNN 모델을 기반으로 학습하여, 실제 손글씨 데이터를 입력하면 출력해 주는 모델을 개발하고, 모델을 배포하기 위하여 웹사이트를 구현하였다.

3. Main

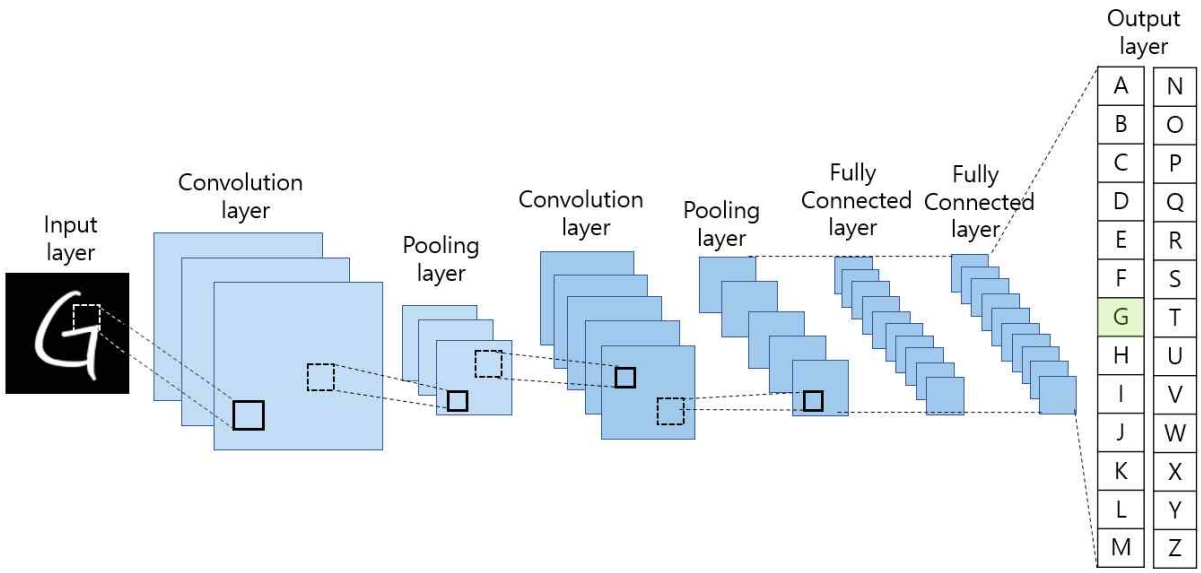
(1) 사용한 데이터

kaggle 사이트 내에서 A-Z Handwritten Alphabets in .csv format을 사용하였고 이는 이미지 파일을 CSV 파일로 전처리한 데이터이다.

(2) CNN 이란?

데이비드 허블과 토르스텐 비셀이 시각피질 구조 연구(뇌가 이미지를 인식하는 방법을 찾는 연구)를 통해 우리의 뉴런(신경)들이 시야의 일부 범위 안에 있는 시각 자극에만 반응한다는 사실을 보였다. 즉, 뉴런이 시야의 몇몇 부분에 반응하고 이 부분들이 합쳐서 전체 시야를 감싼다는 것이다. 이 연구가 CNN, 합성곱 신경망으로 점진적으로 진화되었다.

- CNN 모델의 구조



<Input Layer – Convolution layer – Pooling layer – Fully Connected Layer –

Output layer>
* Input layer : 이미지를 입력
* Convolutional layer : 필터를 통해 이미지에서 특성(feature)을 추출
* Pooling layer : 특성 맵을 다운 샘플링하여 연산량을 감소
* Fully Connected layer : Softmax를 활성화 함수로 사용하여 다중 분류
* Outer layer : 이미지를 분류해 결과 출력

(3) 모델링 속의 수학

1) Min-Max Scaling & Isomorphic

Min-Max Scaling은 입력 데이터의 범위를 0과 1 사이로 조정하는 방법이다. 이 스케일링을 수행하면 입력 데이터의 최솟값은 0, 최댓값은 1이 되도록 조정된다. 이러한 스케일링은 입력 데이터의 모든 값이 일정한 비율로 조정되므로 데이터 간의 상대적인 크기가 보존된다.

Isomorphic은 두 개의 대상 A와 B가 있을 때, A와 B 간의 매핑 f가 존재하고, 이 매핑이 일대일 대응이면서 연산을 보존하며 역함수가 존재하는 경우를 의미한다. 이는 대상 A와 B 간의 구조와 내부 연산이 서로 동일하게 보존되는 것을 의미한다.

