반도체디스플레이기술학회지 제20권 제3호(2021년 9월)
Journal of the Semiconductor & Display Technology, Vol. 20, No. 3. September 2021.

심층신경망을 이용한 PCB 부품의 인쇄문자 인식

조태훈[*†]
[*†]한국기술교육대학교 컴퓨터공학부

Recognition of Characters Printed on PCB Components Using Deep Neural Networks

Tai-Hoon Cho[*†]
[*†]School of Computer Science and Engineering, Korea University of Technology and Education

ABSTRACT

Recognition of characters printed or marked on the PCB components from images captured using cameras is an important task in PCB components inspection systems. Previous optical character recognition (OCR) of PCB components typically consists of two stages: character segmentation and classification of each segmented character. However, character segmentation often fails due to corrupted characters, low image contrast, etc. Thus, OCR without character segmentation is desirable and increasingly used via deep neural networks. Typical implementation based on deep neural nets without character segmentation includes convolutional neural network followed by recurrent neural network (RNN). However, one disadvantage of this approach is slow execution due to RNN layers. LPRNet is a segmentation-free character recognition network with excellent accuracy proved in license plate recognition. LPRNet uses a wide convolution instead of RNN, thus enabling fast inference. In this paper, LPRNet was adapted for recognizing characters printed on PCB components with fast execution and high accuracy. Initial training with synthetic images followed by fine-tuning on real text images yielded accurate recognition. This net can be further optimized on Intel CPU using OpenVINO tool kit. The optimized version of the network can be run in real-time faster than even GPU.

Key Words : Deep Neural Networks, OCR, LPRNet, CRNN, Text Recognition

1. 서 론

최근 부품의 초소형화, 고집적화가 급격히 진행됨에 따라, PCB 부품의 장착 및 납땜 상태를 자동 검사하는 시스템이 PCB 생산 품질 관리의 핵심 장비로서 생산 현장에서 필수적으로 사용되고 있는 추세이다. 이러한 PCB검사 장비에서 부품에 인쇄되어 있는 숫자나 문자를 인식하는 기능이 필수적이다.

일반적으로, 문서를 스캔해서 얻은 영상에서 문자나 숫자를 인식하는 것을 OCR(Optical Character Recognition)이라고 부르는데, 사무자동화에 있어 핵심이 되는 기능이다. 대표적인 오픈소스 OCR 엔진으로 Tesseract OCR[1]이 있다.

보통 문서는 흰 바탕에 검정색 문자로 보이는 데 콘트라스트(contrast)가 크고, 문자의 손상이 거의 없는 것에 비해, PCB 부품에 인쇄된 문자는 어두운 배경에 비해 밝지만 콘트라스트가 일반적으로 적고 문자의 손상 등이 많이 발생하여 인식에 더 어려움이 있다.

영상에서 문자 인식은 일반적으로 문자열 영역을 추출하는 텍스트 검출(text detection)과 검출된 텍스트를 인식하는 텍스트인식(text recognition)으로 구성된다. (최근에는

[†]E-mail: thcho@koreatech.ac.kr

FOTS[2] 같이 텍스트 검출과 텍스트인식을 동시에 수행하는 방법도 등장하고 있다.) 텍스트 검출에 기존에는 휴리스틱(heuristic)영상처리 기법을 사용하는 경우가 많았으나 최근에는 정확도를 높이기 위해 딥러닝을 이용한 텍스트검출 기법이 등장하고 있다. 대표적인 기법으로 EAST[3], Naver의 CRAFT[4] 등이 있다.

기존의 텍스트인식 방법은 흔히 입력 텍스트영역 영상으로부터 각 문자를 분리(segmentation), 추출한 후 추출된 각 문자를 인식하는 두 단계로 이루어져 있다. 하지만, 영상 품질이 좋지 않을 경우 문자 분리가 제대로 되지 않아 문자 인식이 실패하는 경우가 종종 발생하는 문제점이 있다. 그래서 최근에는 딥러닝을 이용한 문자분리 없이 문자를 연속적으로 인식하는 방법이 주로 이용되는 추세이다.

