# 한글 손글씨 문장 인식에 대한 OCR 오픈 소스 성능 비교 분석

# Performance Analysis of OCR Open Source for Korean Handwritten Sentence


**요 약**

본 연구는 대표적인 오픈 소스 기반의 OCR 모델인 PP-OCRv3와 EasyOCR를 한글 손글씨 문장 인식에 적용해보고, 1-NED 평가 방법을 통해 각 모델의 성능을 평가했다. AI Hub의 한국어 글자체 이미지 데이터를 적용한 결과, PP-OCRv3는 1-NED 평가 방법에서 0.721점, EasyOCR은 0.671점을 달성했다. 본 연구는 기존의 글자 단위의 한글 손글씨 인식에서 문장 단위로 확장했다는 점에서 의의가 있다.


## 1. 서 론

오늘날 인간이 생성하는 데이터 양은 매우 방대하기 때문에, 데이터를 효율적으로 이해하고 분석하기 위해선 컴퓨팅 기술이 필수 불가결하다. 컴퓨터 비전을 활용한 OCR(Optical Character Recognition) 기술이 등장함에 따라, 인간의 수작업 없이 자동으로 문서, 영수증, 명함 등을 인식해 이를 전자적 형태로 변환하는 것이 가능해졌다. 이를 통해 문서 및 도서 자료 번역, 기록물 데이터베이스화, 전문 의료 정보 해석 및 공공 서비스 지원 등 다양한 분야에서 OCR 기술이 활용되고 있다.[1]


그림 1. OCR의 동작 과정

OCR의 동작 과정(그림 1)은 이미지에서 문자가 속한 영역을 검출하는 문자 검출(Text Detection) 과정과 해당 영역의 문자가 실제로 어떤 문자인지 인식하는 문자 인식(Text Recognition) 과정으로 나눌 수 있다. 또한 OCR의 성능을 높이기 위해 전처리 및 후처리 단계를 추가하기도 한다.

본 연구는 전자 문서 애플리케이션을 개발하기 위한 목적으로, 대표적인 OCR 오픈 소스인 EasyOCR과 PP-OCRv3[2]를 활용해 한글 손글씨 문장에서 단어를 검출 및 인식하여 이를 전자적 형태로 변환하고 두 오픈 소스 간 성능에 대한 데이터를 제시한다.

기존의 한글 손글씨 인식에 대한 OCR 오픈 소스에는 IBM에서 개발한 tensorflow-hangul-recognition[3]이 있는데, 이는 단일 CNN(Convolutional Neural Network)을 사용하기 때문에 글자 단위의 인식만 가능하다. 이는 대부분 문장으로 이루어진 전자 문서 애플리케이션과 적합하지 않기 때문에, 문자 검출과 문자 인식이 모두 가능한 EasyOCR과 PP-OCRv3를 활용했다. 또한 최신 OCR 오픈소스에 대해 한글 손글씨 문장 인식을 접목한 연구가 전무하기 때문에, 향후 유사한 애플리케이션 개발 시 EasyOCR과 PP-OCRv3를 적용하고자 할 때 본 연구의 실험 결과를 참고할 수 있을 것이다.

## 2. 관련 연구

### 2.1 EasyOCR

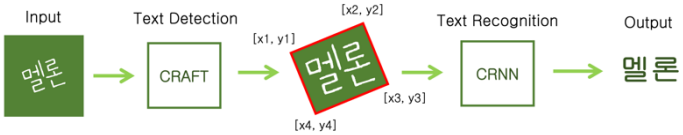


그림 2. EasyOCR 구조도

EasyOCR(그림 2)의 문자 검출 모델인 CRAFT[4]는 Batch Normalization을 사용한 VGG16 기반의 FCN(Fully Convolutional Network) 모델이며, Decoding 부분에 Skip Connection을 이용하여 낮은 레벨의 특성들을 통합한다. CRAFT는 어떤 픽셀이 글자의 중심일 확률을 의미하는 Region Score와 어떤 픽셀이 인접한 두 글자의 중심일 확률을 의미하는 Affinity Score를 이용해 인접한 개별 글자들을 상향식 방식으로 연결하여 단어 단위로 검출하기 때문에, 곡선이나 기형적 형태의 문자들 또한 잘 검출한다. 이로 인해 CRAFT는 다양한 형태의 문자들을 인식해야 하는

*: 공동 제1저자

Scene Text Detection 문제에서 주목받고 있다.

