

OCR 성능 향상을 위한 이미지 전처리 기법 연구

정다훈¹, 최상일²

단국대학교 소프트웨어학과¹, 단국대학교 컴퓨터공학과²,

jdh251425142514@gmail.com¹, choisi@dankook.ac.kr²

A Study on Image Preprocessing Techniques for Enhancing OCR Performance

Dahun Chung¹, Sang-Il Choi²

Dept. of Software Science¹, Dept. of Computer Engineering, Dankook University²

jdh251425142514@gmail.com¹, choisi@dankook.ac.kr²

요약

본 연구는 OCR(Optical Character Recognition) 성능을 향상시키기 위한 다양한 이미지 전처리 기법을 제안하고 평가한다. 전처리 과정은 이미지의 절반 크롭, 수평 크롭, 텍스트 크롭을 포함하며, 각 단계에서의 OCR 결과를 비교 분석한다. 실험 결과, 제안된 전처리 기법이 OCR의 인식 정확도를 유의미하게 향상시킴을 확인하였다.

1. 서론

OCR(Optical Character Recognition) 기술은 이미지에서 텍스트를 자동으로 추출하는 기술로, 문서 디지털화나 자동 채점 시스템 등에 널리 활용되고 있다. 하지만 손글씨가 포함된 시험지나 설문지 같은 경우에는 인식 성능이 크게 떨어진다는 문제가 있다. 특히, EasyOCR[1][2] 같은 기존 OCR 엔진을 시험지에 직접 적용하면 40% 정도의 낮은 정확도를 보인다. 이는 배경 노이즈, 복잡한 레이아웃, 손글씨의 불규칙성 때문이다. 이런 문제를 해결하려면 OCR 적용 전에 이미지를 적절히 전처리해서 텍스트 영역을 명확하게 분리하는 것이 중요하다.

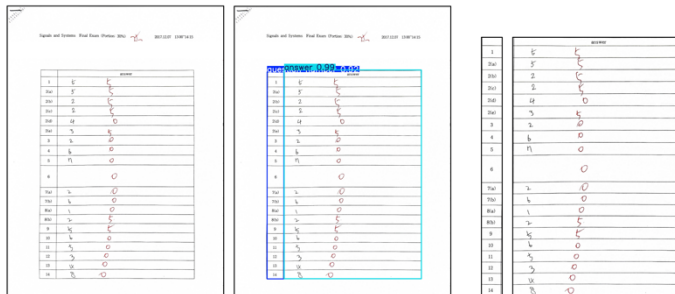
본 연구에서는 YOLOv10[3]과 OpenCV를 활용한 3단계 전처리 파이프라인을 제안한다. 절반 크롭(Half Cropped), 수평 크롭(Horizontally Cropped), 텍스트 크롭(Text Crop)을 순차적으로 적용하고, 최종적으로 MNIST[4] 기반 분류 모델로 개별 숫자를 인식한다. 실험 결과, 기존 EasyOCR의 38.33% 문제 객체 인식 정확도를 98.33%까지 향상시킬 수 있었고, 26.43%의 숫자 인식 정확도를 75.86%까지 향상시킬 수 있었다. 이는 실제 채점 시스템에 활용 가능한 수준이다.¹

해당 논문의 자세한 내용 및 코드는 아래 Github 주소를 참고하길 바란다.

https://github.com/Downy-newlearner/Preprocessing_Techniques_for_Enhancing_OCR_Performance.git

2. 제안 방법

본 연구에서는 세 가지 주요 전처리 기법을 통해 OCR 인식 성능의 향상을 도모하였다. 각 전처리 기법은 YOLOv10 모델을 기반으로 한 객체 탐지와 OpenCV를 활용한 이미지 처리 기법으로 구현되었으며, 전체 전처리 파이프라인은 원본 이미지를 절반으로 분할한 후, 수평으로 영역을 나누고 마지막으로 텍스트 영역을 추출하는 순서로 구성된다.



