# 카카오 OCR 시스템 구성과 모델

**글 | 모종훈 simiro.m@kakaocorp.com** 심리학 공부를 한 특이 경력의 소유자입니다. 그럼에도 불구하고 카카오에 들어와 좋은 사람들을 만나고 나름의 성과를 내며, 이걸로 1년 만에 〈카카오 AI리포트〉에 글도 쓰게 되어 정말 자랑스럽네요. 더 좋은 기술을 만들기 위해 항상 노력하고 주말에도 열심히 공부하고 있습니다. 혼자서 다치는 버릇만 없어지면 앞으로 더 좋은 것들을 팍팍 만들어낼 수 있을 것 같습니다.

**글 | 오형석 hulk.oh@kakaocorp.com** 회사에서 하고 싶은 일들을 마음껏 할 수 있도록 도와주어서 이것저것 많이 실험해보고 연구해왔습니다. 또 연구한 것들이 다행히 순조롭게 진행되어 서비스가 되었습니다. 그 결과물들로 작년에는 번역기 학습데이터 정제에 관한 ABLEU를 다룬 글, 이번에는 OCR 서비스에 대한 글을 〈카카오 AI리포트〉에 게재하게 되었습니다. 앞으로도 좋은 연구 결과, 서비스를 선보일 수 있으면 좋겠습니다.

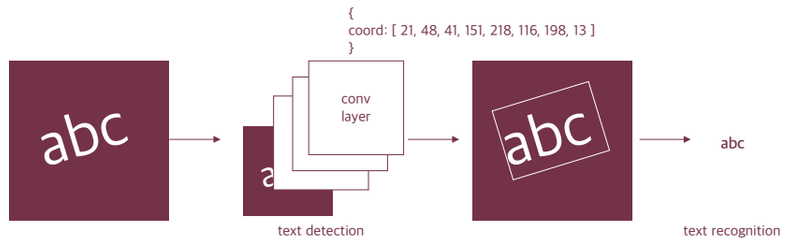




대부분의 사람들은 카카오의 수많은 텍스트 데이터에 초점을 맞추지만, 카카오의 다양한 서비스에서 가장 필요로 하는 핵심적인 기술 중 하나는 이미지 내의 글자를 자동으로 인식하는 것이다. 예를 들어 카카오 드라이버에서 자동차 번호판을 자동으로 인식하거나 카카오뱅크에서 자동으로 신분증의 글자를 뽑아내는 등, 이미지에서 자동으로 글자를 추출해 검색 및 광고에 활용하게 된다면 모바일 서비스의 사용성을 더욱 증대시킬 수 있다. 이러한 카카오 사내 수요의 증가에 발맞춰 최근 카카오 AI 기술팀에서는 사내에 광학 문자 인식(optical character recognition, OCR) 서비스를 베타 론칭했다. OCR란 사람이 쓰거나 컴퓨터로 입력되어 인쇄된 문자의 영상을 기계가 읽을 수 있는 문자로 변환하는 것을 말하는데, 이 분야 역시 딥러닝으로 인해 많은 발전을 이루고 있다. 이 글에서는 카카오 OCR의 전체적인 구조에 대해 먼저 살펴보고 좀 더 나아가 카카오 OCR만이 가지고 있는 차별점에는 무엇이 있는지 살펴보고자 한다. 그리고 끝으로 카카오 OCR가 앞으로 나아가려는 방향을 제시하며 이 글을 마무리하려고 한다.

## 시스템 구조도(System Architecture)

[ 그림 1 ] 카카오 OCR의 두 단계 모델 구조




카카오 OCR는 [그림 1]과 같이 글자 영역 탐지와 글자 인식의 두 단계 과정을 거쳐 이루어진다. 사용자가 요청을 보낸 이미지는 api 서버를 거쳐 먼저 글자 영역 탐지(text detection) 모델로 보내진다. 이 모델은 이미지 내에서 글자 영역을 탐지하고 탐지된 글자 영역을 잘라 글자 인식(text recognition) 모델로 전송한다. 글자 인식 모델은 잘려진 글자 박스들 속 글자를 인식하고 이 결과는 탐지 모델에서 나온 좌표값과 함께 사용자에게 전달된다.

