Character Region Awareness for Text Detection

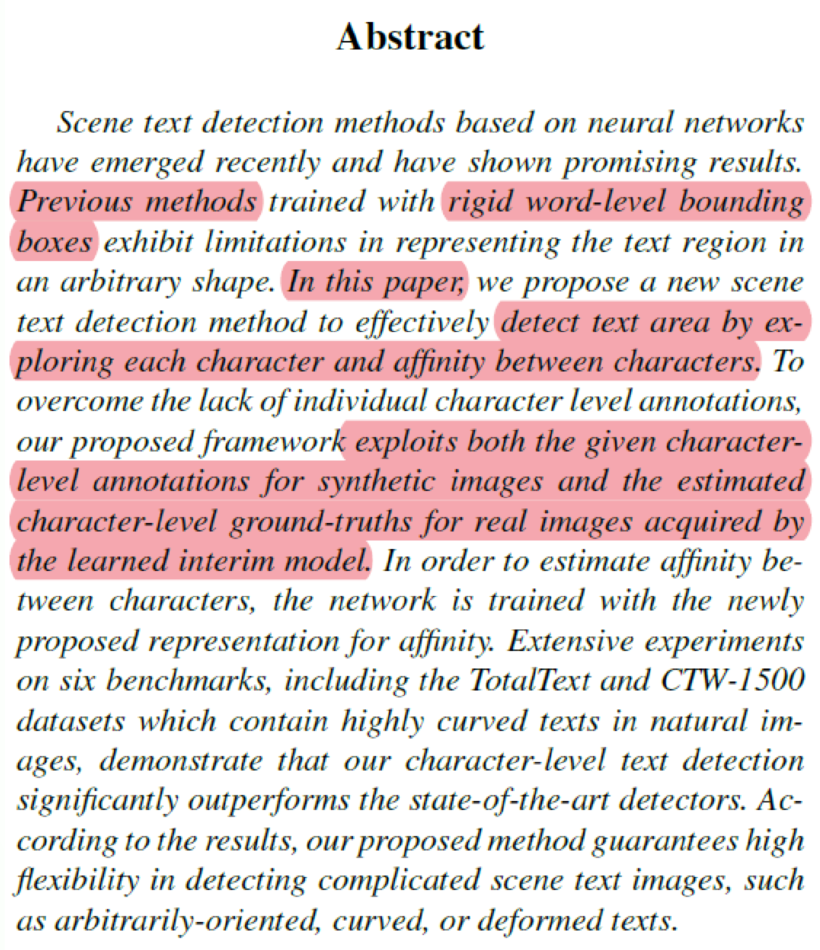
Clova AI Research, NAVER Corp.

INDEX

HYUNHP

OVERVIEW

Abstract



Abstract

Scene text detection methods based on neural networks have emerged recently and have shown promising results. Previous methods trained with rigid word-level bounding boxes exhibit limitations in representing the text region in an arbitrary shape. In this paper, we propose a new scene text detection method to effectively detect text area by exploring each character and affinity between characters. To overcome the lack of individual character level annotations, our proposed framework exploits both the given character-level annotations for synthetic images and the estimated character-level ground-truths for real images acquired by the learned interim model. In order to estimate affinity between characters, the network is trained with the newly proposed representation for affinity. Extensive experiments on six benchmarks, including the TotalText and CTW-1500 datasets which contain highly curved texts in natural images, demonstrate that our character-level text detection significantly outperforms the state-of-the-art detectors. According to the results, our proposed method guarantees high flexibility in detecting complicated scene text images, such as arbitrarily-oriented, curved, or deformed texts.