따라서 Min-Max 스케일링은 입력 데이터에 대한 일대일 대응 매핑을 수행하고, 이에 따라 데이터 간의 상대적인 크기가 보존되므로 Isomorphic이다.

2) ReLU(Reified Linear Unit)

$$f(x) = \max(0, x)$$

입력값이 양수일 경우 그대로 출력하고, 음수일 경우 0으로 출력하는 함수이다.

비선형 함수로서 딥러닝 모델에서 비선형성을 추가할 때 많이 사용된다. 비선형 함수를 사용함으로써 모델이 다양한 패턴과 특징을 학습할 수 있게 된다. 또한 입력값이 음수일 때 기울기가 0이기 때문에 gradient vanishing 문제를 해결할 수 있고, 계산 속도가 빠르기 때문에 딥러닝 모델의 학습 속도를 향상할 수 있다.

3) Softmax

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

분류 문제에서 출력층의 활성화 값을 확률로 변환해주는 함수로, 출력층에서 다중 클래스 분류 문제에 사용되며, 모든 클래스에 대한 예측값의 합이 1이 되도록 만들어준다.

출력값이 확률로 나타낼 수 있어 해석하기 쉽고, 분류 모델에서 예측값과 정답 값 사이의 차이를 계산하는 손실함수인 cross-entropy와 함께 사용하면, 효과적인 학습이 가능하다.

4) Categorical cross-entropy loss

$$loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

다중 클래스 분류에서 사용되는 손실함수로 분류 문제에서 예측값과 실제 값 사이의 오차를 계산하는 데 사용된다.

그래디언트가 0이 아니기 때문에 모든 레이어에서 학습이 가능하고, 확률 분포를 사용하기 때문에 모델이 확률적인 예측을 하게 된다.

5) Optimizer with Adam

최적화 알고리즘(Optimizer)은 모델이 학습하는 과정에서 최적의 가중치(weight)와 편향(bias)을 찾아내는 방법을 제공한다. 이 과정에서 손실함수(loss function)를 최소화하기 위한 가중치와 편향을 찾는 것이 목적이다.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$
$$\hat{m_t} = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v_t} = \frac{v_t}{1-\beta_2^t}$$
$$\theta_t = \theta_{t-1} - \frac{\eta^2}{\sqrt{\hat{v_t} + \epsilon}} \hat{m_t}$$

w : 학습되는 가중치
g_t : 현재 그래디언트
m_t : 이전 시간 단계의 지수 가중 이동 평균
v_t : 이전 시간 단계의 제곱 그래디언트의 지수 가중 이동 평균
β_1 : 첫 번째 모멘트의 지수적 감소율
β_2 : 두 번째 모멘트의 지수적 감소율
η : 학습률
ϵ : 분모를 0으로 나누는 것을 방지하기 위한 작은 값

경사 하강법(gradient descent)의 변형 알고리즘이다. Adam은 각 가중치에 대한 적응형 학습률(adaptive learning rate)을 사용해 효율적인 학습을 가능하게 한다.

학습률을 동적으로 조절하여 학습 속도를 높이고, 과적합(overfitting)을 방지하여, 성능을 최적화하는 효과를 가져온다. 다른 최적화 알고리즘들과 비교했을 때 빠르게 수렴하고, 대부분의 경우에서 성능이 우수하다는 장점이 있다.