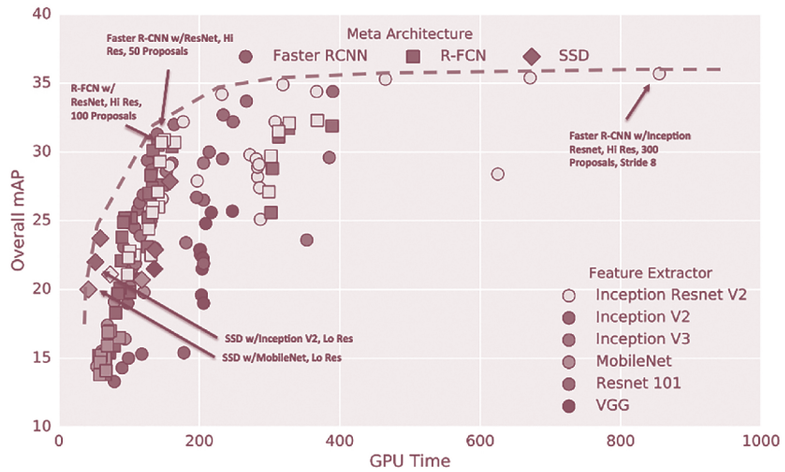
두 단계로 나누어 글자 인식을 진행하게 되면 서로 다른 데이터를 다양하게 활용할 수 있어 원활하게 학습을 진행할 수 있고, 글자 영역 탐지만이 필요한 서비스에서 자원을 좀 더 효율적으로 사용하는 것이 가능하다. 또한 다국어 글자 인식을 위해서 다양한 언어로 각기 학습된 글자 인식 모델을 사용하여 모델에서 필요한 사전 크기를 줄이고 언어별 정확도를 더욱 향상시킬 수도 있다. 다음으로는 위에서 이야기한 두 모델이 어떤 구조를 가지고 있는지에 대해 좀 더 자세히 알아보자.

## 글자 탐지 모델(Text Detection Model)

사물 탐지 모델은 크게 RCNN(region with convolutional neural network) 계열의 리전 프로포절(region proposal)을 거치는 모델과 SSD(single-shot multibox detector)나 YOLO(you

only look once)와 같은 single shot detector 계열로 나눌 수 있다. 위의 두 모델은 속도와 정확도 중 어느 쪽에 더 비중을 두는지에 따라 구조적으로 큰 차이를 가지고 있다. [그림 2]에서 볼 수 있듯 RCNN 계열은 보다 더 나은 정확도를 보이는 반면 시간당 처리할 수 있는 이미지의 수는 적고, 반면 single shot detector 계열은 RCNN에 비해 탐지 성능은 조금 떨어지지만 같은 시간 안에 훨씬 더 많은 양의 이미지를 처리하는 것이 가능하다. 적은 수의 리전 프로포절을 이용하면 Faster RCNN으로 충분히 속도를 향상시킬 수 있지만 이 경우에는 single shot detector 계열에 비해 성능 면에서 훨씬 뒤처지게 된다.