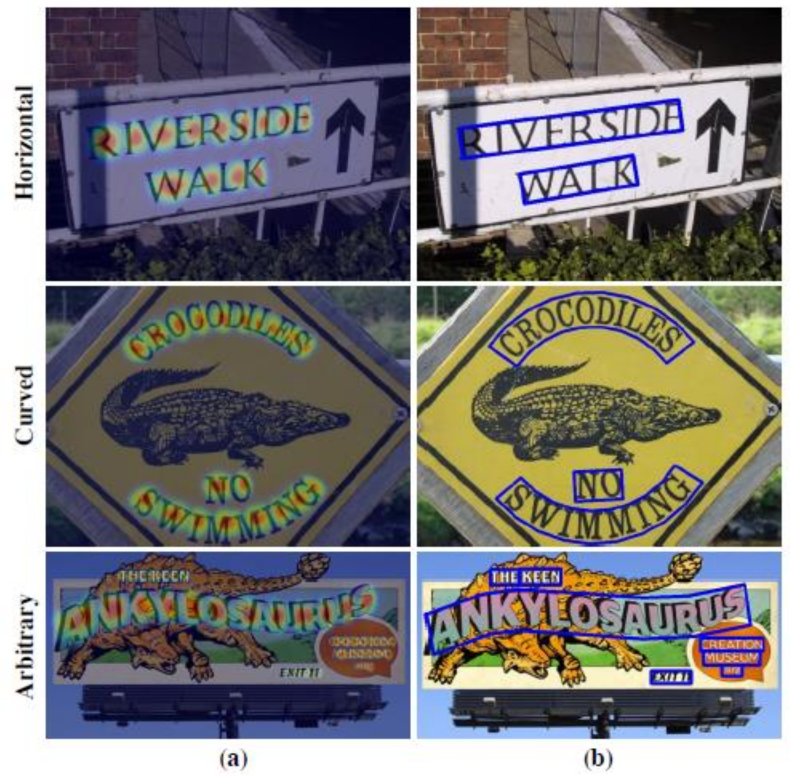
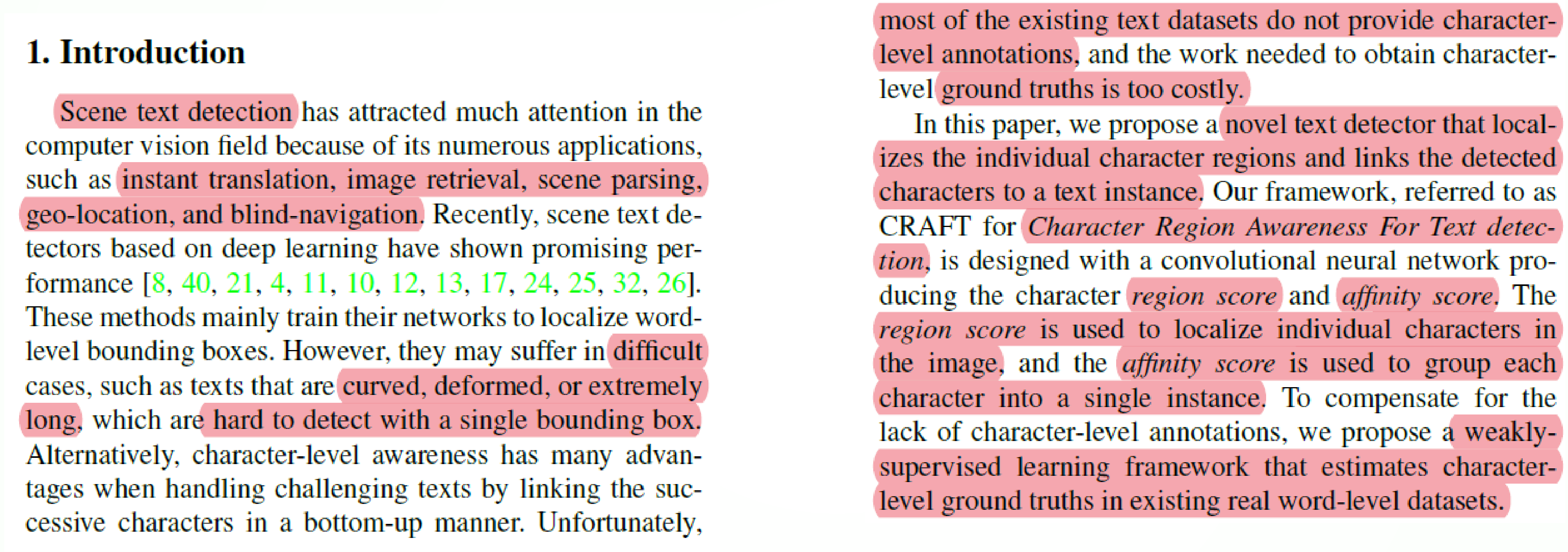


Figure 1. Visualization of character-level detection using CRAFT. (a) Heatmaps predicted by our proposed framework. (b) Detection results for texts of various shape.

What is OCR?

- **OCR stands for Optical Character Recognition.** It is a computer vision technology that is used to recognize and extract text information from images, such as scanned documents, photographs, and screenshots.
- OCR works by analyzing the patterns and shapes of characters in the image, and then using algorithms to translate those patterns into machine-readable text. The process of OCR typically involves several steps, such as **image pre-processing, segmentation, feature extraction, classification, and post-processing.**

INTRODUCTION



1. Introduction

Scene text detection has attracted much attention in the computer vision field because of its numerous applications, such as instant translation, image retrieval, scene parsing, geo-location, and blind-navigation. Recently, scene text detectors based on deep learning have shown promising performance [8, 40, 21, 4, 11, 10, 12, 13, 17, 24, 25, 32, 26]. These methods mainly train their networks to localize word-level bounding boxes. However, they may suffer in difficult cases, such as texts that are curved, deformed, or extremely long, which are hard to detect with a single bounding box. Alternatively, character-level awareness has many advantages when handling challenging texts by linking the successive characters in a bottom-up manner. Unfortunately,

most of the existing text datasets do not provide character-level annotations, and the work needed to obtain character-level ground truths is too costly.

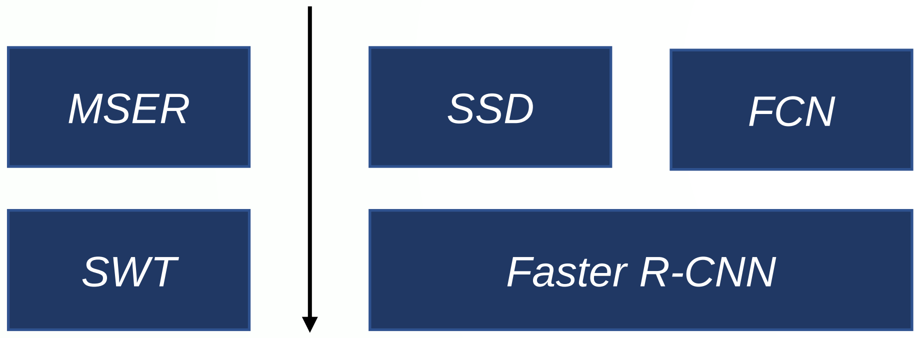
In this paper, we propose a novel text detector that localizes the individual character regions and links the detected characters to a text instance. Our framework, referred to as CRAFT for *Character Region Awareness For Text detection*, is designed with a convolutional neural network producing the character *region score* and *affinity score*. The *region score* is used to localize individual characters in the image, and the *affinity score* is used to group each character into a single instance. To compensate for the lack of character-level annotations, we propose a weakly-supervised learning framework that estimates character-level ground truths in existing real word-level datasets.