(4) 데이터 정보

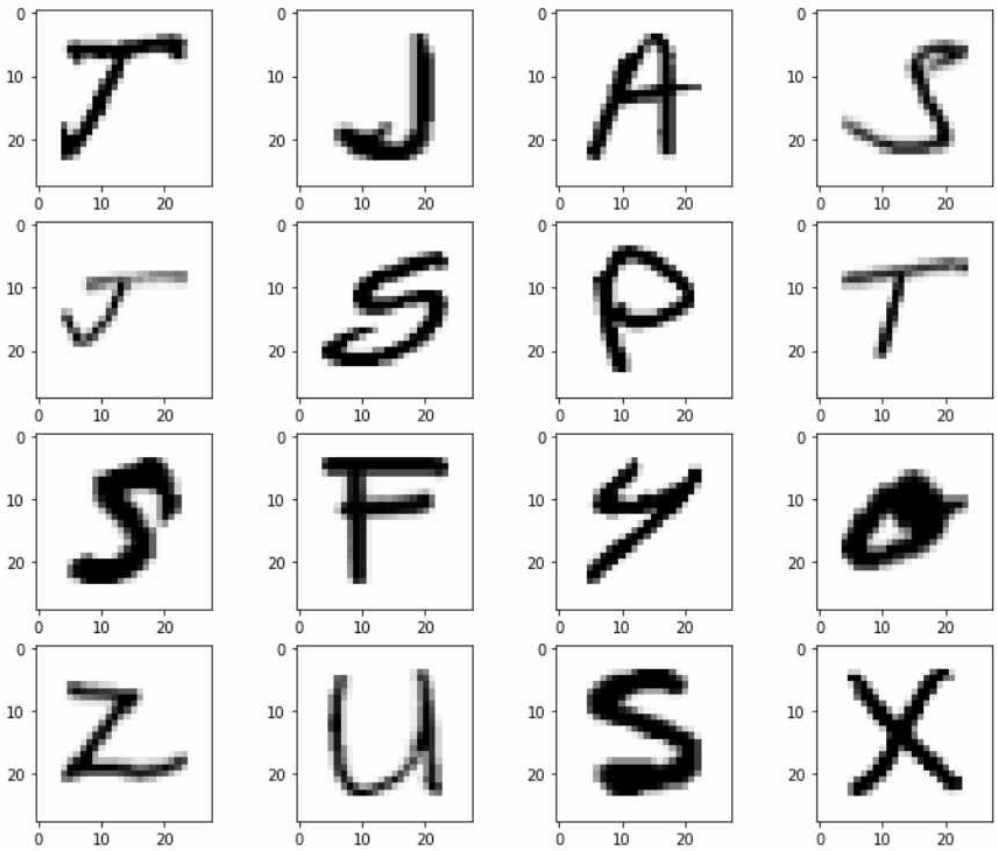
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 372450 entries, 0 to 372449
Columns: 785 entries, 0 to 0.648
dtypes: float32(785)
memory usage: 1.1 GB
```

```
     0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  ...  0.639  0.640  0.641
#
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0

   0.642  0.643  0.644  0.645  0.646  0.647  0.648
0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
1    0.0    0.0    0.0    0.0    0.0    0.0    0.0
2    0.0    0.0    0.0    0.0    0.0    0.0    0.0
3    0.0    0.0    0.0    0.0    0.0    0.0    0.0
4    0.0    0.0    0.0    0.0    0.0    0.0    0.0

[5 rows x 785 columns]
```

```
           0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  ...  0.639  0.640
#
372445  25.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0
372446  25.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0
372447  25.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0
372448  25.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0
372449  25.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0

        0.641  0.642  0.643  0.644  0.645  0.646  0.647  0.648
372445    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
372446    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
372447    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
372448    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
372449    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
```

데이터는 총 372450 rows x 785 columns이며, 각 행이 하나의 문자를 나타내는 것이다. 각 행의 첫 번째 열은 0~25로 labeling이 되어있으며 각각의 숫자는 A~Z를 의미한다. 나머지 784개의 열은 이미지를 수치화한 것이다.

(5) 데이터 시각화



무작위로 16개를 추출하여 시각화해본 결과이다. 각 데이터가 알파벳을 잘 나타내주고 있다.

(6) 데이터 스케일링