Original image

YOLOv10 result

Half cropped images

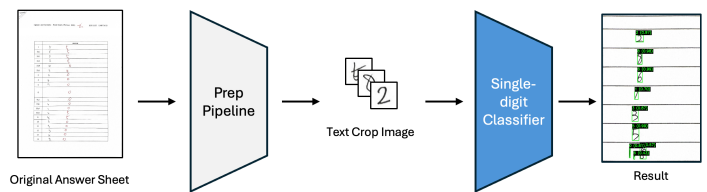
이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원·학·석사연계 ICT 핵심인재양성 지원을 받아 수행된 연구임(RS-2024-00437027)

본 연구는 2024년 과학기술정보통신부 및 정보통신기획평가원의 SW 중심대학사업 지원을 받아 수행되었음(2024-0-00035)

본 연구는 2024년도 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT 연구센터사업의 연구결과로 수행되었음(RS-2024-00437102)

[그림1]. 원본 이미지(좌)와 YOLOv10 인식 결과(가운데)와 YOLOv10 인식 결과를 기반으로 잘라낸 Half cropped images(우)

2.1. 전체 시스템 개요



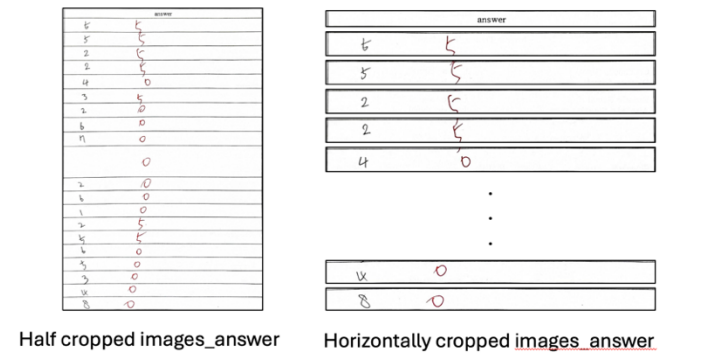
[그림2]. 전체 프로세스는 원본 이미지가 Preprocessing Pipeline과 Single-digit Classifier를 거쳐 최종 결과로 변환된다.

본 연구에서는 세 가지 전처리 기법을 순차적으로 적용하는 통합 파이프라인을 제안한다. 이 파이프라인은 YOLOv10과 OpenCV를 기반으로 구현되며, 각 단계는 다음과 같다.

2.2. 통합 전처리 파이프라인

2.2.1. 절반 크롭 (Half Cropped)

원본 시험지 이미지에는 상단에 시험 정보, 중단에 문제-답안 표, 하단에 공백이 포함되어 있다. 본 연구에서는 YOLOv10을 이용하여 "question number"와 "answer"가 위치한 영역을 탐지하고, 이 부분만을 절반 크롭 방식으로 추출하였다. 이를 통해 시험지의 텍스트 정보가 있는 주요 부분만을 남기고, 배경이나 불필요한 정보는 제거되었다. 절반 크롭은 전체 이미지 크기를 줄이면서도 텍스트 영역을 명확히 하여 OCR 처리 속도와 정확도를 동시에 향상시키는 데 기여하였다.

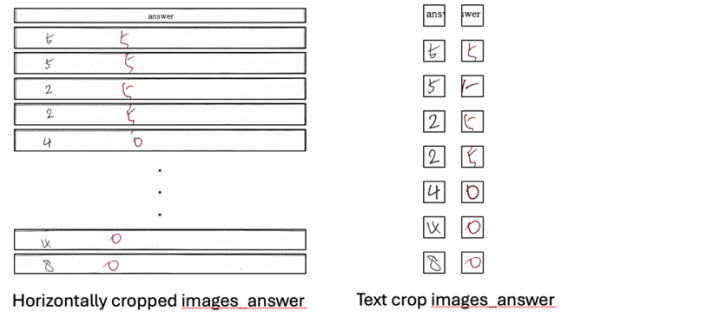


[그림3]. Half cropped images 중 answer 부분(좌)과 이를 가로선 기준으로 자른 Horizontally cropped images(우)

2.2.2. 수평 크롭 (Horizontally Cropped)

절반 크롭된 이미지에는 문제 번호와 학생 답안이 여러 행으로 구성되어 있다. 수평 크롭은 각 행을 독립적으로 분리하여 OCR이 텍스트를 보다

효과적으로 인식할 수 있도록 하기 위한 기법이다. 이미지의 수평 라인을 기준으로 각 행의 y 좌표를 탐지한 후, 행 단위로 이미지를 자르는 방식으로 구현하였다. 수평 크롭을 통해 줄 간 간섭을 최소화하고, 행별 인식 정확도를 높일 수 있었다.