RELATED WORKS

Related text detector works

Major trend in scene text detection

Emergence of deep learning



Research fields in scene text detection

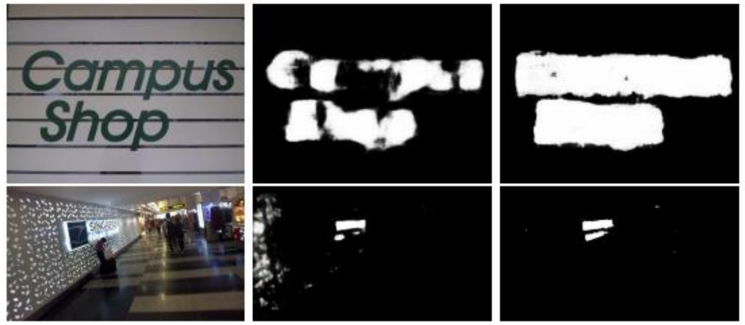
- *Regression-based text detectors*
- *Segmentation-based text detectors*
- *End-to-end text detectors*
- *Character-level text detectors*

Research fields in scene text detection (1)

- **Regression-based text detectors**
  - Various text detectors using **box regression** adapted from object detectors have been proposed.
  - e.g., TextBoxex; DMPNet; Rotation-Sensitive Regression Detector (RSDD)

Ref. https://inlocrobotics.com/

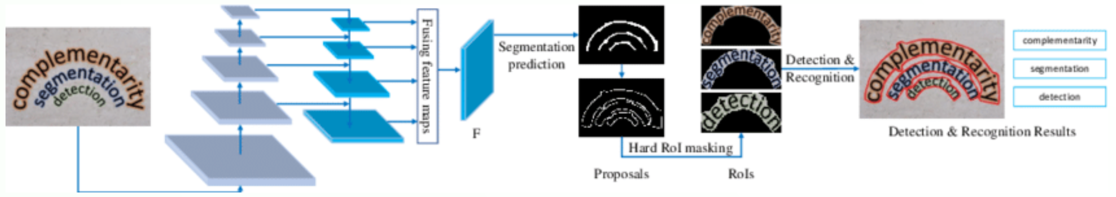
- **Segmentation-based text detectors**
  - Another common approach is based on works dealing with segmentation, which aims to **seek text regions at the pixel level.**
  - e.g., Multi-scale FCN; Holistic-prediction; PixelLink; SSTD; TextSnake



Ref. Multi-scale FCN with Cascaded Instance Aware Segmentation for Arbitrary Oriented Word Spotting In The Wild

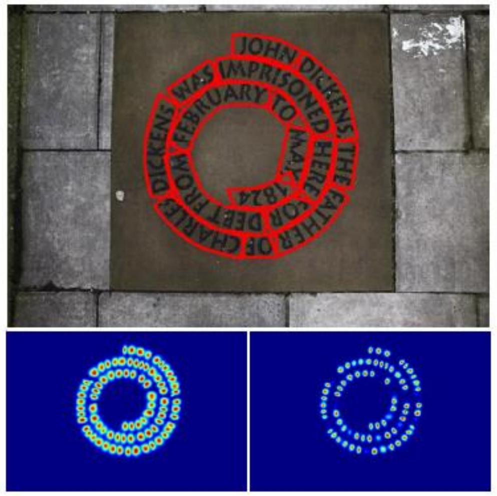
Research fields in scene text detection (2)

- **End-to-end text detectors**
  - Train the detection and recognition modules simultaneously, so as to enhance detection accuracy by leveraging the recognition result.
  - End-to-end model means that input data and no need to preprocess and return output data.
  - Training with the recognition module helps the text detector be robust to text-like background clutters.
  - e.g., FOTs; EAA; Mask TextSpotter



Ref. TextSpotter v3

- **Character-level text detectors**
  - Main objective is to precisely **localize each individual character** in natural images
  - e.g., MSER, WordSup



Ref. CRAFT

METHODOLOGY

Architecture

[Architecture]

- *Backbone*
  - **FCN** on **VGG-16** with batch normalization
- Decoding
  - Skip connections, similar with **U-Net**
- Output
  - Two channels as score maps
    - Region score
    - Affinity score

