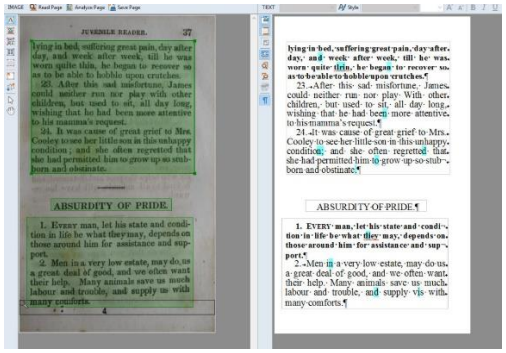
Scene Text Detection and Recognition

OCR
이미지 속 글자를 컴퓨터 문자로 변환하는 문제,
Text Detection과 Text Recognition 문제를 순차적으로 해결하는 문제
OCR을 이용하면 이미지 속 글자 수 검색이나 글자 변경 따위가 가능하다.



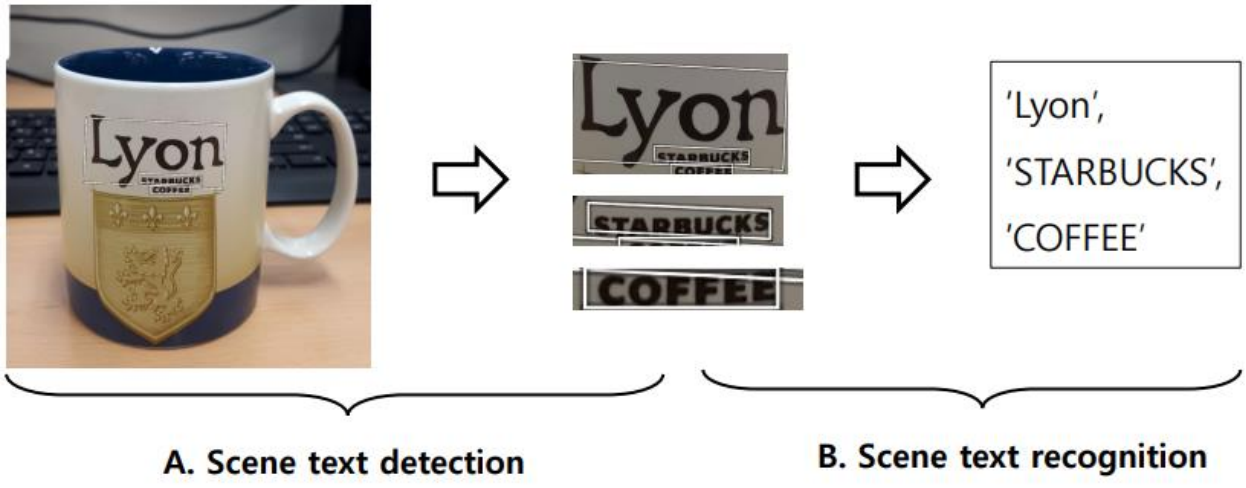
Scene Text Detection and Recognition
일상적인 이미지에서 문자가 있는 영역을 탐지하고 이를 컴퓨터 문자로 변환하는 문제,
이미지 번역, 번호판 인식, 이미지 검색 등 실생활에서 다양하게 활용되고 있음

전통적인 OCR문제는 흰 배경의 책에 있는 문자를 검출하는 경우가 많았다.
그러나 Scene Text Detection and Recognition 문제는 일상적인 이미지를 다루다 보니 비스듬하거나 회전된 문자,
곡선형으로 나열된 문자 등도 처리해야 되니 더 어려운 문제라고 볼 수 있다.

Detection Format의 예시는 다음과 같다.
1. RECT: 기울어지지 않은 문자
2. RBOX: 기울어진 문자 (각도 정보도 알아야 됨)
3. QUAD: 기울어지면서 크기도 다른 문자 (사각형의 꼭짓점을 모두 알아야 됨)
4. POLY: 곡선형으로 나열된 문자 (다각형의 여러 꼭지점을 분석)

