OCR
(Optical Character Recognition)

1조

김성택 정보통계학과 **김수정** 정보통계학과 **박한별** 미디어커뮤니케이션학과
임현희 정보통계학과 **정시훈** 정보통계학과 **정형윤** 정보통계학과

CONTENTS

주제 소개

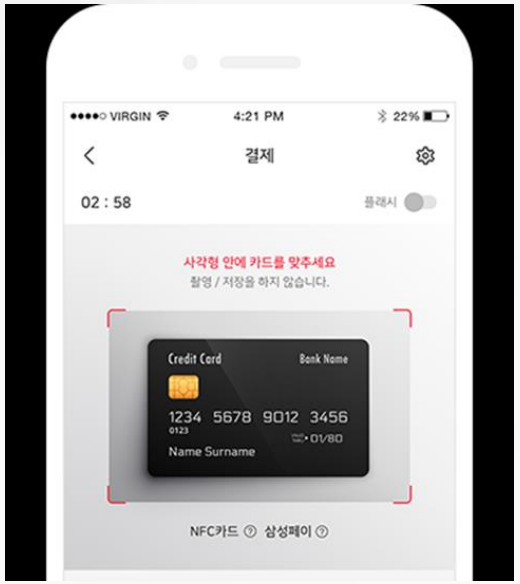
1-1 서론

1-2 OCR 파이프라인

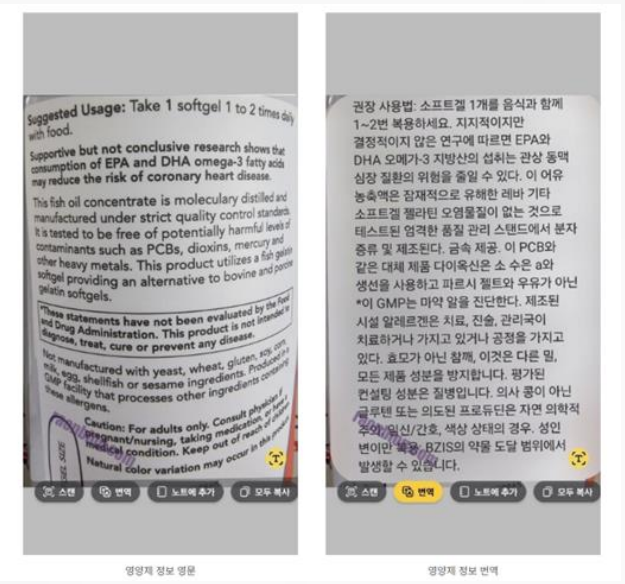
서론

OCR(Optical Character Recognition, 광학 문자 인식)은 사진 속 글자를 텍스트 데이터로 변환하는 기술이다.

*자동차 번호판 인식



*카드 정보 인식

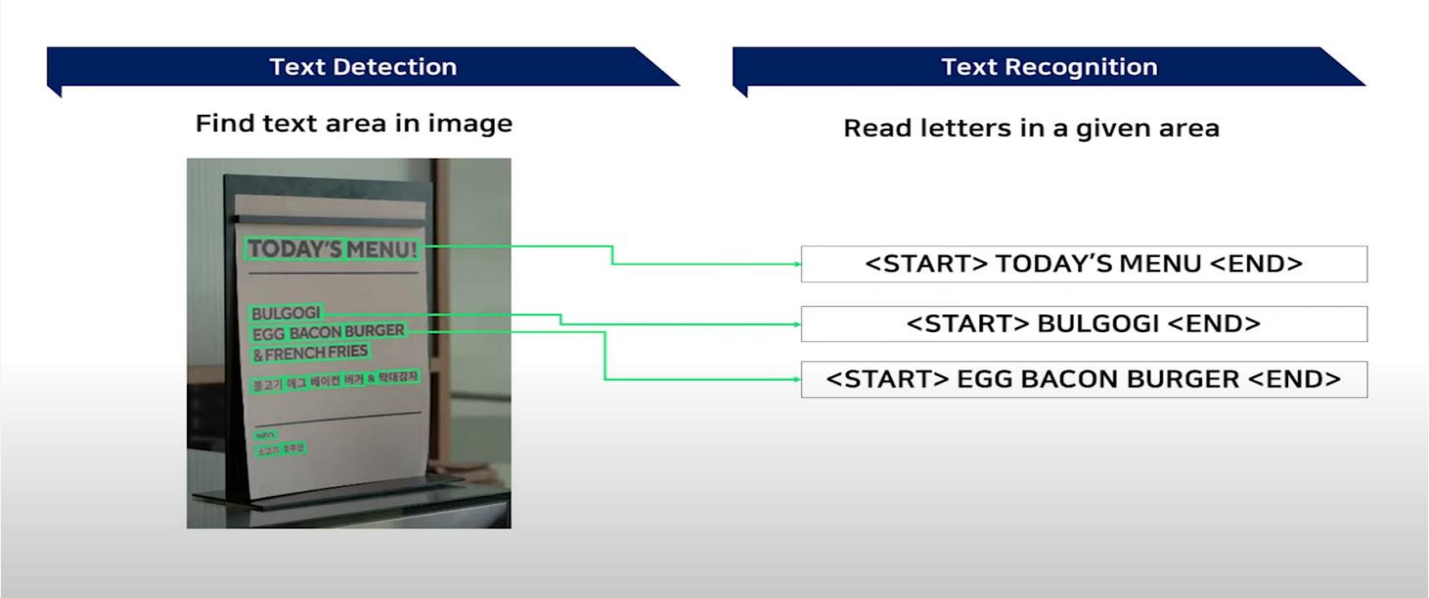


*활자 인식

OCR 파이프라인

OCR = Text detection + Text recognition

- **Text Detection**
  이미지에서 text의 bounding box를 찾아내는 과정
- **Text Recognition**
  bounding box 내에 존재하는 text를 인식하는 과정