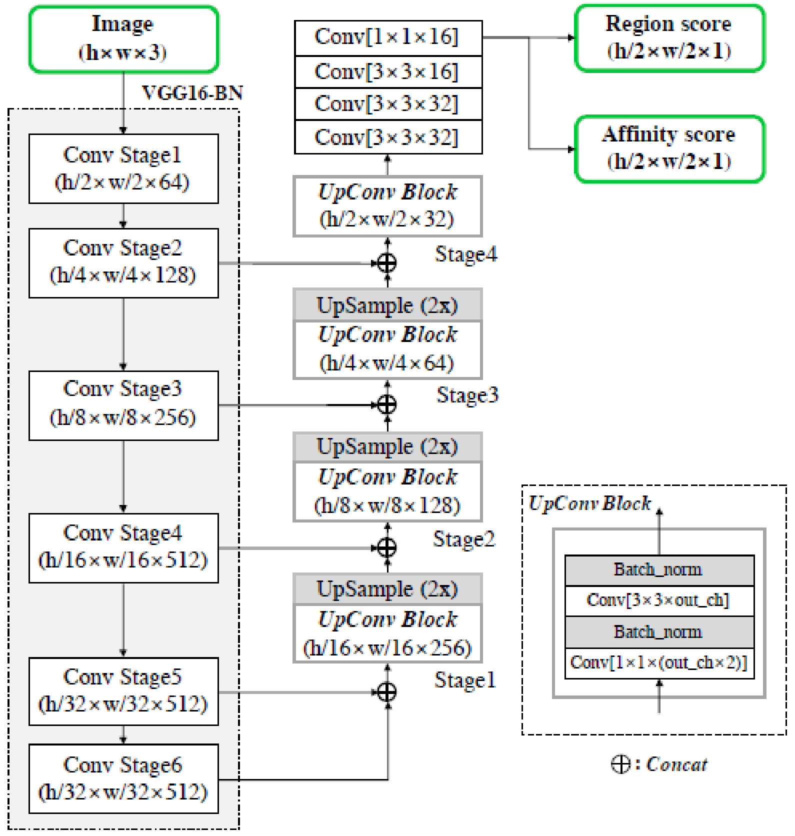


Image (h×w×3)
VGG16-BN
Conv Stage1 (h/2×w/2×64)
Conv Stage2 (h/4×w/4×128)
Conv Stage3 (h/8×w/8×256)
Conv Stage4 (h/16×w/16×512)
Conv Stage5 (h/32×w/32×512)
Conv Stage6 (h/32×w/32×512)
Conv[1×1×16]
Conv[3×3×16]
Conv[3×3×32]
Conv[3×3×32]
Region score (h/2×w/2×1)
Affinity score (h/2×w/2×1)
UpConv Block (h/2×w/2×32)
Stage4
UpSample (2x) UpConv Block (h/4×w/4×64)
Stage3
UpSample (2x) UpConv Block (h/8×w/8×128)
Stage2
UpSample (2x) UpConv Block (h/16×w/16×256)
Stage1
UpConv Block
Batch_norm
Conv[3×3×out_ch]
Batch_norm
Conv[1×1×(out_ch×2)]
⊕: Concat

Figure 2. Schematic illustration of our network architecture.

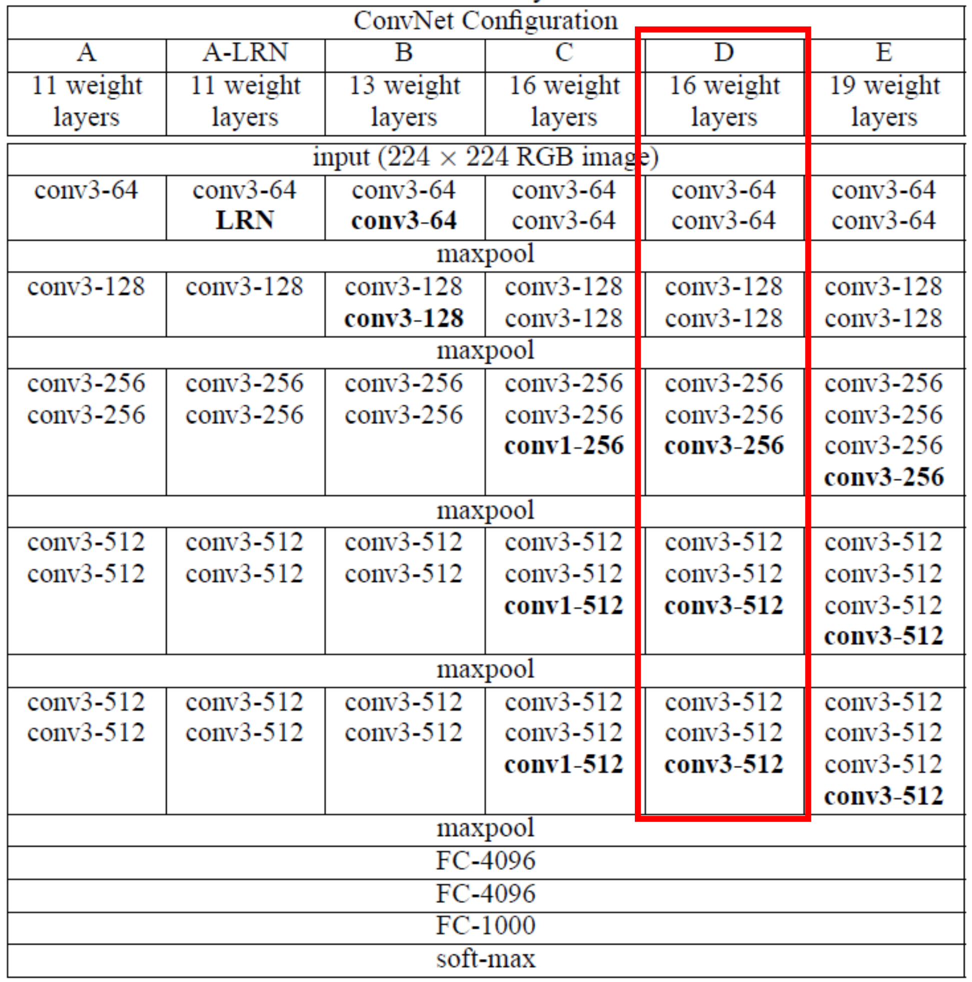
[Backbone] FCN on VGG16

[Architecture]

- *Backbone*
  - **FCN** on **VGG-16** with batch normalization
- *Decoding*
  - Skip connections, similar with **U-Net**
- *Output*
  - Two channels as score maps
    - Region score
    - Affinity score

e.g.,

- **Batch normalization**:
  - Normalize the inputs to each layer by subtracting the mean and dividing by the standard deviation of the inputs.
  - This is done on a batch-by-batch basis during training.