Text Detection
이미지에서 문자가 위치한 '영역'을 탐지하는 문제

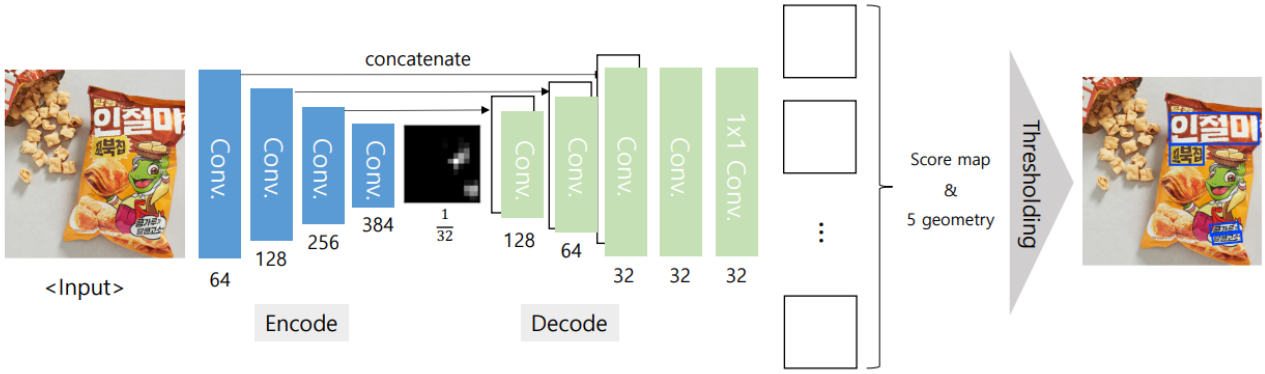
Text Recognition
단어 영역이 어떤 문자인지 찾아내는 분류 문제



지금부터 Text Detection의 대표적인 모델인 **EAST**와 Text Recogniton의 대표적인 모델인 **CRNN**에 대해 소개하려고 한다.

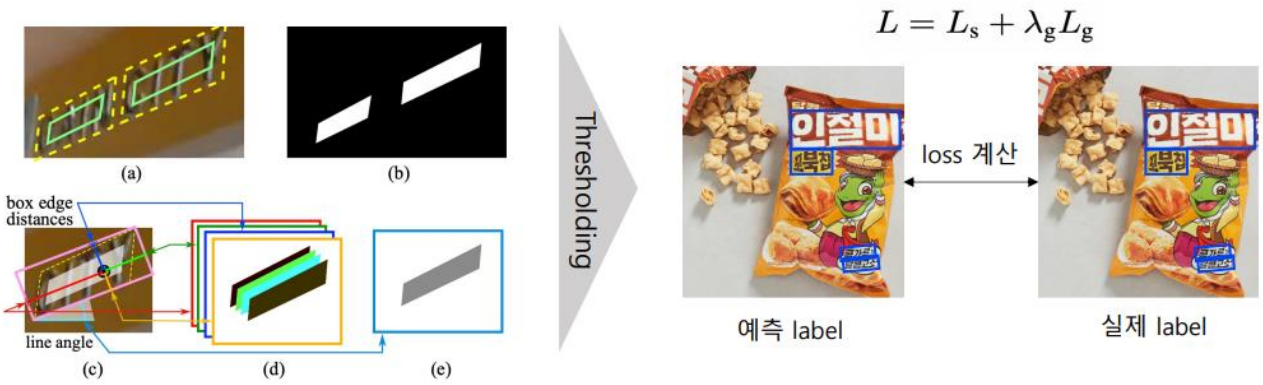
EAST: AN Efficient and Accurate Scene Text Detector

하나의 CONV 블록을 거쳐서 RBOX나 QUAD를 예측하는 모델,
Segmentation에 많이 사용되는 U-Net구조와 유사함



U-Net은 여러 Conv연산을 거쳐서 Feature Map크기를 작게 만들고 어느 순간부터는 DeConv연산을 거쳐서 Feature Map크기를 다시 크게 만드는 구조이고 Skip Connection을 통해 fine-grained한 특징도 잘 찾을 수 있게 하는 네트워크이다. EAST는 U-Net과 비슷한 구조를 통해 RBOX, QUAD 예측에 필요한 특징들을 빠르면서도 정확하게 찾았다.
이 때 Feature Extractor(Encoder)는 PVANet 혹은 VGGNet을 채택했다고 한다.

