EAST: An Efficient and Accurate Scene Text Detector

Abstract

이전까지의 Scene text detection의 접근 방식은 다양한 벤치마크에서 유망한 성능을 달성했습니다. 하지만 전체적인 성능은 파이프라인의 여러 단계와 구성 요소의 상호 작용에 의해 결정되기 때문에 Deep Nerual Network를 이용했음에도 불구하고 어려운 senario를 처리하기에는 부족합니다. 본 연구에서 우리는 Scene text detection에서 빠르고 정확한 text detection을 산출하는 간단하면서도 강력한 파이프라인을 제안합니다. 이 파이프라인은 전체 이미제에서 임의의 방향과 각도를 가지는 단어 또는 텍스트 라인을 직접 예측하여 단일 신경망으로 불필요한 중간 단계를 제거합니다. 파이프라인의 단순성으로 loss function과 neural architecture 설계에 집중할 수 있었고 좋은 성능을 내었습니다.

▼ 정리

불필요한 중간단계를 생략한 우리 EAST 모델 최고

1. Introduction

최근 nartural scenes에서는 구현된 텍스트 정보를 추출하고 이해하는 것이 점점 중요해지고 있습니다.

Text imformation을 extraction하고 understanding하는 것에서 Text detection는 중요한 과정 중 하나입니다. 이전까지의 Text detection접근 방식은 이미 이 분야의 다양한 벤치마크에서 좋은 성능이 나왔음이 입증되었습니다.

Text Detection의 핵심은 Text와 배경을 구별하는 것입니다. 전통적으로 Feature들은 scene text로부터 caputure되도록 수동적으로 설계되지만, 딥러닝 기반 방법은 effective feature들이 training data로부터 직접 학습됩니다.

그러나 기존 방법은 conventional(구형 알고리즘) or deep neural network기반으로 구성되며, 대부분 몇 가지 단계와 구성 요소로 구성되고, 이것은 아마 sub-optimal하고 time-comsuming할 것입니다. 그러므로 이전까지의 방법론들은 정확도와 효율성이 여전히 만족스럽지 못합니다.

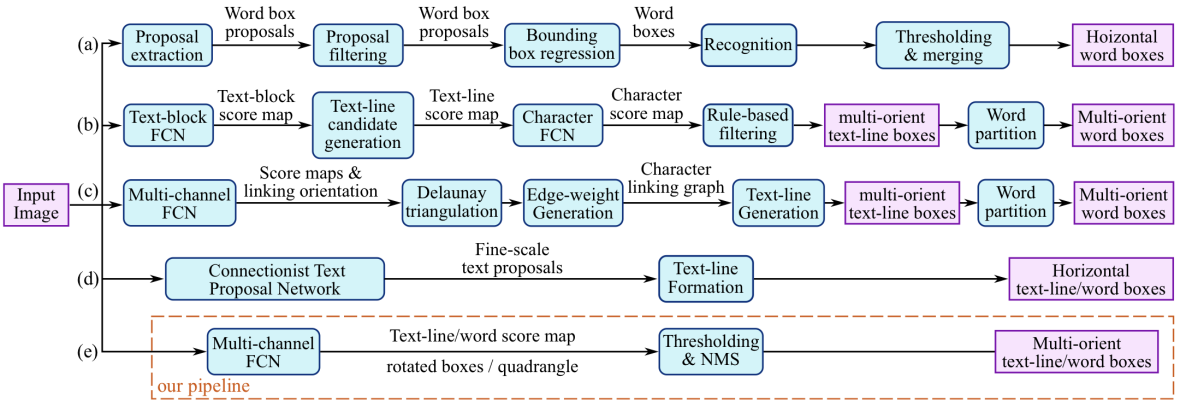
본 논문에서는 **two stage만 있는 빠르고 정확한 Scene text detection pipeline**을 제안합니다. pipeline은 단어 또는 텍스트 라인 수준의 얘측을 직접 생성하고, redundant하고 slow intermediate step이 제외된 FCN model을 사용합니다. 표준 벤치마크에 대한 정성적 및 정량적 실험에 따르면 제안된 알고리즘은 기존의 방법론에 비해 훨씬 빠르게 실행되면서 성능이 크게 향상되었습니다.