[그림4]. Horizontally cropped image_answer(좌)과 text만 남겨 1:1 비율로 잘라낸 Text crop images_answer(우)

2.2.3. 텍스트 크롭 (Text Crop)

수평 크롭된 각 행에는 문제 번호와 답안이 함께 들어있다. 이걸 개별 숫자로 분리해서 인식해야 하기 때문에 텍스트 크롭을 적용했다. 먼저 Suzuki-Abe Algorithm[5]을 사용해서 각 숫자 블록을 찾아내고, 해당 영역만 잘라내서 개별 이미지로 만들었다. 문제 번호와 답안이 붙어있거나 겹쳐있는 경우에도 정확하게 분할할 수 있도록 했고, 이 과정에서 배경 노이즈도 제거되면서 인식 성능이 향상됐다. 이렇게 3단계 전처리를 순차적으로 적용하면 원본 시험지에서 개별 숫자 이미지까지 깔끔하게 분할할 수 있다. 다음에서는 이 전처리 파이프라인의 성능을 실험으로 검증한 결과를 보여준다.

2.3. 단일 숫자 인식

(Single-digit Classification)

전처리 파이프라인을 통해 분할된 개별 텍스트 이미지에는 '37' 같이 여러 숫자가 붙어있는 경우가 많다. 이걸 정확히 인식하려면 먼저 개별 숫자로 분리해야 한다.



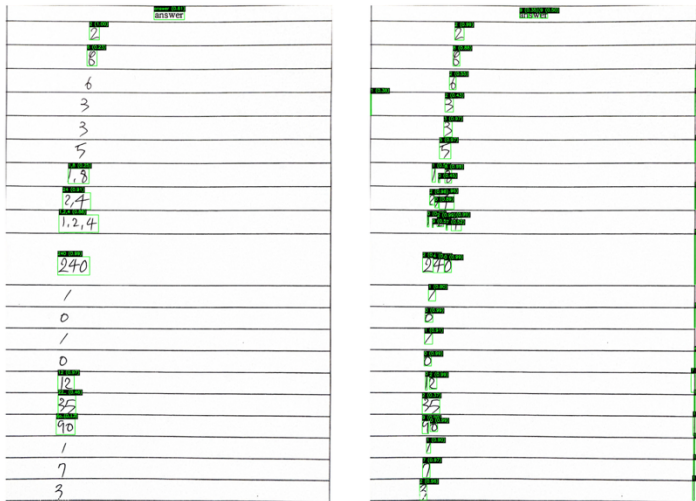
[그림5]. '37' 손글씨 이미지(좌)와 이를 한 글자씩 잘라낸 이미지(우)

각 숫자의 경계를 찾고 Bounding box를 분리한 다음, MNIST 데이터셋으로 학습된 분류 모델을 적용해서 각 숫자를 인식했다. 이 방법을 통해 손글씨 숫자를 높은 정확도로 인식할 수 있었다.