관련 연구

2-1 Text Detection

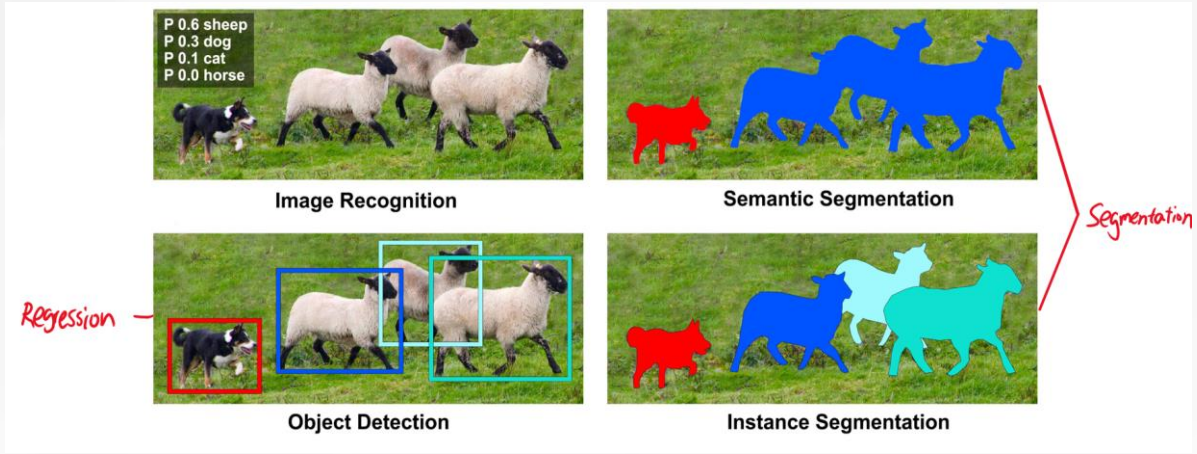
2-2 Text Recognition

Text Detection

Regression vs Segmentation

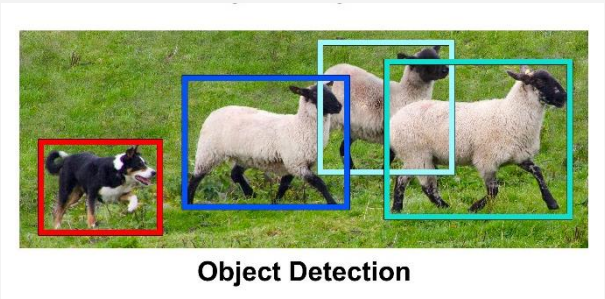
Regression
Bounding box의 좌표와 사이즈를 찾아내는 방법

Segmentation
이미지의 픽셀 단위로 text instance를 분류하는 방법



(*Object detection)

TextBoxes (regression)



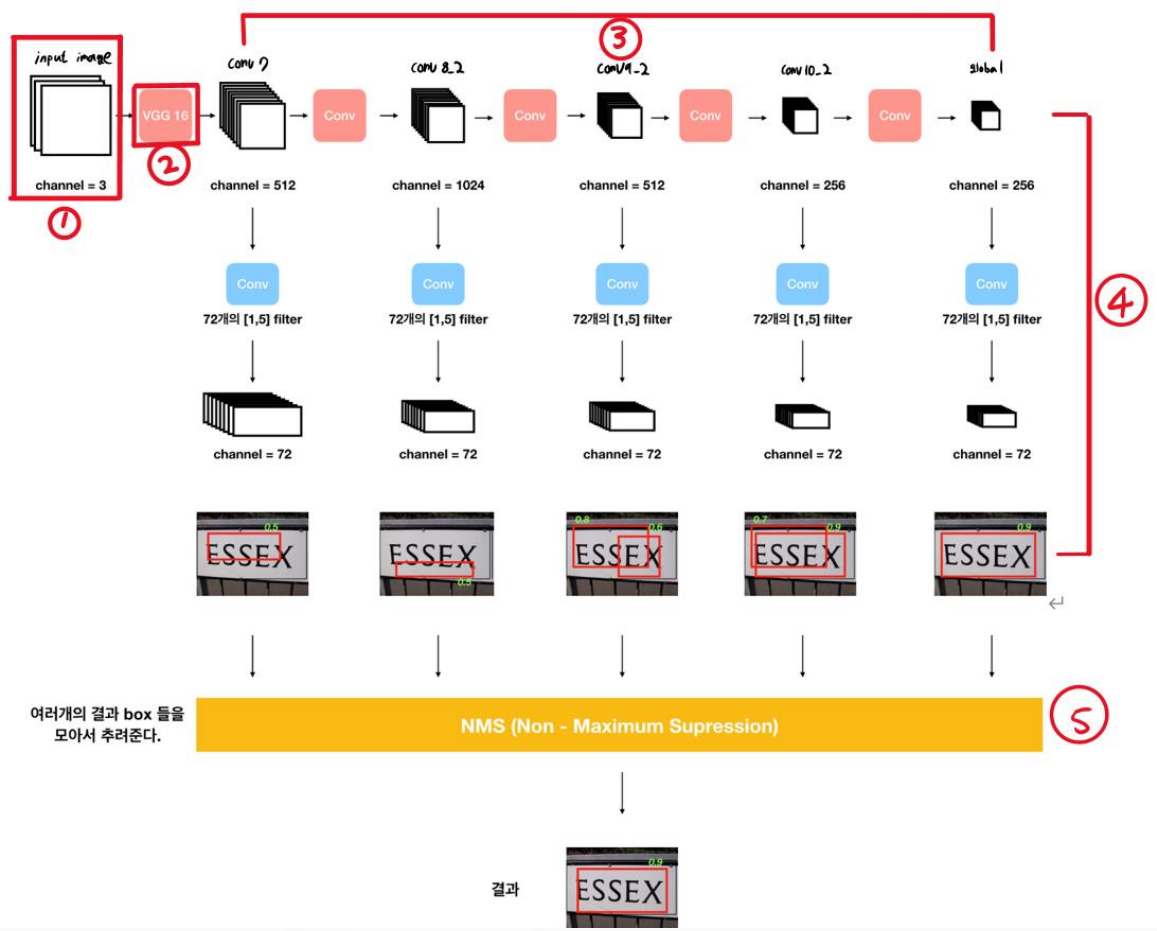
PixelLink (instance segmentation)

Text Detection

TextBoxes

TextBoxes 개요

1. Input Image
2. (FC를 Conv로 바꾼 형태의) VGG-16
3. 각 Conv layer를 거치면서 다양한 크기의 해상도를 가진 feature map 추출
4. 3의 각 feature map에서 conv 연산을 통한 default box 반환
5. 얻은 default box들을 모아 후처리 과정을 통해(NMS) 최종 bounding box를 얻는다.