문자분리 없이 인식하는 방법으로 대표적인 것으로 CRNN(Convolutional Recurrent Neural Networks) [5, 6]을 들 수 있다. CRNN은 입력 영상으로부터 특징을 추출하는 CNN (Convolutional Neural Networks)과 문자열 모델링을 위한 RNN (Recurrent Neural Networks)를 결합한 네트워크로 예측한 문자열을 출력한다. RNN은 주로 LSTM(Long Short Term Memory)을 사용한다. 최근 CRNN을 이용한 문자인식이 많이 적용되고 있는데, 대표적인 것으로 Naver(Clova AI)의 OCR[7], Keras OCR[8], Tesseract OCR(v.4), Paddle OCR[9] 등이 있다. RNN은 sequence modeling하기 때문에 수행시간이 많이 걸리는 단점이 있다.

LPRNet[10]은 RNN을 사용하지 않고 차량번호판을 문자분리 없이 실시간으로 인식하는 네트워크로 실제의 다양한 차량번호판 인식에서 우수한 성능을 보여주고 있다. LPRNet은 RNN 대신 지역적인 문자 컨텍스트(local character context)를 이용하기 위해 넓은 컨볼루션 커널(1x13)을 사용하여 처리시간을 크게 줄였다. 더욱이 LPRNet은 Intel에서 발표한 넷으로서 Intel의 OpenVINO툴킷[11]를 이용하여 Intel CPU에 최적화가 가능한 장점이 있다.

본 논문에서는 차량번호 인식용으로 개발된LPRNet을 기반으로 PCB 부품의 인쇄문자의 고속, 고정도 인식에 최적화된 넷을 제안한다.

2. LPRNet

차량번호판 인식은 교통 제어/감시, 차량 인식, 주차 관리 등에서 이용되는 어렵고 중요한 작업이다. 이는 흐릿한 영상, 열악한 조명 환경, 다양한 관점에 의한 차량번호(특수 기호 포함) 모양의 변동, 날씨조건 등의 이유로 복잡한 문제이다.

LPRNet은 최근 많은 컴퓨터비젼 문제에서 우수한 성능을 보여주고 있는 심층 CNN기반으로 문자분리(character segmentation)없이 차량 번호판 영역에서 문자/숫자를 빠르게 인식할 수 있는 매우 효율적인 신경망이다(1 forward pass에 0.34 Gflops 소요. 참고로 resnet50[12]은 4 Gflops). 또한 이 모델은 OpenVINO의 사용으로 Intel CPU에 최적화되어 실시간 실행이 가능하고, 어려운 중국어 번호판에서 높은 인식 정확도를 내었다.

LPRNet의 백본 넷(backbone network)구조는 Fig. 1과 같다. 여기에 사용되는 small basic block은 기본적으로 convolutional block으로 Fig. 2에 보인다. Fig. 1의 백본 넷 구조의 맨 뒤 부분은 RNN 대신 사용되는 1 × 13의 컨볼루션으로 구성됨을 알 수 있다. LPRNet의 입력 영상 기본 크기는 24 × 94픽셀(Height × Width)이다.

Layer Type	Parameters
Input	94x24 pixels RGB image
Convolution	#64 3x3 stride 1
MaxPooling	#64 3x3 stride 1
Small basic block	#128 3x3 stride 1
MaxPooling	#64 3x3 stride (2, 1)
Small basic block	#256 3x3 stride 1
Small basic block	#256 3x3 stride 1
MaxPooling	#64 3x3 stride (2, 1)
Dropout	0.5 ratio
Convolution	#256 4x1 stride 1
Dropout	0.5 ratio
Convolution	# class_number 1x13 stride 1

Fig. 1. LPRNet Backbone network architecture. [8]