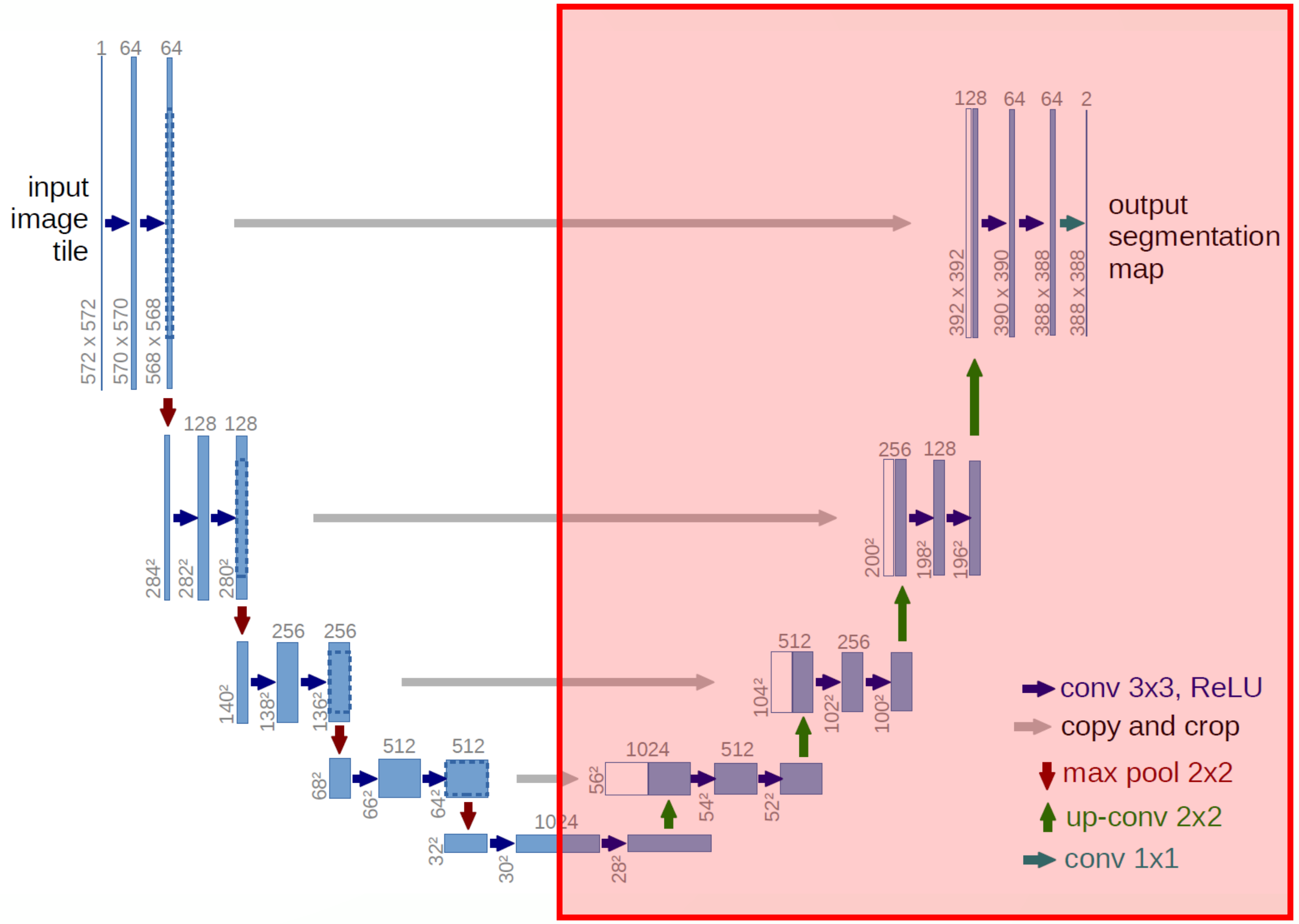


ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

[Decoding] U-Net

[Architecture]

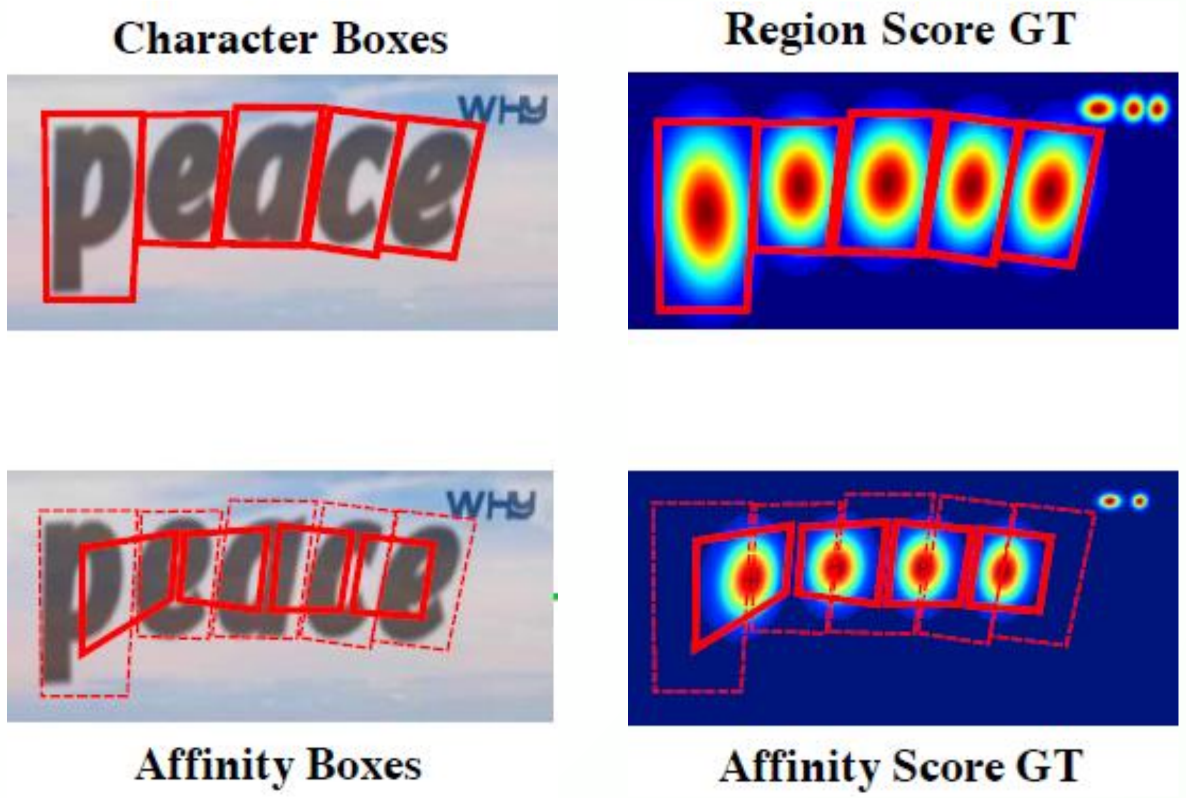
- *Backbone*
  - ***FCN** on **VGG-16** with batch normalization*
- Decoding
  - Skip connections, similar with **U-Net**
- *Output*
  - *Two channels as score maps*
    - *Region score*
    - *Affinity score*