Text Detection

TextBoxes

Network Architecture

구조

- VGG16 NET 모델 기반
- Base Network : convolution 연산을 통해 다양한 크기의 feature map을 추출

  Map Point : text 영역 후보를 default box 형태로 추출해 text-box layers로 변환

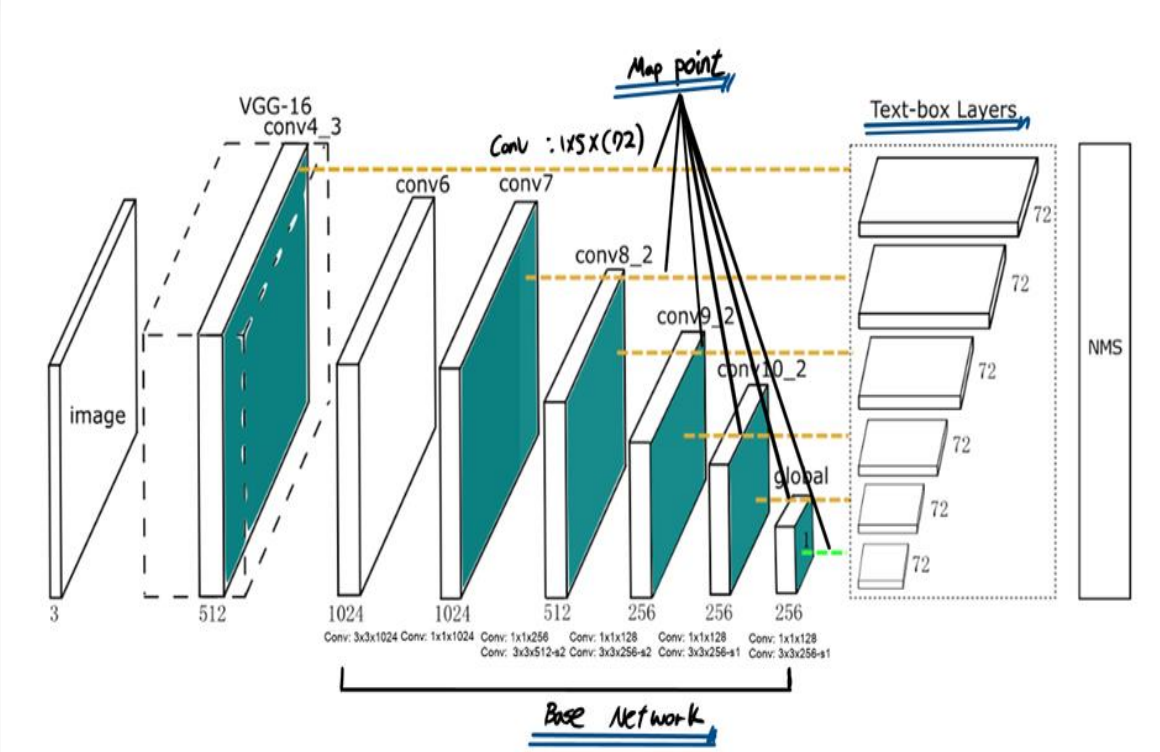
  NMS : 최종 bounding box를 얻는 후처리 과정

Base Network

- 9개 layer의 convolution 연산으로 6개의 feature map을 생성한다.
- 크기가 다른 6개의 feature map은 다양한 크기의 text를 detection하기 위함이다.

Map Point

- 문자의 bounding box가 가로로 긴 점을 고려하여, (1x5) filter로 convolution 연산을 수행한다.
- 각 feature map마다 72개의 채널을 가지는 text-box layers를 추출한다.



Text Detection

TextBoxes

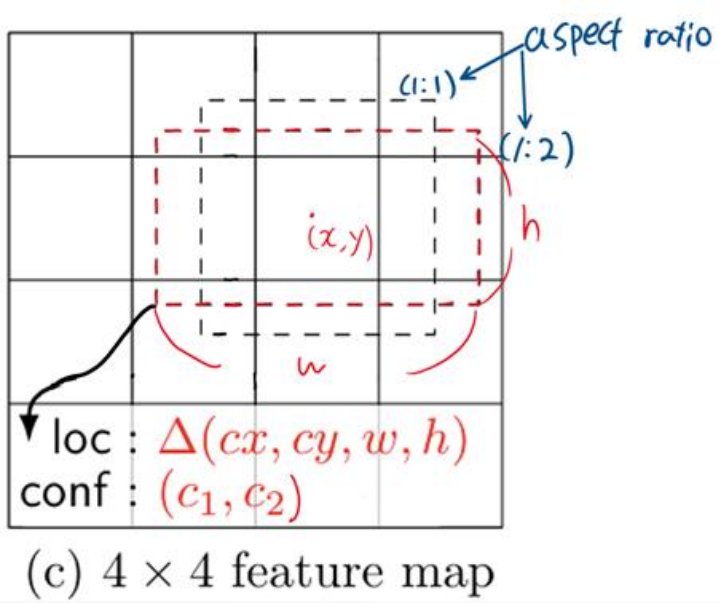
Text-box layers

*72 [Channel] = 12[Default Box] *(4 [offset (dx, dy, dw, dh)] + 2[confidence (c1, c2)])*

- Default Box
  텍스트를 포착하기 위한 box.
  aspect ratio : 1:(1,2,3,5,7,10)[center], 1:(1,2,3,5,7,10)[bottom]

- Offset
  x, y : box의 위치
  w, h : box의 너비와 높이

- Confidence
  text prediction 신뢰도
  c1 = positive box, c2 = negative box



Text Detection

TextBoxes

Loss Function

Total Loss = (Location Loss + Confidence Loss)/N

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)) \quad (1)$$

N → positive box 개수

α → default : α=1

Location Loss : positive box만 사용 (smooth_L1)

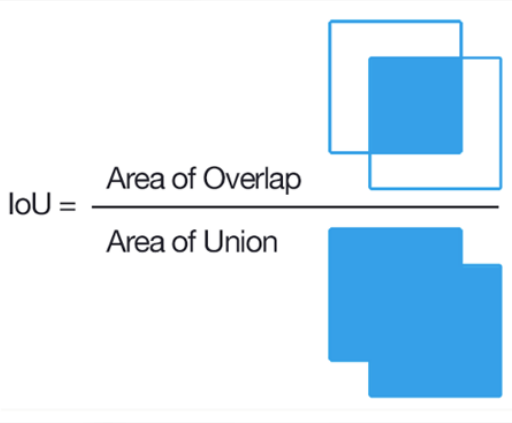
$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

Confidence Loss :
positive box -> c1, negative box -> c2 (softmax)

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \quad (3)$$

*IoU (Intersection of Union)
intersection / union (of Default Box & Ground Truth Box)

- IoU > 0.5 (positive box)
- IoU < 0.5 (negative box)



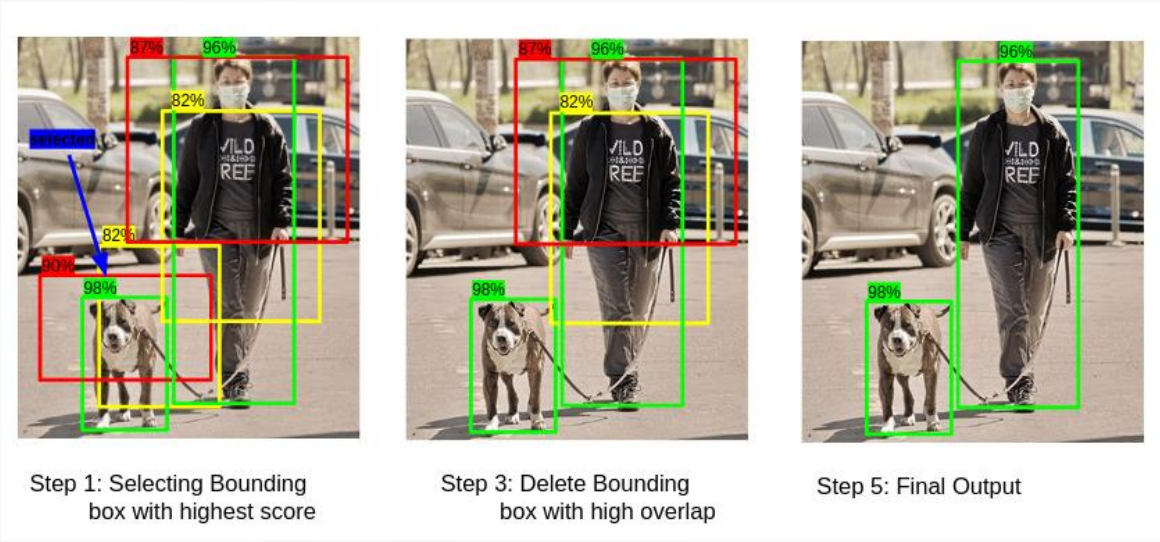
Text Detection

TextBoxes

NMS

NMS(Non Maximum Supression)
객체에 가장 적합한 bounding box를 선택

- step 1 : score가 가장 높은 box를 선택.
- step 2 : IoU > 0.5인 box를 제거.
- step 3 : 다음으로 높은 score의 box를 선택.
- step 4 : 2~3을 반복.