```
# 데이터 스케일링
X_train, X_test, y_train, y_test = train_test_split(X, y)
scaler = MinMaxScaler()
train_scaled = scaler.fit_transform(X_train)
test_scaled = scaler.transform(X_test)
train_scaled[0][:10]
```

X, y 데이터를 train set과 test set으로 분리하고, MinMaxScaler()를 사용하여 데이터를 스케일링하였다. 스케일링한 train set과 test set을 각각 train_scaled, test_scaled에 저장하였다. 스케일링을 통하여 모델 학습의 성능을 개선하였다.

(7) CNN 모델링-1

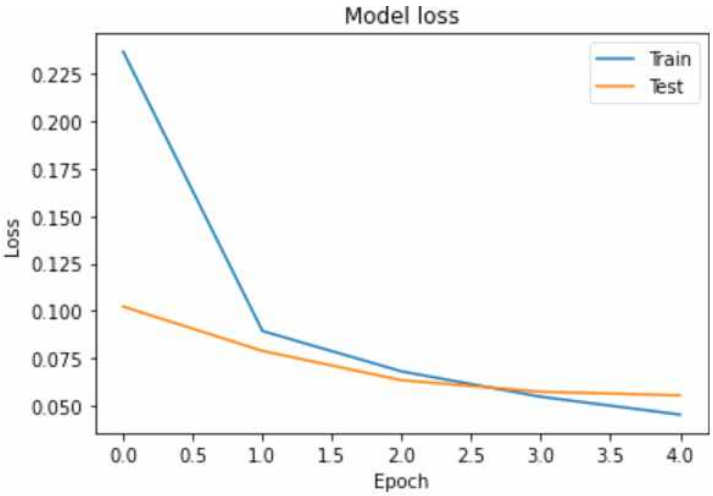
Layer (type)	Output Shape	Param #
conv2d(Conv2D)	(None, 24, 24, 32)	832

max_pooling2d(MaxPooling2D)	(None, 12, 12, 32)	0
dropout(Dropout)	(None, 12, 12, 32)	0
flatten(Flatten)	(None, 4608)	0
dense(Dense)	(None, 128)	589952
dense_1(Dense)	(None, 26)	3354

Total params : 594,138

Trainable params : 594,138

Non-trainable parmas : 0



모델 학습 결과 loss : 0.2390, accuracy : 0.9337의 성능을 보여주었다.

실제로 손글씨를 입력하여 테스트해본 결과 높은 정확도를 보여주며 예측에 성공하였다.

(8) 웹페이지 제작

모델을 만든 이후 배포를 하여 다양한 사람이 써 볼 수 있도록 웹페이지를 제작하였다. 최대 10글자까지 입력받아서 예측할 수 있다.

(9) CNN 모델링-2

1차적인 모델링 이후에 정자로 써진 알파벳들은 잘 예측해 주었지만, 정자로 쓰지 않았을 때는 예측을 잘하지 못하는 문제점이 발생하였다. 그래서 신경망의 안정성과 학습 속도를 향상하기 위해 연구를 진행하였다.

배치 정규화(Batch Normalization)는 각 미니 배치(Mini-batch) 내의 입력값을 정규화하기 위해 적용된다. 이는 평균을 0으로 조정하고 표준 편차를 1로 조정하여 모델의 안정성을 향상하고 학습 속도를 가속화 했다.

이미지 증강(Image Augmentation)은 이미지 변환 기술 중 하나로, 회전, 이동, 기울기 및 크기 조절을 적용하여 데이터 다양성을 증가시켰다. 이를 통해 모델이 다양한 각도와 위치의 데이터에 노출되므로 일반화 성능이 향상되었다.

모델 구조는 합성곱 및 풀링 레이어(Convolutional and pooling layers)를 통합함으로써 향상되었다. 더 깊은 구조가 설계되어 데이터 내의 더 복잡한 특징과 정교한 패턴을 포착할 수 있게 되었다.

과적합을 방지하기 위해 드랍아웃 레이어(Dropout layers)가 무작위로 뉴런을 비활성화시켰으며, 이는 일반화된 패턴의 학습을 촉진했다. 합성곱 및 풀링 레이어의 출력은 1D 벡터로 변환되어 추출된 특징을 기반으로 다중 클래스 분류가 가능해졌다.