Layer Type	Parameters/Dimensions
Input	$C_{in} \times H \times W$ feature map
Convolution	# $C_{out}/4$ 1x1 stride 1
Convolution	# $C_{out}/4$ 3x1 strideh=1, padh=1
Convolution	# $C_{out}/4$ 1x3 stridew=1, padw=1
Convolution	# C_{out} 1x1 stride 1
Output	$C_{out} \times H \times W$ feature map

Fig. 2. Small basic block. [8]

3. 학습 및 실험

문자 분리 없이 텍스트 문자열을 인식하는 넷을 학습시키기 위해서는 대량의 데이터가 요구된다. 실제 문자 데이터를 대량으로 얻기 힘들기 때문에, 실제 문자와 유사하게 인공적으로 생성된 대량의 합성데이터(synthetic data)로 초기 학습을 수행한 후 실제 문자 데이터로 fine-tuning하는 것이 바람직하다.

문서나 자연 영상에서의 텍스트 인식을 위해 만들어진

기존의 합성데이터로 MJSynth[13], SynthText[14] 등 이 있지만, 용량이 너무 크고, 특성이 부품의 인쇄 문자와 많이 달라 사용하지 않고, 자체적으로 실제 부품 인쇄 문자와 유사하게 합성 데이터 셋을 생성하였다.

부품의 문자는 소문자가 거의 없고, 대부분 영문자 대문자와 숫자로 구성되므로 인식하고자 하는 문자는 영문 대문자와 숫자로 정하였다. 인식하고자 하는 대상 텍스트 문자열의 길이(문자개수)는 2~8개로 정하였다. 부품에 인쇄된 문자열은 대부분 8개 이하의 문자들로 구성되나, 8개 보다 많은 문자를 갖는 텍스트 문자열은 나누어 인식하면 된다. 폰트 유형은 부품 인쇄 문자 폰트로 자주 사용되는 Arial, OCR-A, OCR-B폰트 및 dot matrix 폰트를 포함하여 7 가지를 선정하였다. 위와 같이 하여, 약 10만개의 랜덤한 텍스트 문자열을 생성하여 합성 데이터 셋을 구성하였다.

Fig. 3에 합성 데이터 예를 보인다. 각 영상은 32 × 128 (H × W) 크기의 8-bit gray-scale 영상으로 생성되는데, 텍스트 문자열의 길이가 짧은 경우 나머지 영역은 0(영상에서 검정 픽셀)으로 채워진다.

합성 데이터 샘플 생성시 배경영상의 밝기는 50~150사이의 랜덤한 값으로 모든 픽셀을 채우고, 텍스트는 랜덤한 폰트를 선택해 배경밝기보다 40만큼 밝은 값을 갖도록 하며, 여기에 랜덤한 speckle noise를 더해 실제 문자와 유사하게 보이게 하였다. 또한 약간의 랜덤한 회전, 가로/세로 스케일링 변화, shear 등을 적용하여 생성하였다.

Fig. 3. Synthetic text examples.

실제 문자 데이터로 chip 부품 및 IC 표면 위에 인쇄된 157개의 텍스트 샘플을 얻었다. 이 샘플들은 인터넷으로 얻은 것과 실제 카메라로 부품 표면을 촬상한 영상으로부터 추출한 것이다. Fig. 4는 실제 부품 위에 인쇄된 문자열 텍스트 샘플을 보여주고 있다. Fig. 3과 4를 비교해 보면, 합성된 데이터 샘플이 실제 데이터 샘플과 상당히 비슷하게 보이는 것을 알 수 있다.

Fig. 4 Real text examples.

실제 문자 데이터로 chip 부품 및 IC 표면 위에 인쇄된 157개의 텍스트 샘플을 얻었다. 이 샘플들은 인터넷으로 얻은 것과 실제 카메라로 부품 표면을 촬상한 영상으로부터 추출한 것이다. Fig. 4는 실제 부품 위에 인쇄된 문자열 텍스트 샘플을 보여주고 있다. Fig. 3과 4를 비교해 보면, 합성된 데이터 샘플이 실제 데이터 샘플과 상당히 비슷하게 보이는 것을 알 수 있다.