Text Detection

PixelLink

PixelLink 개요

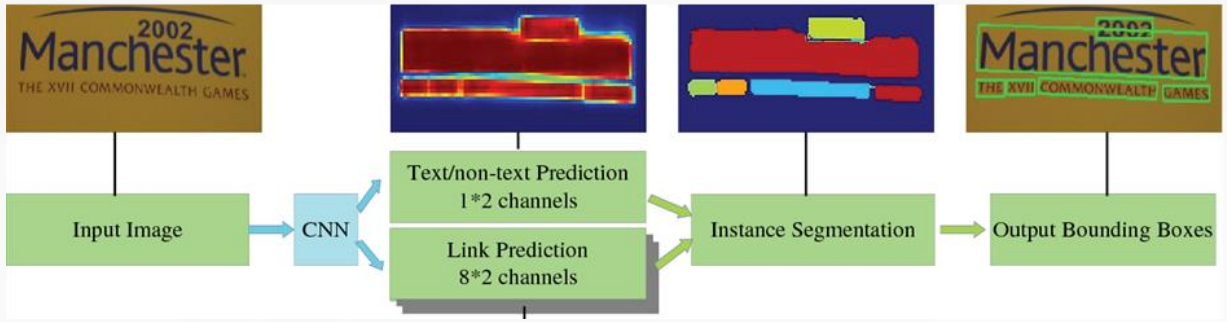
1. Input Image

2. Text/non-text prediction
   text pixel : (positive) | non-text pixel : (negative)
   -> *channels= 1(pixel) * 2(positive/negative)*

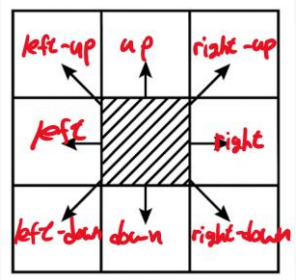
   Link prediction
   주어진 픽셀과 그 이웃들이 같은 instance 내에 있는지에 따라
   link positive or link negative
   *-> channels= 8(pixel) * 2(positive/negative)*

3. Instance Segmentation
   예측된 positive text 픽셀은 예측된 positive link를 통해
   CC(Connected Components)로 결합됨.

4. Output Bounding Box



*PixelLink 파이프라인



*Link Prediction

Text Detection

PixelLink

Network Architecture

구조

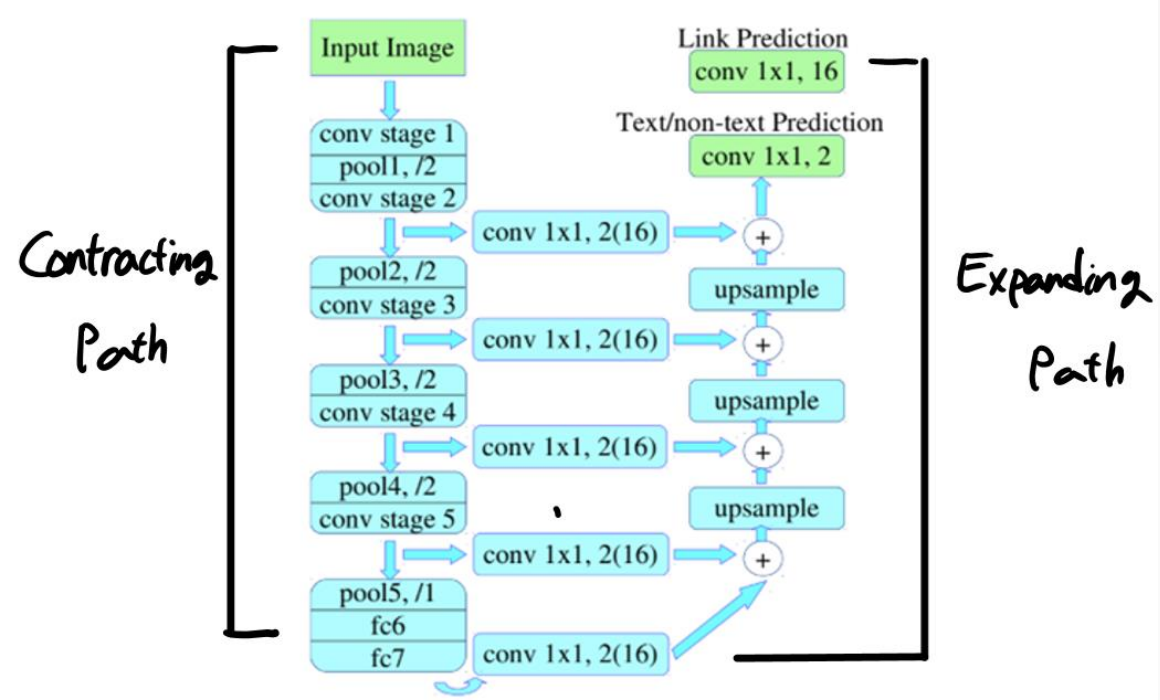
- VGG16(UNET) 모델 기반
- Contracting Path
  이미지의 의미(context)정보 추출

  Expanding Path
  context를 픽셀 위치에 결합(localization)

  Bridge
  얕은 layer의 feature map을 결합

Contracting Path

- Convolution Layer :
  kernel = 3x3 / pad = 1 (feature map 크기 유지)
- Pooling Layer :
  stride = 2 (feature map 크기 1/2)



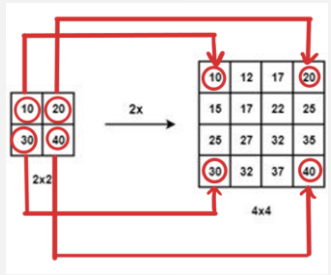
Text Detection

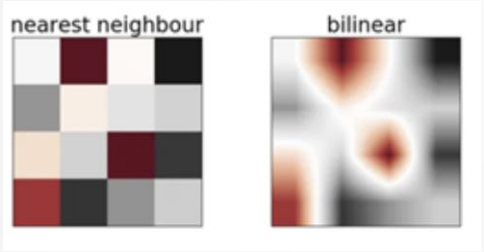
PixelLink

Expanding Path

- 각 층의 convolution(kernel = 1x1) 연산된 feature map과 upsampling layer을 합연산
- Upsampling