13.2fps at 720p resolution on a Titan-X GPU for our best performing model, 16.8fps for our fastest model

이 논문의 contribution은 3개의 단계가 있습니다.

- 우리는 FCN와 NMS merging이라는 2개의 stage scene text detection method를 제안했습니다. FCN는 중복되고 시간이 많이 걸리는 중간 단계를 제외했고, text regions을 direct하게 생성합니다.
- Pipeline은 특정 application에 따라 geometric shape이 rotated box혹은 일반적인 상자의 형태를 띈 단어 수준 또는 문장(line)수준의 예측을 생성이 가능하도록 유연합니다.
- 제안된 Algorithm들은 정확성과 속도 모두에서 SOTA를 크게 능가합니다.

2. Related Work



기존 방법론 보다 DNN 기반의 방법론들이 훨씬 성능이 좋다. 특히 Task가 복잡해지는 경우 더 좋았다.

이전 논문 중에서는 수평 텍스트 라인을 감지하기 위한 CNN-RNN joint model을 구성한 경우도 있었습니다.(CRNN) FCN을 이용한 경우도 있었습니다.

위의 Figure를 보듯이 대부분의 경우 Multi-stage한 경우들이 많습니다. 전처리나 candidate aggregation을 하면서 false positive를 삭제하는 방향으로 진행했습니다.(Multi stage의 이유)

Multitude of stages혹은 componenets들은 대부분 training에 exhaustive tuning이라는 문제가 발생합니다.

본 논문에서는 FCN을 기반으로한 text detection pipeline을 제안합니다. 그래서 Two Stage로 단계를 단축해 학습 pipeline을 간략화 함과 동시에 성능을 높였습니다.

3. Methodolgy

우리의 모델은 text Detection에 adapted된 FCN으로 output은 dense pixel prediction of words or text lines입니다.

Post-processing step에서는 thresholding을 준 NMS와 geometric shapes을 예측하도록 합니다. 이 Detector의 이름은 EAST이고, Efficient and Accuracy Scene Text detection pipeline입니다.

3.1 Pipeline

본 모델의 전체적인 pipeline은 여기(Figure 2)에 자세히 나와있으며, input image를 multiple channels of piexel-level text score map과 geometry가 생성되도록하는 FCN으로 넣습니다

predicted channel 중 하나는 score map입니다. 이것은 pixel value가 [0, 1]입니다. 나머지 채널은 각 픽셀에서 봤을 때, 단어를 둘러싸는 geometrical한 정보들이 담겨있습니다. 해당

점수들은 동일한 위치에서 예측된 형상 모형의 신뢰도를 나타냅니다.

텍스트 영역에 대한 두 가지 기하학적 모양인 회전 상자(RBOX)와 사각형(QUAD)을 실험하고 각 지오메트리마다 다른 손실 함수를 설계했다. 그런 다음 각 예측 영역에 임계값이 적용되며, 여기서 점수가 미리 정의된 임계값을 초과하는 예측값은 유효한 것으로 간주되고, 나중에 NMS을 위해 저장됩니다. NMS 이후의 결과를 파이프라인의 최종 출력으로 간주됩니다.

3.2 Network Design

텍스트 감지를 위한 신경망을 설계할 때는 몇 가지 요소를 고려해야 한다. 그림 5와 같이 단어 영역의 크기는 매우 다양하기 때문에, 큰 단어의 존재를 결정하는 것은 신경망의 후기 단계의 Feature를 필요로 하는 반면, 작은 단어 영역을 둘러싼 정확한 기하학을 예측하는 것은 초기 단계의 low level의 Feature 정보가 필요하다. 그러므로 **Networks는 앞서 나온 requirements를 충족하기 위해 서로 다른 level의 Feature를 사용해야만 합니다**. HyperNet은 앞서나온 조건들을 충족하지만 large feature map들을에서 많은 수의 Channel을 병합하면 이후 단계에 대한 계산 오버헤드가 크게 증가할 것입니다.