최근 문자 인식 모델들은 개별 글자가 아닌 주로 단어 단위로 인식하는데, CNN을 이용해 문자 이미지의 특성을 추출하거나 LSTM(Long Short-Term Memory) 또는 GRU(Gated Recurrent Unit)가 있는 Recurrent 모델을 사용한다.[1] EasyOCR의 문자 인식 모델인 CRNN[5]은 CNN과 RNN(Recurrent Neural Network)을 이용해 단어 단위 인식을 수행한다. CRNN은 입력 이미지로부터 Feature Sequence를 추출하는 Feature Sequence Extraction 단계와 추출한 Sequence의 각 Frame에 대한 Label Distribution을 예측하는 Sequence Labeling 단계, 이들을 최종 문자로 변환하는 Transcription 단계로 이루어진다.

### 2.2 PP-OCRv3

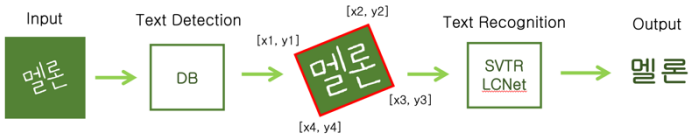


그림 3. PP-OCRv3 구조도

PP-OCRv3(그림 3)는 DB(Differentiable Binarization)[6] 기반의 문자 검출 모델을 사용한다. PP-OCRv3는 모바일 환경에서 동작하는 것을 가정하기 때문에, 모델의 경량화를 목표로 하며 이를 위해 다양한 방법을 사용했다. 딥러닝 모델은 Backbone의 크기가 대부분을 차지하는데, PP-OCRv3는 모바일 기기를 대상으로 하는 MobileNetV3[7]을 Backbone으로 사용했으며 MobileNetV3에서 많은 수를 차지하는 SE(Squeeze and Excitation) 모듈을 제거해 모델을 경량화했다. 또한 객체 탐지에 많이 사용되는 FPN(Feature Pyramid Network)와 유사한 Head를 사용했으며, 1x1 Convolution 필터를 사용해 Inner Channel을 줄임으로써 모델을 경량화했다.

PP-OCRv3는 CRNN과 같이 Feature Extraction을 위한 Visual 모델과 Text Transcription을 위한 Sequence 모델로 이루어진 많은 Scene Text Recognition 모델과는 다르게, 하나의 Visual 모델만을 사용한 SVTR[8]와 경량 CNN 기반의 PP-LCNet[9]을 결합한 SVTR-LCNet을 문자 인식 모델로 사용한다. PP-OCRv3는 SVTR 중 작은 크기의 SVTR-Tiny를 선택했는데, SVTR-Tiny는 구조적 한계로 인해 Intel의 수학 커널 라이브러리인 MKLDNN을 사용하는 CPU에서 PP-OCRv2의 문자 인식 모델인 CRNN보다 10배나 느리다는 문제점이 있다. 이를 해결하기 위해 SVTR-Tiny 속도의 대부분을 차지하는 Mix 모듈을 제거하고 PP-LCNet의 일부분으로 교체했다. 또한 Global Mix 모듈의 속도가 입력 특성의 크기와 관련 있다는 점을 발견해, 이를 Pooling Layer 다음으로 옮겨 예측 속도를 향상시켰다.

## 3. 한글 손글씨 문장에 대한 성능 비교 분석

### 3.1 데이터셋 및 실험 환경

본 실험에서는 한글 손글씨 문장 인식을 위해 한국지능정보사회진흥원 AI Hub의 “한국어 글자체 이미지” 데이터셋을 각 모델에 학습시켰다. 해당 데이터셋에는 현대 한글 11,172자를 사용한 폰트 50종의 글자체와 성별, 연령층 별로 직접 제작한 이미지 파일 구축 및 간판, 상표, 교통 표지판 등의 한글이 포함된 이미지 10만 장을 구축한 이미지들이 제공된다. 본 실험에서는 손글씨 단어 데이터 일부를 학습 및 검증 데이터로, 손글씨 문장 데이터 일부를 시험 데이터로 사용했다. 실험은 Google Colab Pro+에서 진행했으며, 운영 체제는 Ubuntu 18.04.6 LTS, GPU는 NVIDIA A100-SXM4-40GB이다.