*bilinear interpolation
작아진 feature map의 크기를 늘리면서 해상도를 복원하는 방법

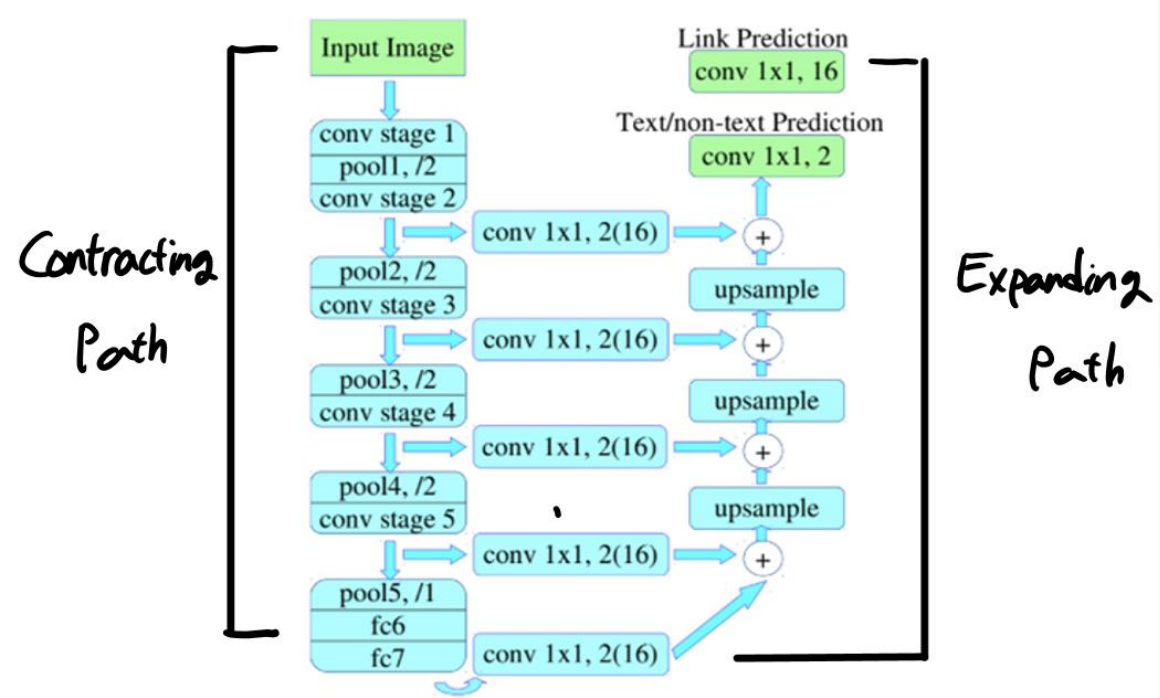




각 픽셀은 4개 지점까지의 거리에 반비례하여 곱연산
upsampling에 의한 feature map의 크기는 2배가 됨

Output

- 2개의 헤더 (softmax)
  -> positive pixel과 positive link로 instance를 나타내는 CC 생성



Text Detection

PixelLink

Loss Function

Total Loss

L_link는 positive 픽셀에서만 계산되기 때문에, 픽셀의 classification 문제가 link보다 더 중요하다. -> λ = 2.0

$$L = \lambda L_{pixel} + L_{link}$$

Pixel Loss

각 픽셀마다 같은 weight를 주게 되면, 크기가 큰 instance에 weight가 집중되는 문제가 발생함.
-> 각 pixel의 weight는 해당 instance의 넓이에 반비례하는 Loss함수를 가짐.

$$L_{pixel} = \frac{1}{(1+r)S} W L_{pixel\_CE},$$

Link Loss

W: 위의 positive pixel과 negative pixel의 weight matrix / Ylink: link의 label matrix

$$W_{pos\_link}(i,j,k) = W(i,j) * (Y_{link}(i,j,k) == 1)$$
$$W_{neg\_link}(i,j,k) = W(i,j) * (Y_{link}(i,j,k) == 0)$$

L_link_CE: link prediction에 대한 Cross Entropy Loss matrix

$$L_{link\_pos} = W_{pos\_link} L_{link\_CE},$$
$$L_{link\_neg} = W_{neg\_link} L_{link\_CE},$$

rsum: reduce sum으로 tensor의 모든 요소를 scalar로 더한 것

$$L_{link} = \frac{L_{link\_pos}}{rsum(W_{pos\_link})} + \frac{L_{link\_neg}}{rsum(W_{neg\_link})}$$