[ 그림 2 ] 사물 탐지(object detector) 모델의 성능 비교*1

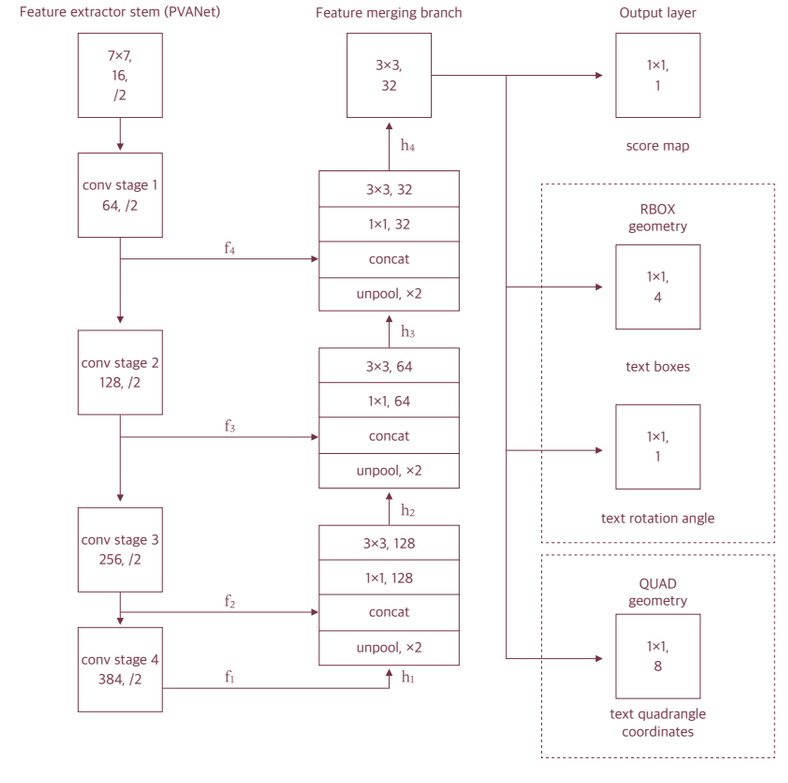



이러한 측면들을 고려하여 결정한 카카오의 OCR 글자 탐지 모델 구조는 single shot detector 계열에 속한다. 이는 실시간 서비스가 가능하기 위해서는 탐지 성능만큼이나 충분히 빠른 처리 속도가 필요했기 때문이다. 좀 더 구체적으로 이야기하자면, 카카오 OCR에서 사용하는 사물 탐지 모델은 YOLO와 거의 흡사한 구조를 가진 EAST(efficient and accurate scene text detector)를 기반으로 하되 조금 변형된 모델이다.

EAST의 구조는 [그림 3]과 같다. 이미지는 특징 추출(feature extraction) 레이어를 통과한 후 top-down skip connection을 이용해 좀 더 복잡한 정보와 저수준의 정보를 통합한 실제 이미지의 16분의 1배 크기의 특징 지도(feature map)를 만들어낸다. 사물 탐지는 이 특징 지도의 모든 셀(cell)에서 개별적으로 이루어지는데, 각 셀에서는 총 6개의 값을 예측하게 된다. 이 6개의 값은 셀에 글자가 존재하는지를 나타내는 점수, 글자 영역을 나타내는 영역 박스를 예측하는 AABB(axis-aligned bounding box), 마지막으로 회전각이다. YOLO에서는 박스를 x, y, h, w를 이용해서 나타냈지만 EAST에서는 top, left, bottom, right의 네 거리를 이용한 AABB를 이용한다는 점이 YOLO와의 가장 큰 차이점 중 하나라고 할 수 있다. 모델이 예측한 영역 박스는 점수를 기준으로 한 번 필터링된 후 NMS(non-maximum suppression)을 통해 비슷한 영역을 나타내는 박스들을 합치고 AABB와 회전각을 이용해 꼭짓점 4개 좌표를 계산한 다음 최종 결과를 산출한다.

*1 논문 | Huang, Jonathan, et al. 'Speed/accuracy trade-offs for modern convolutional object detectors', IEEE CVPR. Vol. 4. 2017.