이를 해결하기 위해 U자형 아이디어를 채택하여 상향 샘플링 브랜치를 작게 유지하면서 Feature map을 점진적으로 병합합니다. 우리는 다양한 level의 feature를 활용하고 적은 계산 비용을 유지할 수 있는 network로 종결됩니다.

우리의 모델에 대한 개략적 그림 3에 그려져 있다. 그 모델은 Feature extraction system, feature-meraging, 그리고 output layer로 구성되어 있다. 4 level of feature map, f_i로 표시된 것들은 extraction layer들이다. input image의 $\frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ and $\frac{1}{4}$각각 다음과 같은 size입니다. 우리는 VGG16모델도 적용해 poolling-2 ~ pooling5를 이용해 extraction을 해보았습니다.

$$g_i = \begin{cases} unpool(h_i) & \text{if } i \leq 3 \\ conv_{3\times 3}(h_i) & \text{if } i = 4 \end{cases}$$

$$h_i = \begin{cases} f_i & \text{if } i = 1 \\ conv_{3\times 3}\big(conv_{1\times 1}([g_{i-1}; f_i]) & otherwise \end{cases}$$

unpooling operation는 maxpooling의 반대라고 생각하면 되고, 이것은 merge의 기본(base)가 됩니다. h_i는 channel wise하게 feature map을 합치게되고, 마지막 Feature map은 먼저 Unpooling layer로 Fed되어 크기를 2배로 한 다음에 현재의 Feature map과 연결됩니다. 다음으로, Conv_{1x1}로 bottlneck형태로 만들어 channel reduction을 수행하고, 계산을 줄이며, 정보를 융합하여 최종적으로 meging 단계의 output을 생성합니다. 마지막 병합 layer를 거쳐 conv_{3x3} layer는 마지막 Feature map을 생성하고, output layer의 input으로 들어가게 됩니다.

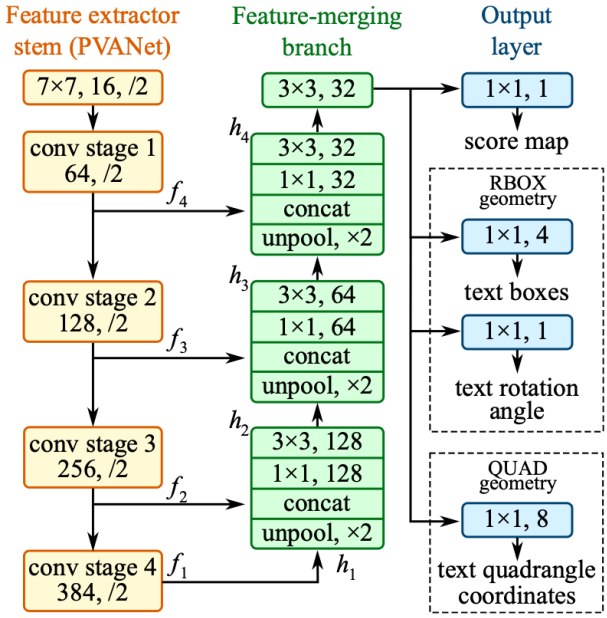


Figure 3. Structure of our text detection FCN.

각 convolution채널의 output들은 Figure 3에 있다. 우리는 branch에서의 convolution별 channel 수를 적게 유지하며, 각 stem에 대한 compuatation overhead를 일부만 추가해 computation-efficient를 이끌어냈습니다. 최종 출력 레이어는 32채널의 Feature map을 1채널의 score map F_s와 멀티채널 Geometry map F_g로 투영하는 여러 $Conv_{1\times1}$연산을 포함한다. shape 출력은 table1에 요약된 RBox 또는 QUAD 중 하나일 수 있습니다.