[Output] Region score & Affinity score

[Architecture]

- *Backbone*
  - ***FCN** on **VGG-16** with batch normalization*
- *Decoding*
  - *Skip connections, similar with **U-Net***
- Output
  - Two channels as score maps
    - **Region score**
    - **Affinity score**



[Training] Ground Truth Label Generation

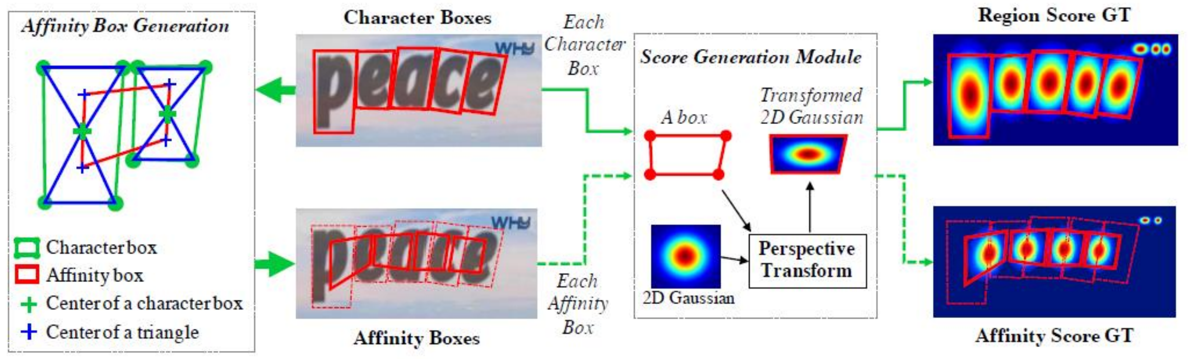


Figure 3. Illustration of ground truth generation procedure in our framework. We generate ground truth labels from a synthetic image that has character level annotations.

- For each training image,

- Generate the **ground truth label** for the **region score** and **the affinity score** with **character level bounding boxes.**
  - **Region score** represents the **probability** that **the given pixel is the center of the character**
  - **Affinity score** represents the **center probability of the space between adjacent characters**
    - Affinity boxes are defined using ***adjacent character boxes***, by drawing diagonal lines to connect opposite corners of each character box, we can *generate two triangles, which refer to as the upper- and lower-character triangles.*
    - Then, for each adjacent character box pair, an **affinity box** is generated by setting the **centers of the upper and lower triangles as corners of the box.**

[Training] Ground Truth Label Generation

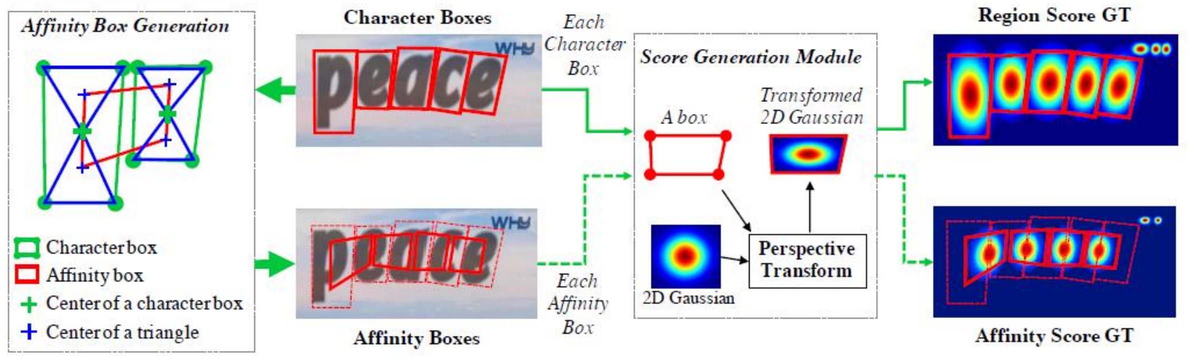


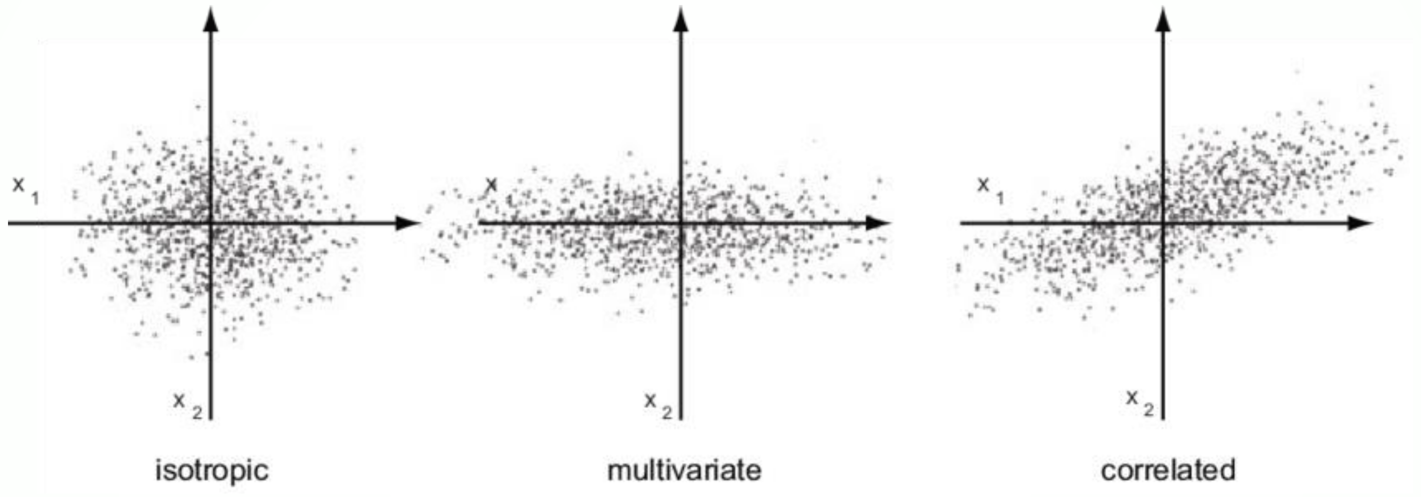
Figure 3. Illustration of ground truth generation procedure in our framework. We generate ground truth labels from a synthetic image that has character level annotations.