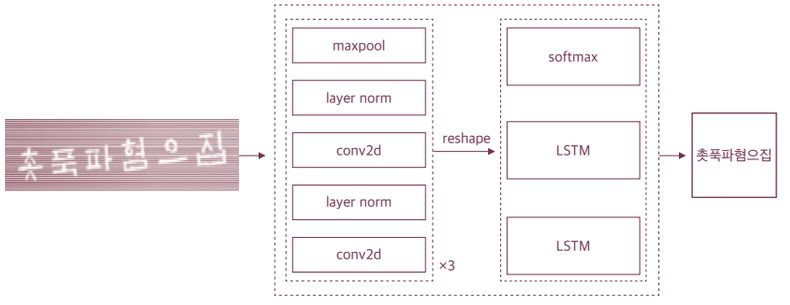
[ 그림 3 ] EAST 모델의 구조*2




**글자 인식 모델(Text Recognition Model)**

글자 인식 모델은 [그림 4]와 같은 구조를 가지고 있으며, 글자 이미지를 입력값으로 받아서 글자를 예측한다. 최근 카카오 서비스에 활용되고 있는 모델은 이 그림과 조금 다른 구조를 택했는데, 이 내용은 글의 마지막 부분에서 설명하도록 하겠다. 모델은 (1) convolutional layer, (2) recurrent layer로 이루어져 있으며, loss는 connectionist temporal classification loss(이하 CTC loss)를 쓰고 있다. 하나씩 알아보도록 하자.

[ 그림 4 ] 글자 인식 모델 구조(Text Recognition model)




*2 논문 | Zhou, Xinyu, et al. 'EAST: an efficient and accurate scene text detector', Proc. CVPR. 2017.

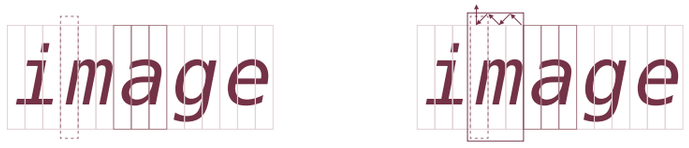
### (1) Convolutional layer

CNN layer는 글자 예측에 필요한 특징(feature)을 추출하는 역할을 하며, convolution 연산, normalization 연산, maxpool 연산, 이렇게 3가지로 이루어져 있다. Convolutional 필터는 필터의 파라미터를 학습해가며, 글자 예측을 위해 필요한 필터들을 자동으로 학습하게 한다. Layer normalization은 가변 길이 이미지를 지원하기 위해 높이와 채널 축에 대해서만 진행하였다. Maxpool의 경우 처음 입력은 높이와 너비를 2분의 1로 줄이며, 그 다음 두 번은 높이만 2분의 1로 줄였다. 너비를 더 줄이지 않은 이유는 CTC alignment를 하기 위해서는 실제 글자보다 많은 수의 단계가 필요하기 때문이다.

### (2) Recurrent layer

이처럼 CNN layer를 거치면 이미지의 각 부분마다 글자 예측에 필요한 정보를 얻어낼 수 있다. 그러나 이미지의 각 부분 정보만 가지고는 글자를 제대로 예측하기 힘들 수 있다. 따라서 RNN layer를 통해 이미지의 각 부분에서 예측할 때 주변의 정보를 활용하도록 만들어야 한다.
다음 [그림 5]을 예시로 보자. 왼쪽 그림의 빨간색 사각형은 이미지의 네 번째 특징 셀(feature cell)의 수용장 크기이다. 오른쪽 그림의 파란색 영역은 이미지의 네 번째 feature cell에서 'm'을 예측하기 위해 RNN layer를 통해 이용할 수 있는 유용한 주변 정보들이다. 이 그림 속 글자 'm'을 예측하려는데, CNN layer에서는 이미지의 일부분밖에 볼 수 없고, 이런 점을 고려할 때 'm' 글자의 첫 부분을 예측하려면 빨간색 부분만 보고 예측할 수밖에 없다. 따라서 'l(엘)', 'l(아이)', 'm(엠)' 등 여러 글자 사이에서 헷갈릴 수 있다. 그러나 RNN layer를 거치면 해당 부분에서 예측할 때 주변의 정보들을 이용할 수 있으며, 따라서 [그림 5]의 오른쪽 그림처럼 글자 'm'을 예측하기 위해 필요한 정보를 모두 볼 수 있다.