EAST의 출력으로는 Score map & 5 geometry이 있다. (RBOX 기준)



Score Map (그림:b)은 각 픽셀이 단어 영역 내에 있을 확률이 담긴 Map으로 확률이 높을 수록 1에 가까우니 흰색으로 나타낼 수 있다. 확률이 낮은 픽셀은 0에 가까우니 검정색으로 나타난다. 그러나 이 Score Map이 정확히 흰 색과 검정 색으로 나타내지는 것은 쉽지 않다.
따라서 5 geometry를 추가로 이용한다.
5 geometry는 text box와 text rotation angle 정보로써 그림 (d)와 (e)를 참고하면 된다.
text box(d)는 각 픽셀과 추정한 box의 4개 변 사이 거리가 담겨 있다. 예를 들어 첫 번째 채널은 픽셀과 box 밑 변과의 거리, 두 번째 채널은 픽셀과 box 윗 변과의 거리 등으로 이루어지므로 총 4개의 채널로 이루어져 있다. 이 때 모든 채널의 값이 높게 나타나면 박스의 중심으로 추정할 수 있다.
text rotation angle(e)은 말 그대로 추정한 box가 회전된 각도이다.

이러한 score map과 5 geometry를 적절하게 조합한 뒤 특정 임계치를 넘는 추정 box는 진짜 box라고 예측을 한다.
(적절하게 조합하는 방법을 너무 복잡하므로 생략한다. 다만 올바르지 않은 추정 box는 box의 중심이나 angle을 잘 계산하지 못 해 점수가 낮을 것이라고 생각한다.)

결국 모델의 결과(score map, 5 geometry)를 가지고 box들을 추정한 뒤 각각 점수를 계산해서 특정 점수 이상의 가진 box들만 선별하는 알고리즘이라고 볼 수 있다.

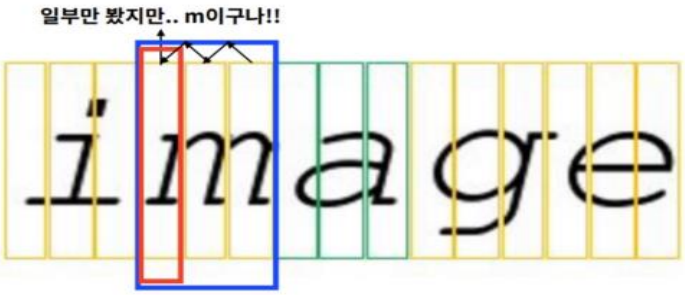
하지만 이 알고리즘을 거쳐도 실제 box들과 차이가 있을테니 loss를 계산 후 backpropagation해서 더 정확한 정보를 찾는다. Ls는 score_map의 손실, Lg는 5 geometry의 손실이 된다.

다음은 EAST 모델의 실험 결과이다. (ICDAR 2015)
SOTA모델의 F-Score는 90% 이상을 가지지만 EAST는 80% 를 가진다. 하지만 Scene Text Detection 분야에서 중요한 방향성을 제시한 논문이라고 한다.

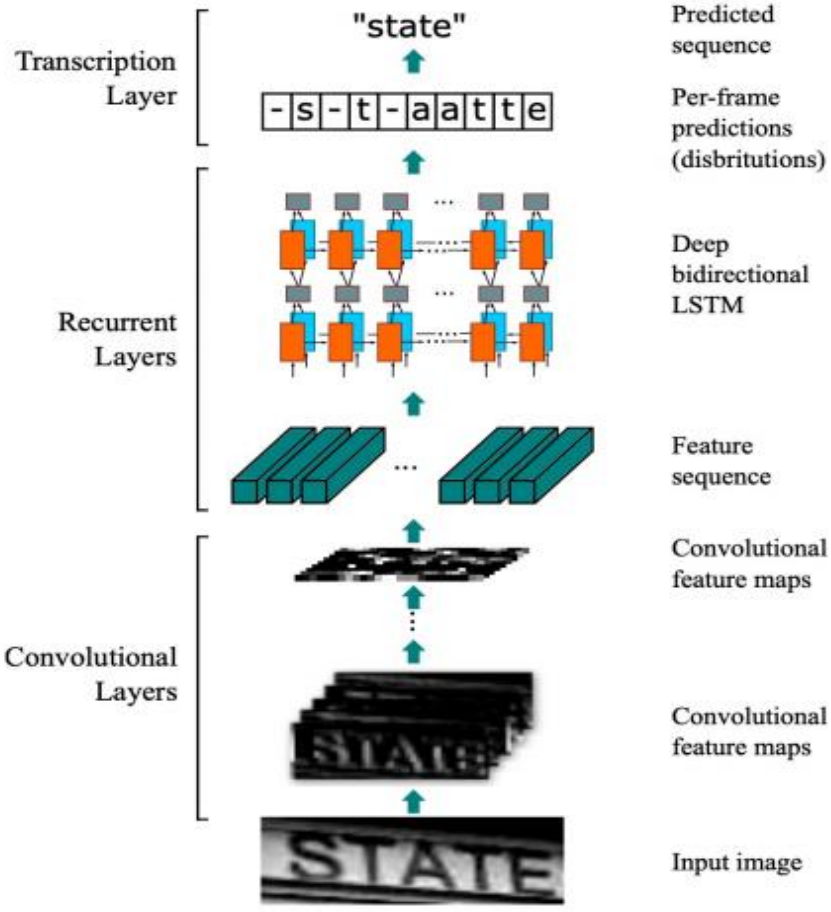
Algorithm	Recall	Precision	F-score
Ours + PVANET2x RBOX MS*	**0.7833**	0.8327	**0.8072**
Ours + PVANET2x RBOX	0.7347	**0.8357**	**0.7820**
Ours + PVANET2x QUAD	0.7419	0.8018	0.7707
Ours + VGG16 RBOX	0.7275	0.8046	0.7641
Ours + PVANET RBOX	0.7135	0.8063	0.7571
Ours + PVANET QUAD	0.6856	0.8119	0.7434
Ours + VGG16 QUAD	0.6895	0.7987	0.7401
Yao *et al.* [41]	0.5869	0.7226	0.6477
Tian *et al.* [34]	0.5156	0.7422	0.6085
Zhang *et al.* [48]	0.4309	0.7081	0.5358