합성 데이터는 95%의 학습 셋과 5%의 테스트셋으로 나누고, 학습 셋으로 학습하였다. 학습 시 원 영상의 밝기, 콘트라스트를 변화시키고, blurring시키면서 학습을 진행하였다. 합성데이터 영상의 크기는 32 × 128이므로, 넷 입력 시 24 × 94로 resize되어 입력된다. 학습 시 Adam optimizer[15]를 사용하고 batch size는 32로 설정하였다. 300000 iterations 학습 후 합성 데이터 테스트 셋에 대해서는 99.3%의 정확도를 보였지만, 실제 텍스트 데이터에서는 71.3%의 낮은 정확도를 내었다(Table 1 참조). 여기서, 텍스트 문자열 인식 정확도는 문자열의 모든 문자가 정확하게 인식해야 되며 한 문자라도 틀리면 인식 실패로 간주한다. 따라서, 인식 정확도를 높이기 위해서는 실제 텍스트 샘플 데이터로 fine-tuning할 필요가 있는 것을 알 수 있다.

Table 1. Recognition accuracy of the network trained using synthetic text data

data	accuracy [%]
Synthetic test dataset	99.3
Real text data (157 samples)	71.3

Table 2. Recognition accuracy of the network after fine-tuning using read text data

data	accuracy [%]
Real train data	100
Real test data	93.8

실제 텍스트 데이터 영상은 텍스트 문자열의 길이에 따라 영상의 가로/세로 길이 비율이 차이가 많이 나기 때문에 원 영상을 일률적으로 24 × 94크기로 resize하면 문자의 가로/세로 비율이 왜곡이 되어 문제가 된다. 따라서, 원 영상의 높이(세로)를 24가 되도록 원 영상의 aspect ratio를 유지하면서resize한 후, 폭(가로)이 94보다 작으면 나머지 영역은 0으로 채우고, 그렇지 않으면 가로를 94로 resize 하여 인식 넷에 입력한다.

실제 텍스트 데이터로 텍스트 인식 넷을 fine-tuning하기 위해서 먼저 실제 데이터를 90%의 학습 셋(141개)과 10%의 테스트 셋(16개)으로 나누고, 학습 셋으로 학습을 수행하였다. Fine-tuning 학습(50000 iterations) 후 실제 데이터데 대한 인식 정확도는 학습 셋에 대해서는 1.0(에러가 하나도 없이 모두 정확하게 인식)이 나왔고 테스트 셋에 대해서는 16개 중 1개의 인식에러를 내어 93.8%의 텍스트 인식정확도를 얻었다. 따라서, 실제 데이터를 fine-tuning한 결과 인식정확도가 71.3%에서 93.8%로 크게 증가했음을 볼 수 있다.

학습에 걸리는 시간을 살펴보면 합성 데이터로 학습할 때는 데이터의 개수가 매우 많기 때문에 시간이 상당히 걸리지만, 실제 데이터로 fine-tuning하는데는 실제 데이터의 개수가 매우 작기 때문에 시간이 매우 짧게 걸린다.

실험에 사용된 PC는 Intel Core i7 CPU (4GHz, 16GB)와 GPU보드로 Nvidia GeForce GTX 1080ti (11GB)에 운영체제 Windows 10 Pro (64bit)가 설치된 것이다. 딥러닝 프레임워크로 Anaconda 3, Tensorflow, Keras를 사용하였다. 또한, CPU에 최적화된 실행을 위해 Intel OpenVINO Toolkit (ver. 2021.1.110)의 IE(Inference Engine)을 사용하였다.