[ 그림 5 ] RNN layer를 통해 글자를 예측하는 방법




### (3) CTC loss

사실 CTC loss도 여느 기계 학습 방법과 같이 조건부 확률의 음의 로그값(negative log likelihood)이다. 그러나 조건부 확률을 나타내는 데 CTC alignment라는 특별한 방법을 사용하는 것뿐이며, 이는 이미지의 각 부분마다 어떤 글자가 있는지 정답을 알려주기가 굉장히 힘들기 때문이다. CTC alignment를 통해 이미지 X가 주어졌을 때 글자 Y가 나타날 조건부 확률을 계산하는 방법은 다음과 같다.

$$p(Y \mid X) = \sum_{L \in map(Y)} \prod_{t=1}^{T} p_t(l_t \mid X)$$

여기서 map(Y)는 글자 Y를 모델의 step-size T만큼으로 표현할 때 가능한 모든 alignment의 집합이며, $l_t$는 L의 $t$번째 글자이다. L이 map(Y)가 되려면 다음 변환을 거쳤을 때 L이 Y가 되면 된다.

(1) 인접한 똑같은 글자는 하나로 합친다. (2) Δ 토큰을 지운다.

[ 그림 6 ] 12구간으로 나눈 'hello'


위의 그림을 12구간으로 나눠서 순차적으로 살펴보며 hello라는 글자를 만든다고 해보자. (1) 'hheΔΔlllΔlloo'나 (2) 'ΔΔhheelΔlloo'는 모두 위의 방법을 통해 hello라는 글자에 도달할 수 있다(Δ는 blank 토큰을 의미한다). 결국 위의 식에 나타난 map(Y)은 (1), (2) 같은 hello를 완성시킬 수 있는 패스(path)들의 집합이며, 각 패스에 대해 확률을 더한 것이 'hello'가 될 조건부 확률이다. 모든 alignment에 대해서 확률을 더하는 방법은 동적 프로그래밍(dynamic programming)으로 구하게 된다.

## Self-Attention을 이용한 더 빠르고 정확한 인식 모델

앞서 설명한 모델들로 서비스하면서 모델을 개선하기 시작하였다. 그중 첫 번째 성과는 인식(recognition) 모델의 속도 개선이었는데, 이 섹션에서는 그 방법에 대해서 설명하겠다.
우리의 1차 인식 모델 RNN layer는 모든 입력값(input)에 의존적이며 순차적으로 글자를 뽑아낼 수밖에 없는 한계를 가졌다. 이를 개선하기 위해 RNN layer처럼 이미지의 다른 부분을 참고해 패러럴(parallel)하게 글자를 뽑아낼 수 있는 여러 가지 방법을 실험했으며, 최종적으로 self attention을 이용한 구조로 성능은 유지하면서 속도를 빠르게 만들었다. Self attention은 지역성(locality)이 존재하는 convolution filter와 달리 모든 이전 입력에 대해서 참조를 한다.

[ 그림 7 ] Convolution 필터와 Self-Attention 비교*3


기존 모델에 LSTM layer 대신 multi-head self-attention을 적용하여 학습 속도는 약 4배 빠르게 만들 수 있었으며, 실제 서비스에서도 다음과 같은 효과를 얻었다.

- Character Error Rate 기존 모델 대비 25% 감소
- TPS 27% 증가
- Latency 21% 감소
- CPU 사용량 50% 감소
- GPU 사용량 24% 감소

*3 참고 | http://deeplearning.hatenablog.com/?page=1513937688

**결론**

지금까지 카카오 OCR의 시스템 구성에 대해 살펴보고, 시스템을 구성하는 2가지 모델인 글자 탐지 모델(text detection model)과 글자 인식 모델(text recognition model)에 대해 알아봤다. 그리고 특히 인식 모델(recognition model)을 개선하기 위해 어떤 노력들을 했고, 결론적으로 self attention layer를 사용하여 좋은 결과를 얻은 경험담을 공유했다. 이제 막 첫 걸음을 뗀 카카오 OCR는 아직도 미흡한 점이 많다. 그러나 앞으로 지속적으로 성능을 개선하고 지원하는 언어의 수를 늘려가 정말 멋진 서비스를 만들 수 있을 것이라고 믿고 있다. 다음 번 <카카오 AI리포트>에서 더 발전한 OCR에 대한 내용을 또다시 소개할 수 있기를 희망한다.

〈카카오 OCR 시스템 구성과 모델〉 브런치로 연결되는 QR 코드입니다.