An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition (CRNN)

CRNN은 CNN과 RNN을 결합해 Text Recognition 문제를 해결한 초기 모델이다.
Text Recognition은 이미지에서 글자 특징을 찾아 분류하는 문제니까 CNN만 필요하다고 생각할 수 있지만 RNN도 반드시 필요하다. 왜냐하면 CNN은 시계열 데이터와 같이 앞 뒤로 연관성이 있는 정보는 파악하기 힘들기 때문이다. 문자 같은 경우, 시간 순서대로 왼쪽에서 오른쪽으로 작성이 되므로 순차 데이터이고 다음 단어까지 고려해서 작성이 되는 경향이 있다. 따라서 각 글자들이 앞뒤로 연관성이 있다고 볼 수 있다. 또한 글자 단위가 아닌 글자 벡터 단위에서도 앞뒤로 연관성이 있다고 볼 수 있다. 아래 그림에서 m은 세 벡터로 나뉘어 지고 다시 세 벡터를 기준으로 feature가 추출된다. 그런데 단순히 특정 위치에 어떤 feature가 있다는 정보 만으로는 해당 벡터를 i로 분류해야 되는지 m으로 분류해야 되는지 알 수가 없다.
따라서 앞뒤의 다른 정보를 종합적으로 고려하는 모델(ex) BI-LSTM 등)이 필요하다.

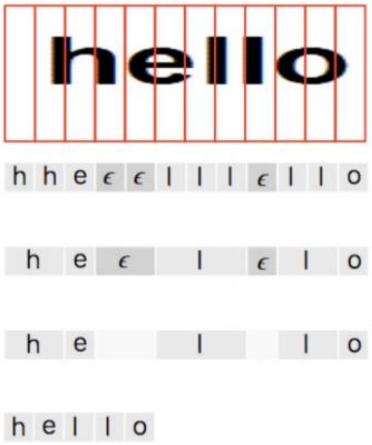


아래는 CRNN 모델의 구조이다.



1) CONV 연산을 통해 특징을 추출한다.
2) 추출된 Feature Map을 열 벡터 단위로 나눠 시퀀스 형태로 변환한다.
(각 열 벡터는 글자 혹은 공백에 대한 특징을 가진다.)
3) BI-LSTM모델을 이용해 각 벡터에 대한 글자 예측값을 출력한다.
4) CTC 알고리즘을 통해 중복 문자와 공백 등을 제거한 후 최종 단어 예측 값을 출력한다.

이 때 CTC 알고리즘을 간단하게 설명하면 순차적으로 데이터를 확인하면서 이전 state와 중복되면 합치는 것이다.



예를 들어, 'hello'라는 글자가 담긴 영상이 BI-LSTM을 통과해서 'hhe--lll-llo'라는 분류 결과가 나왔을 때, 순차적으로 확인해 나가면서 중복을 제거하면 'he-l-lo'가 나온다. 여기서 다시 공백(blank)을 제거하고 최종적으로 합쳐서 'hello'가 나오도록 하는 것이다.