3. 실험 및 결과

3.1. 실험 설정

우리 결과의 정확도와 성능을 파악하기 위하여 표 형식의 대학 시험지 10장을 임의의 답으로 만들어 이를 대상으로 어느 정도의 인식률이 나오는지 알아보았다. 각 시험지에는 문제 번호와 학생의 손글씨 답안이 작성되어 있으며, 본 연구에서는 이를 얼마나 정확하게 인식할 수 있는지를 평가하였다. 비교 대상은 두 가지 방법으로 구성된다. 첫 번째는 기존 방법으로, 원본 이미지에 EasyOCR을 직접 적용하는 방식이다. 두 번째는 제안 방법으로, 3단계 전처리 파이프라인을 거친 후 MNIST 기반 분류 모델을 통해 인식하는 방식이다. 평가 지표로는 문제 객체 자체의 존재 여부를 판단하는 ‘문제 객체 인식’과, 인식된 객체가 어떤 숫자인지를 정확히 판단하는 ‘숫자 인식’의 두 가지를 사용하였다. 두 지표 모두 전체 시험지에서 문제 번호와 답안을 정확하게 인식한 비율을 기준으로 계산하였다. 실험 환경은 Python 3.8, OpenCV 4.8, YOLOv10, EasyOCR 1.7을 사용했고, MNIST 모델은 Hugging Face의 farleyknight/mnist-digit-classification-2022-09-04[6]를 적용했다. 예시 답안지 이미지를 사용하여 원본 이미지를 OCR에 입력했을 때와 전처리 과정과 인식 과정을 거쳤을 때를 실험한다. OCR은 오픈소스로 제공되는 EasyOCR을 사용하였다.



[그림 6]. EasyOCR로 답안지를 인식한 모습(좌). 손글씨 인식률이 떨어져 객체 인식을 실패하는 것을 확인할 수 있었다. 전처리 과정과 Single-digit Recognition을 통해 답안지를 인식한 모습(우). 모든 답안에 대해 빠짐없이 인식하며 높은 인식 성공률을 보임을 확인할 수 있었다.

3.2. 인식 성능 결과 분석

(accuracy)	문제 객체 인식	숫자 인식 실험
EasyOCR	38.33%	26.43%
ours	98.33%	75.86%

4. 기여

실험 결과, 단순히 EasyOCR만을 적용한 경우보다 본문에서 제안한 전처리 및 인식 과정을 함께 수행한 경우가 훨씬 높은 인식률을 보이는 것으로 확인되었다. 채점처럼 소수의 오차도 치명적인 작업에서는 이러한 정밀한 인식 과정이 필수적이었다. 또한, 인식 결과의 신뢰도가 낮은 경우는 대부분 Contour가 겹쳐 두 개 이상의 숫자가 중첩되어 모델에 입력된 경우로 분석되었다. 해당 연구에서 사용한 전처리 방식은 모든 답안 이미지에 대해 접근 가능하므로, 인식이 완전히 실패하거나 신뢰도가 임계치 이하인 경우에도 사용자에게 결과를 전달함으로써 결과적으로 100%의 인식을 확보가 가능할 것으로 보인다. 따라서, 채점과 같이 매우 높은 인식 정확도가 요구되는 작업에서는 본문의 과정을 따르는 것이 효과적인 해결책이 될 수 있을 것으로 기대한다.

5. 참고문헌

1. Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, Hwalsuk Lee. “CRAFT: Character-Region Awareness for Text Detection”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9365-9374, 2019.
2. Baek, Youngmin, Kim, Geewook, Lee, Junyeop, Park, Sungrae, Han, Dongyoon, Yun, Sangdoo, Lee, Hwalsuk. “Show, Attend and Read: A Simple and Strong Baseline for Irregular Text Recognition”, AAAI Conference on Artificial Intelligence, vol. 33, pp. 8732-8739, 2019.
3. Chien-Yao Wang, Alexey Bochkovskiy, Hong-Yuan Mark Liao. “YOLOv10: Real-Time End-to-End Object Detection”, arXiv preprint, arXiv:2404.03900, 2024.
4. Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner. “Gradient-Based Learning Applied to Document Recognition”, Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998. <https://ieeexplore.ieee.org/document/726791>
5. S. Suzuki, K. Abe. “Topological Structural Analysis of Digitized Binary Images by Border Following”, Computer Vision, Graphics, and Image Processing, vol. 30, no. 1, pp. 32-46, 1985. <https://www.sciencedirect.com/science/article/abs/pii/0734189X85900167>
6. Hugging Face. “MNIST Digit Classification with farleyknight/mnist-digit-classification-2022-09-04”, <https://huggingface.co/farleyknight/mnist-digit-classification-2022-09-04>