Text Recognition

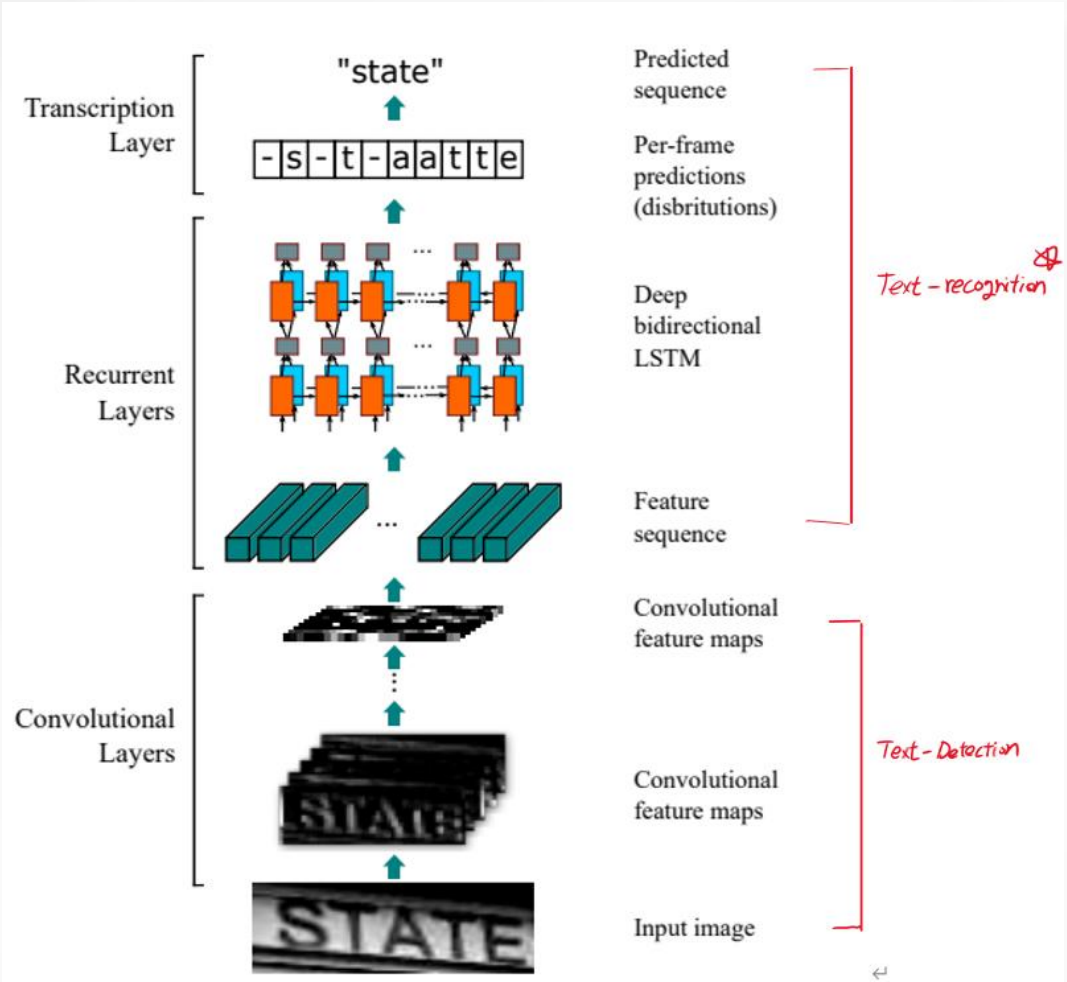
CRNN

CRNN 개요

1. Convolution Layer (CNN)
   feature map(bounding box)를 feature sequence로 분할

2. Recurrent Layer (RNN)
   Bi-LSTM으로 feature sequence의 레이블 분포 예측

3. Transcription Layer (CTC)
   Recurrent layer의 출력 sequence를 최종 label sequence로 반환



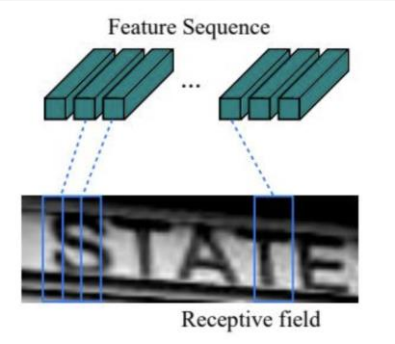
Text Recognition

CRNN

Network Architecture

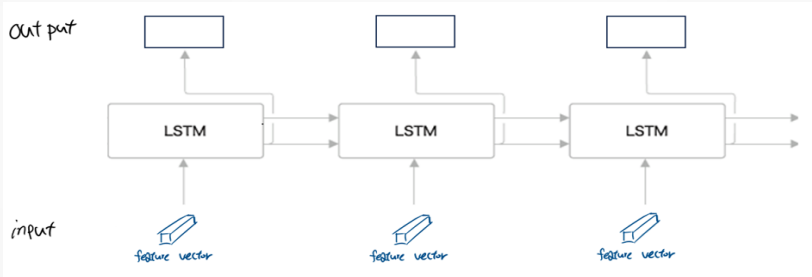
Recurrent Layers

- Feature sequence
  i번째 feature vector는 feature map의 i번째 column이다.