- Encode **the probability of the character center with a Gaussian heatmap**.
  - This heatmap representation has been used in other applications, such as pose estimation works due to its high flexibility when dealing with ground truth regions that are not rigidly bounded

- We use the heatmap representation to learn both the region score and the affinity score.
  - **Computing the Gaussian distribution** value directly for **each pixel within bounding box** is **time-consuming**

- To solve, **3 steps to make transformed Gaussian distribution**
  - 1) prepare a 2-dimensional isotropic Gaussian map;
  - 2) compute perspective transform between the Gaussian map region and each character box;
  - 3) warp Gaussian map to the box area.

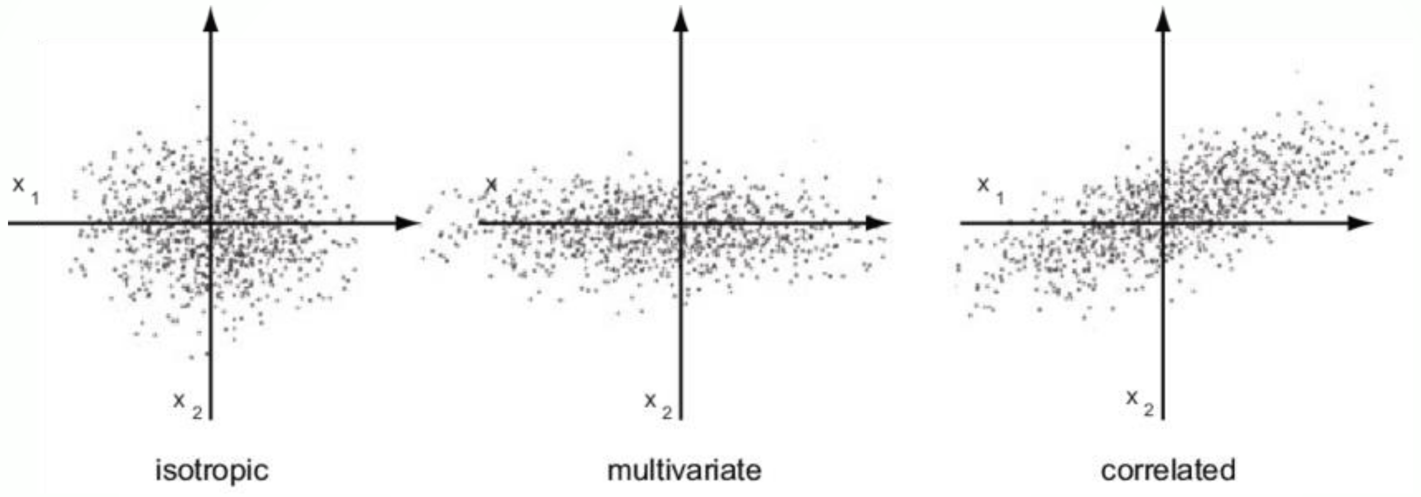
Cf. Isotropic Gaussian Map (1)



- In mathematics and physics, **isotropy** refers to the property of being **invariant under rotation or orientation.**
- **An isotropic Gaussian map**, therefore, is a Gaussian map that has the **same standard deviation or spread** in **all directions and is rotationally symmetric around its center**. In other words, the isotropic Gaussian map has the **same probability density in all directions around its mean.**

- The isotropic Gaussian map is often used in image processing and computer vision applications where rotational symmetry is desirable. For example, in edge detection or corner detection, the response of a filter or a feature detector should be **invariant to the orientation of the image**. By using an isotropic Gaussian filter, the filter's response will be the same regardless of the orientation of the image.

Cf. Isotropic Gaussian Map (2)



- **Invariance to the orientation of an image** means that a particular image processing algorithm or feature detector produces **the same result regardless of the orientation of the image**. In other words, if we rotate the image, the output of the algorithm should be the same as if we had not rotated the image.

- For example, consider the task of detecting edges in an image. An edge is a sharp transition in intensity between adjacent pixels. We can use various edge detection algorithms, such as the Sobel operator or the Canny edge detector, to find the locations of edges in an image. However, the response of these algorithms can depend on the orientation of the image. If we rotate the image, the locations of the edges will change, and the response of the algorithm may also change.