RBOX의 경우 Geometry는 축 align bounding box의 AABB(R)의 채널과 1개의 회전각도 θ로 표시됩니다. R의 공식은 아래의 논문의 공식과 동일합니다. **여기서 4채널은 픽셀 위치에서 직사각형의 상단, 오른쪽, 왼쪽 boundaries까지의 거리를 나타냅니다.**

Geometry	channels	description
AABB	4	$\mathbf{G} = \mathbf{R} = \{d_i \vert i \in \{1,2,3,4\}\}$
RBOX	5	$\mathbf{G} = \{\mathbf{R}, \theta\}$
QUAD	8	$\mathbf{G} = \mathbf{Q} = \{(\Delta x_i, \Delta y_i) \vert i \in \{1,2,3,4\}\}$

Table 1. Output geometry design

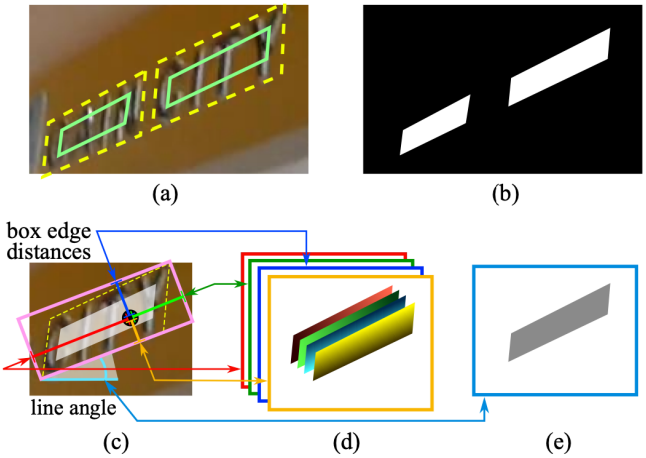


Figure 4. Label generation process: (a) Text quadrangle (yellow dashed) and the shrunk quadrangle (green solid); (b) Text score map; (c) RBOX geometry map generation; (d) 4 channels of distances of each pixel to rectangle boundaries; (e) Rotation angle.

QUAD Q의 경우 8개의 숫자를 이용하여 사각형의 네 모서리 꼭지점 $\{p_i \vert \in \{1,2,3,4\}\}$에서 각 픽셀 위치로 좌표로의 이동$(\Delta x, \Delta y)$를 의미합니다.

3.3 Label Generation

3.3.1 Score Map Generation for Quadrangle

Generality를 잃지 않고, 우리는 geometry가 quadrangle인 것들만 고려하고 있습니다. Figure (4)에서 그려진대로 Score map 상의 사각형의 positive area는 원래의 사각형의 축소판으로 대략 설계됩니다. 정사각형 $Q = \{q_i \vert i \in \{1,2,3,4\}\}$의 경우, $p_i = \{x_i, y_i\}$는 시계 방향으로 정사각형에 있는 꼭지점입니다.

Q를 shrink(줄이기, 간격을 줄이는 뜻인듯?), 우리는 먼저 꼭지점 p_i에 대한 기준 r_i를 계산합니다. 여기서 $D(p_i, p_j)$는 p_i와 p_j 사이의 $L2$ distance입니다.

우리는 먼저 사각형의 긴 두 모서리를 축소하고, 그 다음에 더 짧은 모서리를 축소합니다. 마주보는 두 모서리의 각 쌍에 대해길이의 평균을 비교하여 "더 긴" 쌍을 결정합니다. 각 에지($\langle p_i, p_{(i \bmod 4)+1} \rangle$에 대해, 우리는 그것의 두 끝점을 따라 안쪽으로 $0.3r_i$와 $0.3r_{(i \bmod 4)+1}$이동함으로써 그것을 축소합니다.

3.3.2 Geometry Map Generation

우리는 3.2에서 geometry map이 RBOX 혹은 QUAD에 대해서 얘기했습니다. Generation process for RBOX는 Figure 4에 나와있습니다.

텍스트 영역이 QUAD 스타일로 주석이 다린 Dataset인 경우, 먼저 최소 먼적으로 영역을 Cover한 Rotated Rectangle형태로 생성합니다. 그런 다음 Positive Score를 가진 각 픽셀에 대해 Text box의 4개의 boundaries까지의 거리를 계산하고, 해당 값들을 4개의 channel에 RBOX GT로 넣습니다. QUAD의 GT의 경우, 8channel geometry map에서 positive score를 가진 각 픽셀의 값은 4개의 꼭지점으로부터의 coordinate shift(좌표 이동)입니다.