*feature sequence

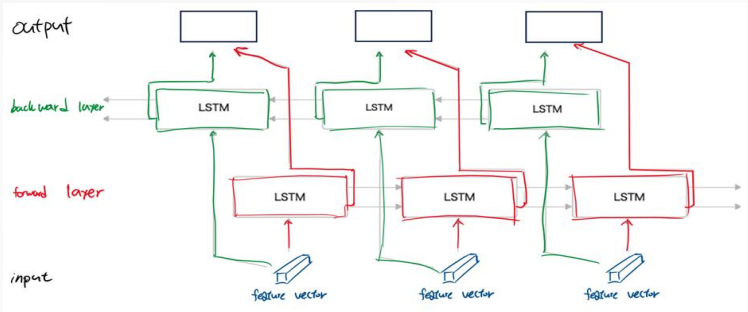
Basic LSTM



- Bi-LSTM(bidirectional LSTM)
  시퀀스를 왼쪽→오른쪽(Forward)과 오른쪽→왼쪽(Backward) 두 방향으로 처리

  두 방향의 출력을 결합해 앞뒤 문맥 정보를 모두 반영 -
  ->반복 문자나 비슷한 글자 구분 정확도 상승

Bi-LSTM



Text Recognition

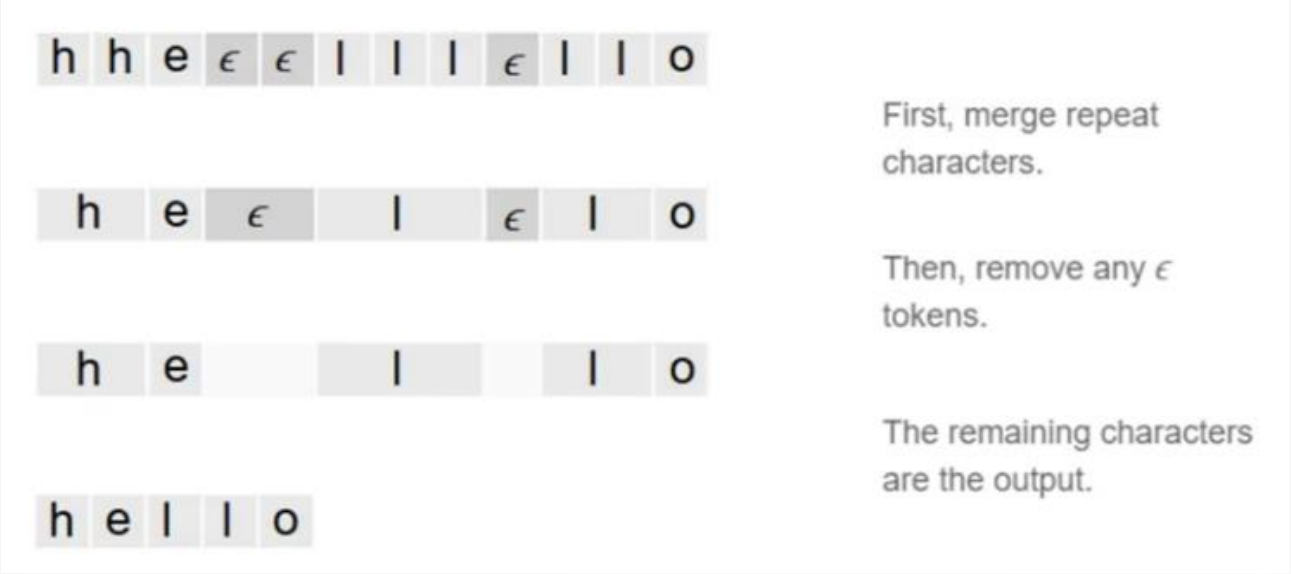
CRNN

Network Architecture

Transcription Layers

- CTC : 출력의 정렬 정보가 없는 상황에서 사용되는 손실 함수

RNN의 출력 시퀀스 -> 최종 문자열을 생성



향후 발전 방향

3-1 Trending

3-2 Challenging problems

Trending

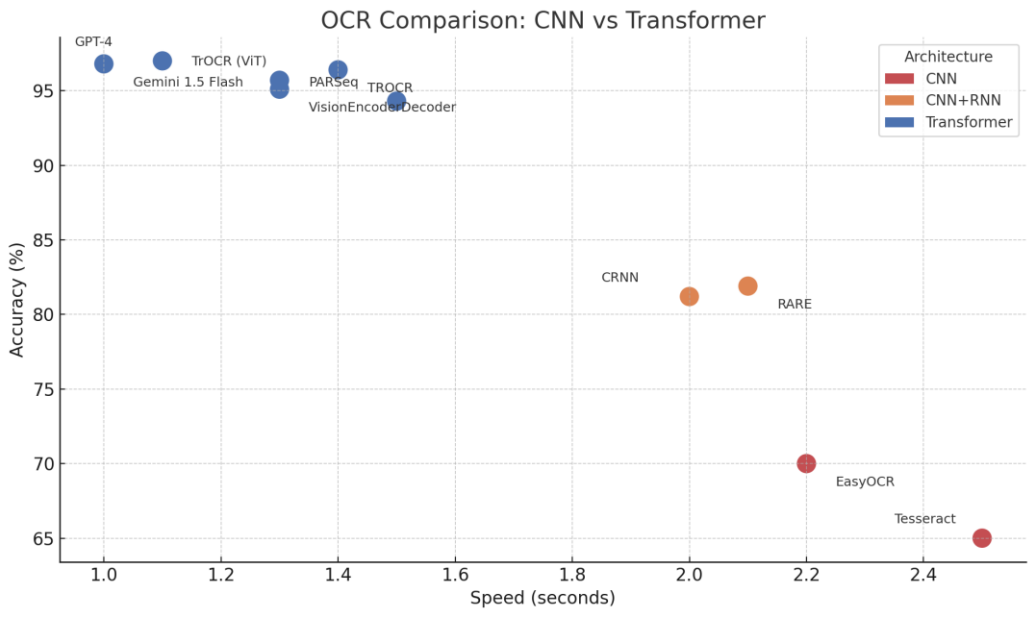
CNN 기반 모델의 한계

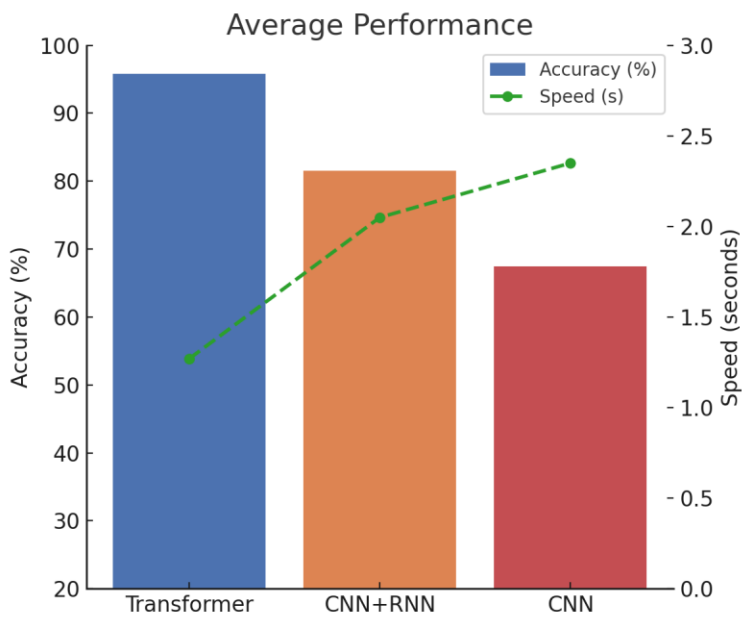
CRNN :(

정확도 ↓ 처리 속도 ↑

TRANSFORMER :)