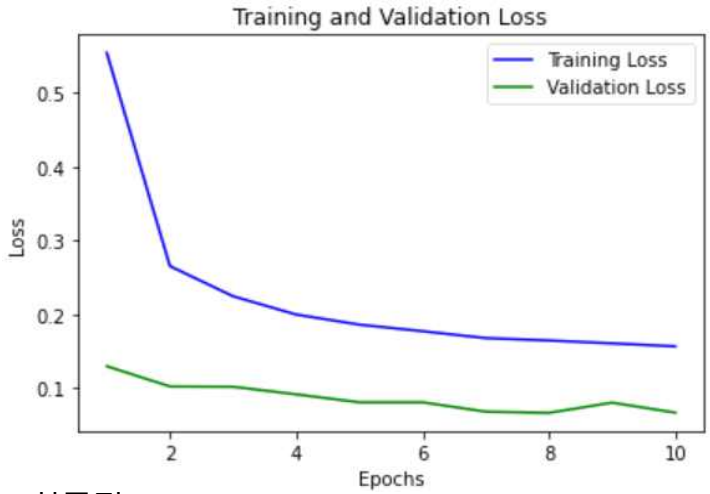
학습률(Learning rate), 배치 크기(Batch size) 및 드롭아웃 비율(Dropout rate)과 같은 다양한 하이퍼파라미터(Hyperparameters)는 모델 수정 중에 세밀하게 조정되었다. 모델은 범주형 크로스엔트로피 손실함수(Cross-entropy loss function)와 아담 옵티마이저(Adam Optimizer)를 사용하여 성능을 최적화하도록 컴파일했다.

Layer(type)	Output Shape	Params #
conv2d_14(Conv2D)	(None, 24, 24, 32)	832
batch_normalization_4(BatchNormalization)	(None, 24, 24, 32)	128
max_pooling2d_13(MaxPooling2D)	(None, 12, 12, 32)	0
conv2d_15(Conv2D)	(None, 10, 10, 32)	9248
max_pooling2d_14(MaxPooling2D)	(None, 5, 5, 32)	0
conv2d_16(Conv2D)	(None, 3, 3, 32)	9248
batch_normalization_5(BatchNormalization)	(None, 3, 3, 32)	128
max_pooling2d_15(Maxpooling2D)	(None, 1, 1, 32)	0
dropout_4(Dropout)	(None, 1, 1, 32)	0
flatten_4(Flatten)	(None, 32)	0
dense_8(Dense)	(None, 128)	4224
dense_9(Dense)	(None, 26)	3354

Total params : 27,162

Trainable params : 27,034

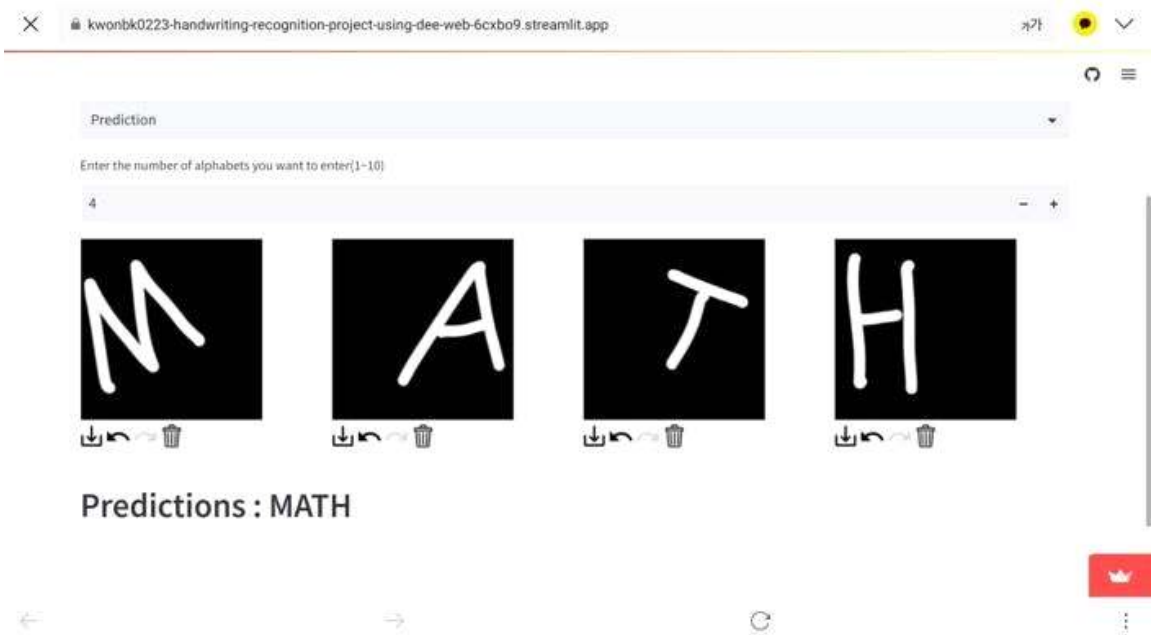
Non-trainable paramas : 128



최종적으로 loss : 0.0661, accuracy : 0.9822로 accuracy는 5%, loss는 17%의 성능 향상이 있었다.

	Loss	Accuracy
CNN Modeling-1	0.2390	0.9337
CNN Modeling-2 Add convolutional and pooling layers Add batch normalization Use Hyperparameter tuning Image Augmentation	0.0661	0.9822

(10) 웹페이지 개선



CNN 모델링-2의 결과를 바탕으로 웹페이지를 개선하였다. 이전과 페이지와 달리 글자를 정자로 쓰지 않고 기울여 쓰거나 좌·우로 치우쳐 쓰더라도 잘 인식하게 됐다.

4. Summary & Conclusion

딥러닝과 이미지 처리 기술의 발전을 활용하여 필기 인식을 위한 CNN 모델의 개발을 하였다. 합성곱 및 풀링 레이어를 생성하고 배치 정규화, 드롭아웃, 하이퍼파라미터 튜닝 및 이미지 증강을 구현함으로써, 모델의 성능과 손글씨를 정확하게 인식하고 처리하는 능력을 향상했다.

모델 성능 개선을 통해 이전 버전의 모델보다 뛰어난 정확도를 보였으며 손실을 크게 줄였다. 합성곱 및 풀링 레이어의 포함은 배치 정규화와 이미지 증강과 같은 기술과 함께 모델의 능력을 향상했다. 이를 통해 다양한 각도나 구석에서 작성된 손글씨를 인식할 수 있는 능력이 강화되었다. 이는 이전 모델이 깔끔하게 쓰인 손글씨에 한정되었던 제약을 뛰어넘게 되었다.