### 3.2 성능 평가 기준

자주 사용되는 문자 인식 성능 평가 기준에는 WEM(World based Exactly Matching) 평가 방법과 1-NED(1-Normal Edit Distance) 평가 방법이 있다. WEM 평가 방법은 정답 단어와 예측 단어가 정확히 일치하는지 평가하는 방법인데, 이는 문장 단위의 평가 시 정확히 예측한 단어 수를 반영하지 못하기 때문에 본 실험에 적합하지 않다. 이로 인해, 본 실험에서는 1-NED 평가 방법을 사용했다. 1-NED 평가 방법은 두 단어 간 편집 거리를 긴 단어의 길이로 정규화한 값을 평가하는 방법이다. 1-NED는 정답과 예측 단어 간 편집 거리가 작을수록 1에 가깝다. 1-NED는 전체 단어에서 글자 단위의 점수를 평가하는 방법이기 때문에, WEM보다 본 실험에 적합하다.

### 3.3 성능 평가

| Model | Model Size(M) | 1-NED | Speed(ms) (A100 GPU) |
|---|---|---|---|
| EasyOCR (Pretrained Only) | 24.8 | 0.481 | 0.158 |
| PP-OCRv3 (Pretrained Only) | **15.6** | 0.402 | 0.097 |
| EasyOCR (Fine-tuned Recognition) | 24.8 | 0.671 | 0.139 |
| PP-OCRv3 (Fine-tuned Recognition) | **15.6** | **0.721** | **0.096** |

표 1. EasyOCR과 PP-OCRv3의 성능 비교표

표 1은 EasyOCR과 PP-OCRv3의 문자 인식 모델을 학습시키기 전(Pretrained Only)과 후(Fine-tuned Recognition)에 대한 한글 손글씨 문장 인식 성능을 나타낸다. 학습 후 EasyOCR은 동작 속도가 0.139 ms로 0.158 ms에서 1.13배 상승했고, PP-OCRv3는

0.097 ms에서 0.096 ms로 1.01배 상승했다. 또한 1-NED 평가 방법에 대해 EasyOCR이 0.481에서 0.671로 1.40배 상승했고, PP-OCRv3이 0.402에서 0.721로 1.80배 상승했다. 여기서 주목할 점은 PP-OCRv3가 EasyOCR에 비해 모델 크기가 1.59배 더 작음에도 불구하고 1.07배 더 좋은 성능을 보여주었으며 속도 또한 1.45배 더 빨랐다.

## 4. 논의

### 4.1 문자 인식 학습의 중요성

본 실험은 EasyOCR과 PP-OCRv3의 문자 검출 모델로 각 오픈 소스가 제공하는 사전 학습 모델을 사용했다. 이는 최신 문자 검출 모델들이 다양한 언어들에 대해 다음과 같이 높은 정확도를 보여주기 때문이다. EasyOCR의 CRAFT는 ICDAR13에 대해 재현율과 정밀도의 조화 평균이 95.2[4], PP-OCRv3의 문자 검출 모델은 LCSVT, RCTW-17, MTWI 2018 등의 데이터셋 모음에 대해 85.4에 이른다.[2] 그림 4, 5, 6은 각각 CRAFT와 PP-OCRv3의 문자 검출 모델을 사용해 한국어, 일본어, 영어에 대해 문자 검출을 한 결과이다.