3.4 Loss Functions

loss는 다음과 같이 공식화 할 수 있습니다.

$$L = L_s + \lambda_g L_g$$

L_s와 L_g는 score map과 geometry에 대한 loss를 represent하고, λ_g는 둘 중 중요한 것에 weighted 두는 가중치 입니다.우리의 실험에서는 λ_g는 1로 적용했습니다.

3.4.1 Loss for Score Map

대부분의 SOTA detection pipeline은 balanced sampling되고, target object에 대한 imbalanced distribution에 tackle을 하기 위해 hard negative mining을 진행한다.이렇게하면 네트워크의 성능이 향상될 수 있습니다. 그러나 이러한 기술을 사용하면 “필연적으로” 미분이 불가능한 단계와 튜닝할 더 많은 매개 벼수와 더 복잡한 파이프라인도 도입되며, 이는 우리의 설계 원칙과 모순됩니다.

간단한 훈련 절차를 위해, 우리는 아래와 같은 class balanced cross-entropy를 사용한다.

$$\begin{aligned} L_s &= \text{balanced-xent}(\hat{Y}, Y^*) \\ &= -\beta Y^* \log \hat{Y} - (1-\beta)(1-Y^*)\log(1-\hat{Y}) \end{aligned}$$

$\hat{Y} = F_s$일 때 prediction of the score map이고, Y^*이 GT입니다. β는 positive와 negative사이의 balancing factor이고 다음과 같습니다.

$$\beta = 1 - \frac{\sum_{y^* in Y^*} y*}{|Y^*|}$$

이러한 balanced cross-entropy loss는 text detection Task의 score map prediction의 objective function으로써 처음 적용되었습니다.

3.4.2 Loss for Geometries

텍스트 감지의 한 가지 과제는 natrual scene image에서의 text들의 size가 굉장히 tremendously(dynamic)합니다. 그저 L_1 혹은 L_2 Loss를 regression을 위해 directly하게 사용하는 것은 larget 그리고 longer text region에 대해 loss bias를 야기합니다. 그래서 우

리는 accurate text geometry prediction을 위해 새로운 loss를 생각해야했습니다. regression인 scale-invariant인 loss여야 합니다. 그래서 RBOX 형태의 AABB part에서 IOU loss를 적용했습니다. 그리고 scale-normalized smoothed-L1 loss를 QUAD regression에 적용했습니다.

$$L_{AABB} = -\log \text{IoU}(\hat{R}, R) = -\log \frac{|\hat{R} \cap R^*|}{|\hat{R} \cup R^*|}$$

여기서 $\hat{R}$은 AABB의 predicted geometry이고, R^*는 GT입니다.

$$L_\theta = (\hat{\theta}, \theta^*) = 1 - \cos(\hat{\theta} - \theta^*)$$

여기서 $\hat{\theta}$은 prediction한 rotated angle이고, θ^*는 GT입니다.

$$L_g = L_{AABB} + \lambda_\theta L_\theta$$

여기서 λ_θ는 10입니다.

▼ 위 loss에 대한 제 생각

각도에 대한 weight가 10정도로 엄청 크게 잡혀 있는 것으로 보아 “각도”가 먼저 잡혀진 뒤 혹은 parameterer들이 “각도”를 맞출 수 있는 범위 내로 들어온 뒤에 backprob을 통해 세부적인 내용을 수정하는 형태로도 생각할 수 있습니다.

질문

▼ 여기서 이건 어떻게 구현되는가?

▼ hard negative이란?

- link : https://blog.naver.com/sogangori/221073537958
- hard negetive는 실제로는 negative인데, positive라고 잘못 예측하기 쉬운 데이터들입니다.

그래서 prediction을 진행해 hard negative로 잘못 예측된 이미지들은 “다시” sampling하여 학습 데이터로 사용하는 것입니다.

정확한 pipeline은 모르겠으나 개념적으로 이해하면 좋을 거 같습니다.