Table 3은 fine-tuning된 넷을 CPU와 GPU를 사용하여 길이 7인 문자열을 인식하는데 걸리는 시간을 나타낸 것이다. Nvidia GPU 1080ti 에서는 약 3 ms 정도 걸리고, 최적화되지 않은 CPU환경에서는 50 ms 정도로 오래 걸리지만, OpenVINO를 사용하여 최적화된 추론엔진을 사용하면 2 ms 정도밖에 걸리지 않아 GPU보다도 더 빠른 추론 속도를 보이는 것을 알 수 있다.

Table 3. Inference time for one text string with length 7 on various hardware platforms

target platform	1 text inference time [ms]
GPU (Nvidia GTX1080 ti)	~ 3
CPU (Keras, tensorflow)	~ 50
CPU (IE from Intel OpenVINO	~ 2

4. 결 론

본 논문에서는 문자 분리 없이 부품이 실장된 PCB에서 부품의 인쇄 문자/숫자를 빠르고 정확하게 인식할 수 있는 심층신경망을 제안하였다. 차량번호판 인식에서 우수한 성능과 고속 실행으로 검증된 LPRNet을 기본으로 하여, 실제와 유사하게 생성된 합성데이터로 1차 학습을 수행한 후 적은 수의 실제데이터로 fine-tuning한 결과, 우수한 정확도를 가지면서, CPU에서 GPU보다도 더욱 빠르게 인식할 수 있는 신경망을 얻었다. 정확도가 높은 인식 넷을 값비싼 GPU없이 CPU에서 실시간 실행 가능한 것이 가장 장점으로 다른 유사한 응용분야에도 성공적으로 적용 가능할 것으로 보인다.

감사의 글

이 논문은 2020년도 한국기술교육대학교 교수연구제 파견연구비 지원에 의하여 연구되었음.

참고문헌

1. Tesseract OCR [Internet]. Available: https://github.com/tesseract-ocr/tesseract
2. X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, J. Yan, “FOTS: Fast Oriented Text Spotting with a Unified Network,” arXiv:1801.01671, 2018.
3. X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, “EAST: An Efficient and Accurate Scene Text Detector,” IEEE Conference on Computer Vision and Pattern Recognition, pp.2642-2651, 2017.
4. Y. Baek, B. Lee, D. Han, S. Yun, H. Lee, “Character Region Awareness for Text Detection,” IEEE Conference on Computer Vision and Pattern Recognition, pp.9357-9366, 2019
5. B. Shi, X. Bai, and C. Yao, “An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition,” arXiv:1507.05717, 2015.
6. M. Namysl, I. Konya, “Efficient, Lexicon-Free OCR using Deep Learning,” 2019 International Conference on Document Analysis and Recognition (ICDAR), pp.295-301, 2019.
7. J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S.J. Oh, and H. Lee, “What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis,” 2019 IEEE International Conference on Computer Vision, pp.4714-4722, 2019.
8. Keras-ocr [Internet]. Available: https://github.com/faustomorales/keras-ocr.

9. Y. Du, C. Li, R. Guo, X. Yin, W. Liu, J. Zhou, Y. Bai, Z. Yu, Y. Yang, Q. Dang, H. Wang, “PP-OCR: A Practical Ultra Lightweight OCR System,” arXiv:2009.09941, 2020.
10. S. Zherzdev and A. Gruzdev, “LPRNet: License Plate Recognition via Deep Neural Networks,” arXiv:1806.10447, 2018.
11. OpenVINO Toolkit [Internet]. Available: https://docs.openvinotoolkit.org/latest/index.html.
12. K. He, X. Zhang. S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in Proceeding of 2016 IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-778, 2016.
13. M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In Workshop on Deep Learning, NIPS, 2014.
14. A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In CVPR, 2016.
15. D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” the 3rd International Conference for Learning Representations, San Diego, 2015.

접수일: 2021년 7월 24일, 심사일: 2021년 9월 1일,
게재확정일: 2021년 9월 11일