	IIIT5k			SVT		IC03				IC13
	50	1k	None	50	None	50	Full	50k	None	None
ABBYY [34]	24.3	-	-	35.0	-	56.0	55.0	-	-	-
Wang *et al.* [34]	-	-	-	57.0	-	76.0	62.0	-	-	-
Mishra *et al.* [28]	64.1	57.5	-	73.2	-	81.8	67.8	-	-	-
Wang *et al.* [35]	-	-	-	70.0	-	90.0	84.0	-	-	-
Goel *et al.* [13]	-	-	-	77.3	-	89.7	-	-	-	-
Bissacco *et al.* [8]	-	-	-	90.4	78.0	-	-	-	-	87.6
Alsharif and Pineau [6]	-	-	-	74.3	-	93.1	88.6	85.1	-	-
Almazán *et al.* [5]	91.2	82.1	-	89.2	-	-	-	-	-	-
Yao *et al.* [36]	80.2	69.3	-	75.9	-	88.5	80.3	-	-	-
Rodrguez-Serrano *et al.* [30]	76.1	57.4	-	70.0	-	-	-	-	-	-
Jaderberg *et al.* [23]	-	-	-	86.1	-	96.2	91.5	-	-	-
Su and Lu [33]	-	-	-	83.0	-	92.0	82.0	-	-	-
Gordo [14]	93.3	86.6	-	91.8	-	-	-	-	-	-
Jaderberg *et al.* [22]	97.1	92.7	-	95.4	80.7*	**98.7**	**98.6**	93.3	**93.1***	**90.8***
Jaderberg *et al.* [21]	95.5	89.6	-	93.2	71.7	97.8	97.0	93.4	89.6	81.8
CRNN	**97.6**	**94.4**	**78.2**	**96.4**	**80.8**	**98.7**	97.6	**95.5**	89.4	86.7

CRNN의 결과를 살펴 보면 Scene Recognition Dataset 중 IIIT5k에서 97.6%라는 가장 높은 성능을 보인다.
이전 모델 같은 경우, 많이 알려진 단어들만 잘 인식하는 것이 한계였다면 CRNN은 임의의 단어(예:전화번호, SVT Dataset)에 대해서도 잘 처리할 수 있다고 한다.

P.S. OCR 성능을 높이려면?

마지막으로 OCR 성능을 높이기 위한 데이터 전처리 방법을 소개하겠다.

1. Grayscale

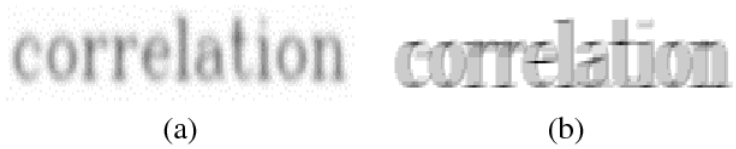
글자를 학습하는 데 배경의 색상까지는 필요하지 않다. 그레이스케일을 하면 CNN에서 Feature를 찾을 때 배경색이 아닌 글자에 대해서 더욱 집중하도록 만든다.

2. Thresholding

일반적으로 글자는 다른 영역에 비해 어두운 색을 지닌다. 따라서 어두운 글자 색을 기준으로 흰 색과 검은 색으로 영상을 분할할 수 있다. (ex) 글자는 검정, 나머지는 흰 색) 이 과정을 거치면 배경에 대한 디테일이 많이 떨어져서 글자에 더욱 집중할 수 있다. (글자에 대한 특징이 더 많이 나옴)

3. Blurring

글자 자체에 noise가 많거나 글자 주변에 noise가 많으면 recognition 성능이 떨어질 수 있다.
따라서 영상 자체를 흐릿하게 해서 noise를 제거한다.



(a) (b)

Fig. 3. Example of blurred images in *Gaussian* set (a) and *Motion* set (b).

4. Erosion, Opening

Erosion과 Opening은 검은 픽셀 주변의 객체를 축소(remove)시켜서 검은 색 영역을 더 늘리는 역할을 한다.
글자가 검은 색이기 때문에 글자 입장에서는 글자가 더 두꺼워지는 셈이다. 또한 점 형태의 글자인 경우 점이 메꿔지는 효과도 보인다. 따라서 글자를 더 잘 인식할 수 있다.