웹페이지의 구현은 개선된 모델을 실시간으로 보여주었다. 이는 다양한 상황의 손글씨를 성공적으로 인식하며, 우편 번호 인식, 차량 번호판 인식 및 금융 문서 처리와 같은 실제 응용 분야의 잠재력을 보여주었다. 모델에 의한 자동화와 효율성은 다양한 분야에서 프로세스를 최적화하고 생산성을 높이는 데 기여한다.

이 모델은 영어 알파벳 손글씨 데이터에 초점을 두었지만, 모델 구조와 기술에서 이루어낸 발전은 다른 언어와 문자에도 적용할 수 있으며, 손글씨 인식 기술의 범위를 확장할 수 있다.

이 연구는 CNN 모델과 딥러닝 기술이 필기 인식 분야에서 가진 능력과 잠재력을 보여주었고, 결과적으로 이 분야의 미래 발전을 위한 강력한 기반을 마련한다면, 더 정확하고 효율적인 다재다능한 손글씨 인식 시스템을 구축하여 다양한 도메인에서 자동화와 생산성을 촉진 시킬 것이다.

5. Utilization Plan & Expected Effect

머신 러닝을 활용하여 손글씨를 인식하여 텍스트로 변환하는 기술은 기업이나 공공기관에서의 문서, 서류 처리 업무에서 매우 유용하게 사용될 수 있다. 스캔된 문서를 자동으로 인식하여 텍스트로 변환하면, 작업 처리 속도와 정확도가 향

상된다. 특히 병원 등에서 간호사들이 수기로 작성하는 중요한 숫자나 문자들이 문서로 디지털화 되어 의사들에게 바로 전송되면 서로의 오해로 잘못 인지하여 환자가 위험에 처하는 상황을 막을 수 있을 것이다. 또한 대규모 손글씨 데이터를 분석하면 빅데이터 분석에 활용할 수 있다. 이를 통해 다양한 분야에서 데이터 분석 및 검색이 쉬워지며, 더욱 높은 생산성을 달성할 수 있다.

기대효과로는 업무 효율성을 증대시킬 수 있다. 수작업으로 처리해야 했던 작업을 자동화하여 생산성을 높일 수 있다. 또한 손글씨로 작성된 문서를 텍스트로 변환하면, 데이터 분석 및 검색이 쉬워져 다양한 응용이 가능해진다. 이를 통해 다양한 분야에서 활용할 수 있으며, 협업을 통해 작업 처리 속도 및 정확도를 높일 수 있다.

6. Future Work

현재 이 연구는 대문자 알파벳을 하나씩 인식한다는 문제점이 있다. 더 나아가 소문자 알파벳, 한글, 한자 등의 다양한 언어 데이터셋을 수집하여 폭넓은 손글씨 인식 모델을 만들 것이다. 또한 한 글자씩 인식하는 것이 아닌 단어나 문장을 입력하면 모델 스스로 글자를 분할하고 인식 후 예측을 통해 텍스트로 출력하는 웹 페이지를 만들 것이다.

7. GItHub & Web page

GitHub :
https://github.com/KwonBK0223/Handwriting_recognition_project_using_deep_learning
Web page :
https://kwonbk0223-handwriting-recognition-project-using-dee-web-6cxbo9.streamlit.app/

8. Reference

* AurélienGéron(2019), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition,O'ReillyMedia, Inc.