정확도 ↑ 처리 속도 ↓





Trending

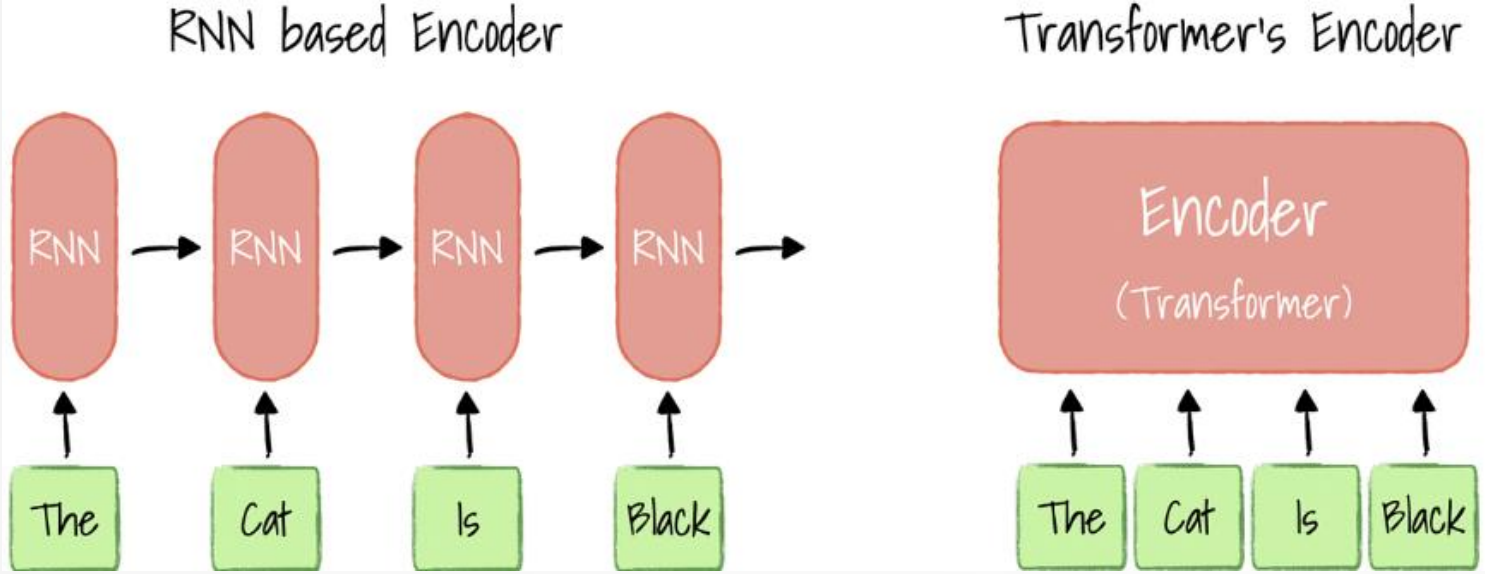
Transformer

RNN

순차적(sequential) 처리 -> 속도 저하
gradient vanishing -> 긴 문장의 앞뒤 정보 연결의 불안정성

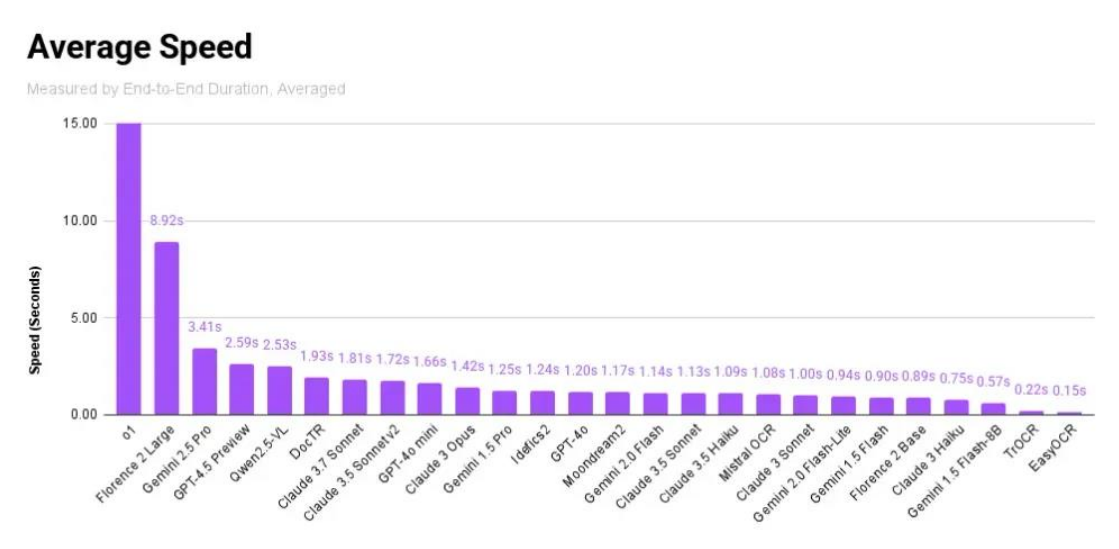
TRANSFORMER

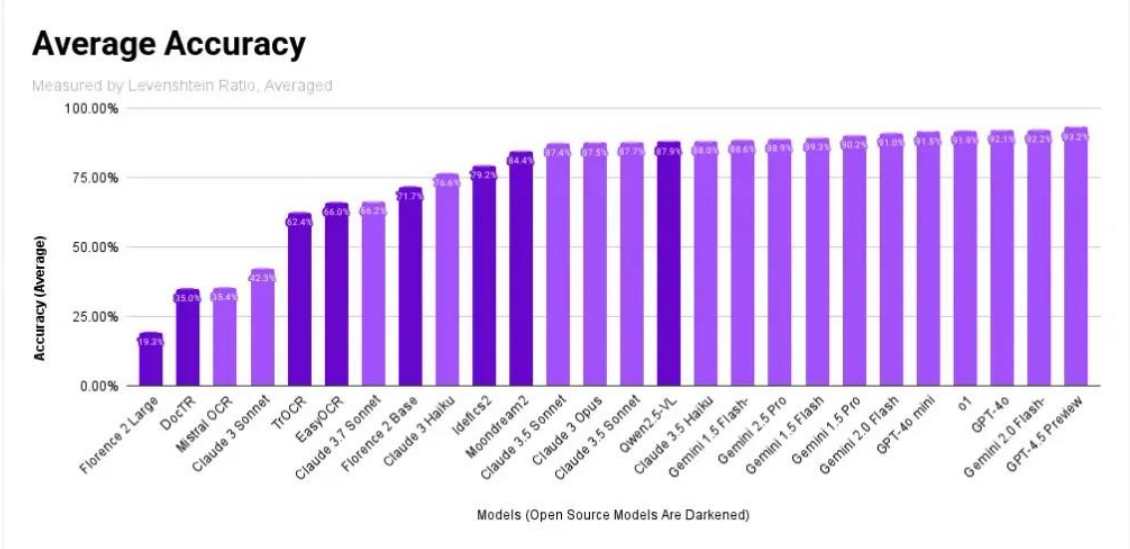
병렬 처리 -> 속도 상승
self-attention -> 모든 위치의 정보를 한 번에 고려



Trending

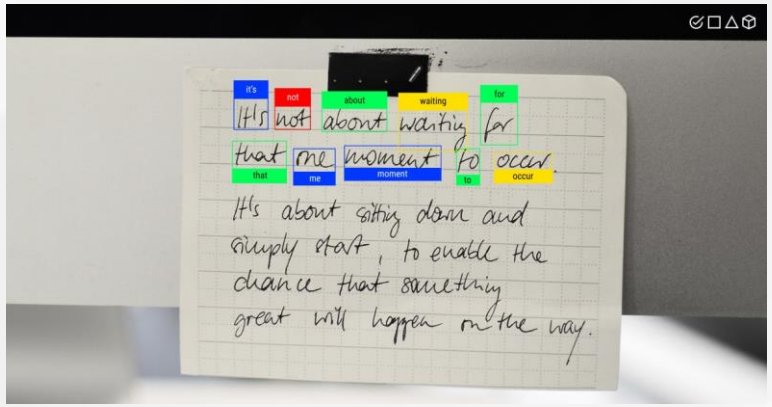
Trending Models (based on transformer)





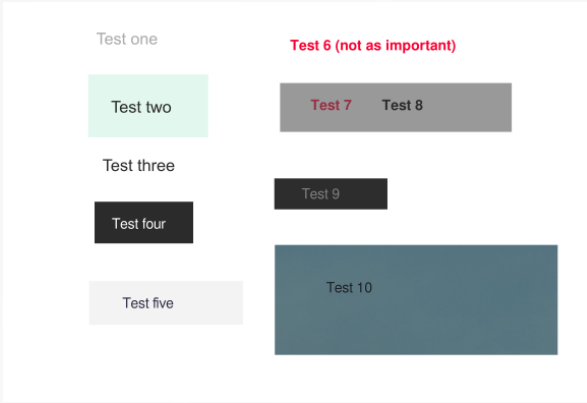
OpenAI – ChatGPT, Google - Gemini

Challenging problems

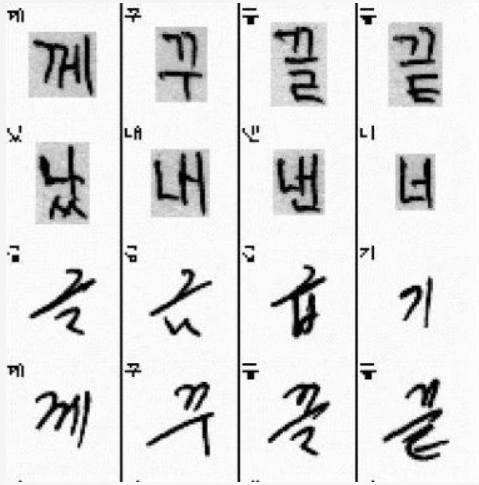


*handwritten character

*special symbols



*colored background



*language limitation

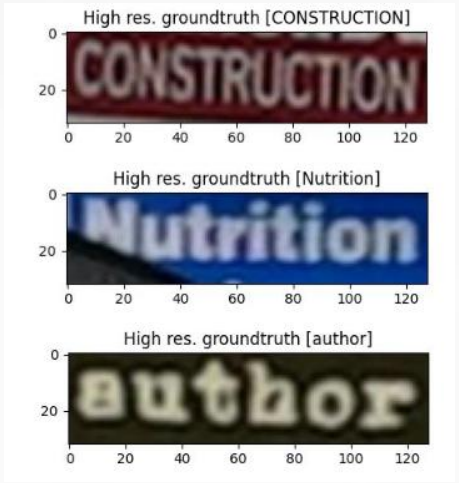


*image quality issue

THANKS!