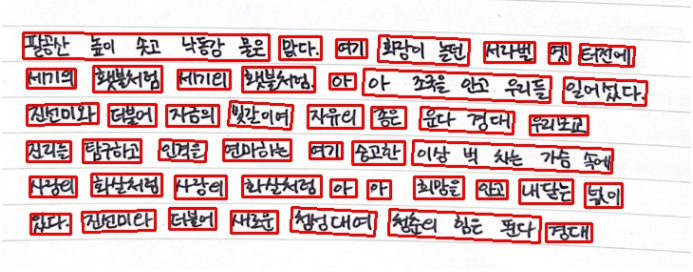

그림 4. PP-OCRv3를 이용한 한국어 문자 검출 결과

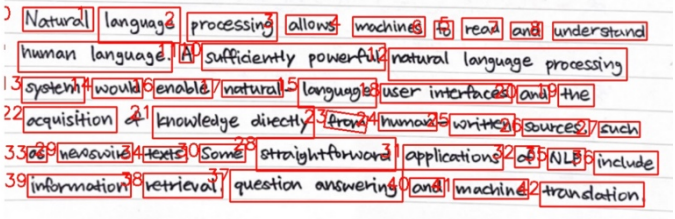


그림 5. CRAFT를 이용한 영어 문자 검출 결과

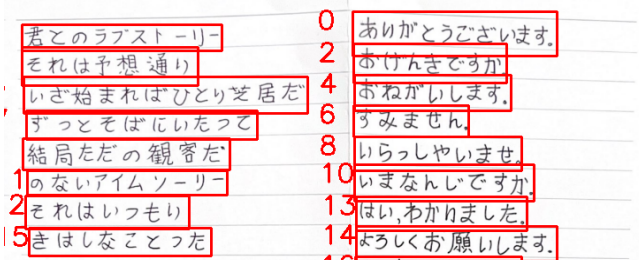


그림 6. CRAFT를 이용한 일본어 문자 검출 결과

위와 같은 이유로 문자 검출 모델에 비해, 문자 인식 모델의 성능을 높이는 것이 전체적인 성능 상승에 더 큰 영향을 줄 것이라고 판단해 본 실험에서는 문자 인식 모델의 학습에 중점을 두었다. 그러므로 성능 평가 또한 문자 인식에 대한 평가만을 진행했다.

### 4.2 평가 기준의 한계

문자 인식의 성능 평가 기준은 단어뿐만 아니라 띄어쓰기까지 평가 범위에 포함되기 때문에, 띄어쓰기를 검출하지 못하면 낮은 점수를 받게 된다. 문장은 띄어쓰기가 적지 않은 비율을 차지하며, 손글씨의 경우 사람에 따라 띄어쓰기 간격이 매우 좁거나 일정하지 않기 때문에 인식이 잘되지 않는 경우도 발생했다. 이러한 경우 모든 문자를 잘 인식했음에도 불구하고 충분한 점수를 받지 못하기 때문에, 이에 대한 성능 평가 기준의 개선이 필요하다고 판단된다.

## 5. 결론

본 연구는 최근 OCR 분야에서 많이 사용되는 EasyOCR과 PP-OCRv3에 한글 손글씨 문장 인식을 접목시켜 보고, 이에 대한 성능 비교 분석을 진행했다. PP-OCRv3는 모델 구조 상 모바일 환경에 적합할 뿐만 아니라, EasyOCR에 비해 더 좋은 성능을 보여주기 때문에 앞으로 많은 한글 손글씨 문장 인식 OCR 분야에서 적극적으로 활용될 것으로 예상된다.

## 6. 참고 문헌


[1] 민기현 외, “딥러닝 기반 광학 문자 인식 기술 동향”, 전자통신동향분석, 37권, 5호, pp. 25-30, 2022.

[2] Chenxia Li et al., “PP-OCRv3: More Attempts for the Improvement of Ultra Lightweight OCR Sytesm”, arXiv preprint, CoRR, arXiv:2206.03001v2, 2022.

[3] https://github.com/IBM/tensorflow-hangul-recognition

[4] Youngmin Baek et al., “Character Region Awareness for Text Detection”, IEEE Conf. Comput. Vis. Pattern Recognit., pㅍp. 9365-9374, 2019.

[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 Trans. Pattern Anal. Mach. Intell., Vol. 39, No. 11, pp. 2298-2304, 2016.

[6] Minghui Liao et al., “Real-Time Scene Text Detection with Differentiable Binarization”, AAAI Conf. Artif. Intel., Vol. 34, No. 7, pp. 11474-11481, 2020.

[7] Andrew Howard et al., “Searching for MobileNetV3”, IEEE Int. Conf. Comput. Vis., pp. 1314-1324, 2019.

[8] Yongkun Du et al., “SVTR: Scene Text Recognition with a Single Visual Model”, Int. Joint Conf. Artif. Intell., pp. 884-890, 2022.

[9] Cheng Cui et al., “PP-LCNet: A Lightweight CPU Convolutional Neural Network”, arXiv preprint, CoRR,

arXiv:2109.15099, 2021.

**부록**

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음 (2021-0-01082)