[Training] Weakly-Supervised Learning

- Unlike synthetic datasets, real images in a dataset usually have **word-level annotations.**
- Generate character boxes from each word-level annotation in a **weakly-supervised manner**

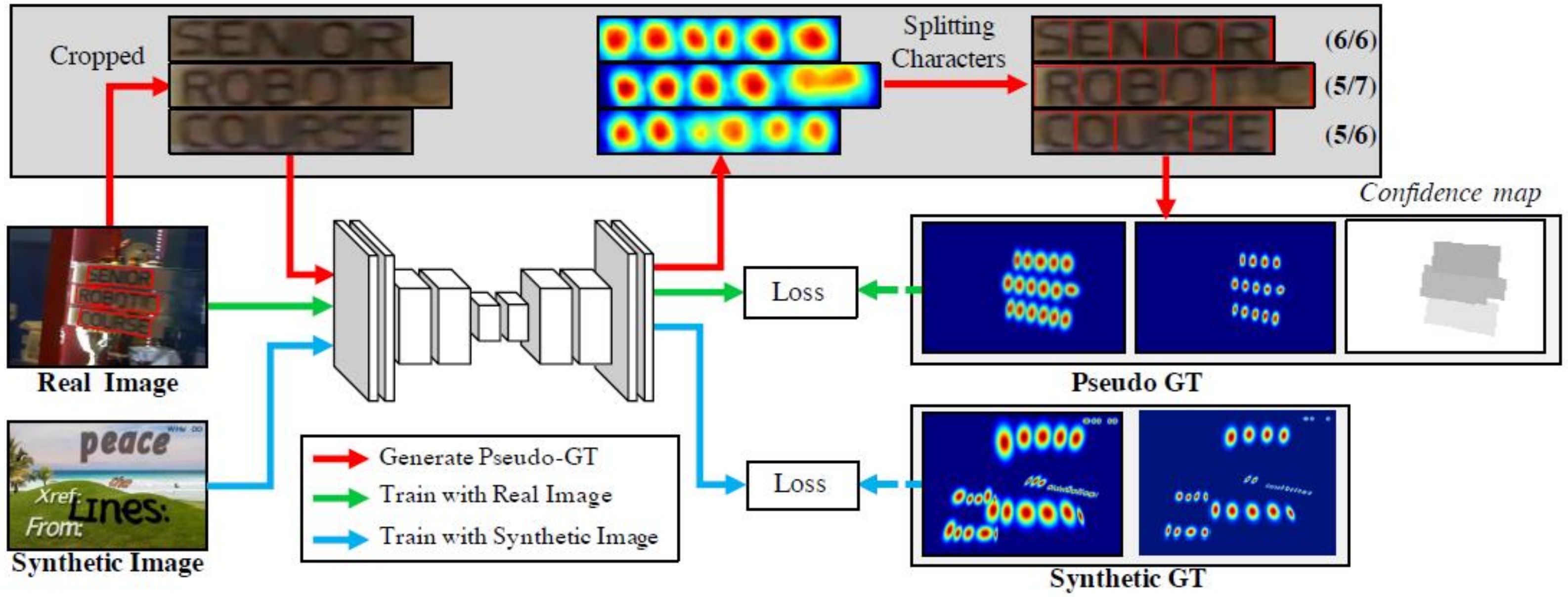
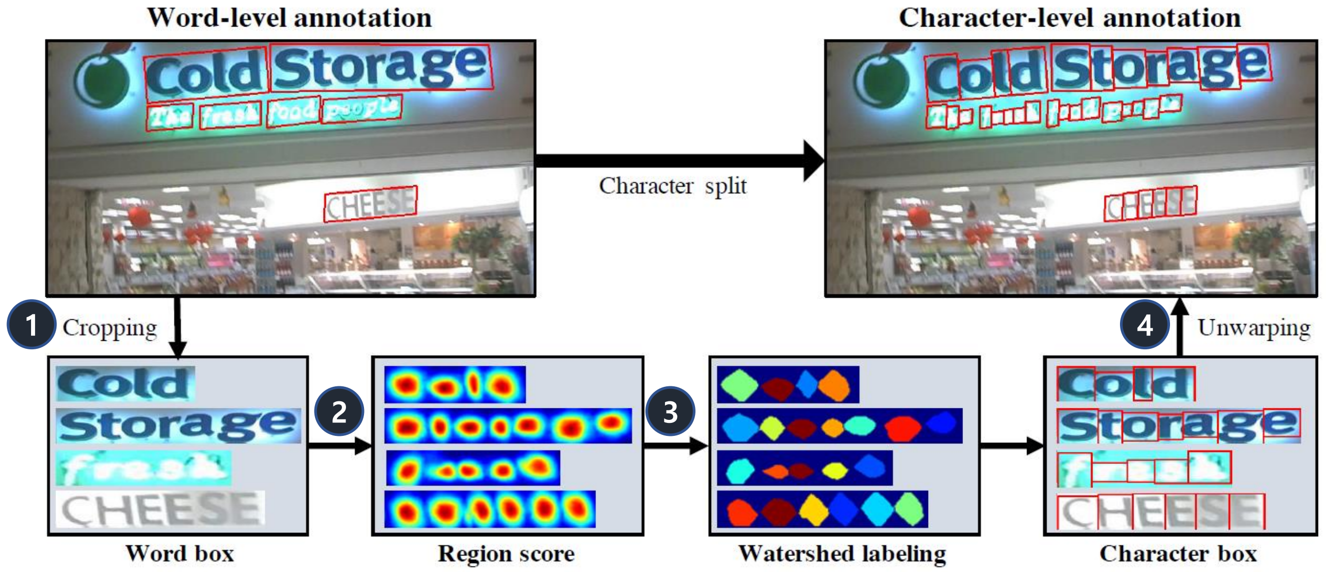


Figure 4. Illustration of the overall training stream for the proposed method. Training is carried out using both real and synthetic images in a weakly-supervised fashion.

[Training] Weakly-Supervised Learning



1. First, the word-level images are **cropped** from the original image.
2. Second, the model trained up to date **predicts the region score.**
3. Third, the watershed algorithm is used **to split the character regions**, which is used to make the character bounding boxes covering regions.
4. Finally, the coordinates of the character boxes are **transformed back into the original image** coordinates using the inverse transform from the cropping step.

The **pseudo-ground truths (pseudo-GTs)** for the **region score** and the **affinity score** can be generated by the steps described in *Fig. 3* using the obtained *quadrilateral character-level bounding boxes.*

Measure the quality of each pseudo-GTs (1)

■ **Purpose**

- When the model is trained using weak-supervision, compelled to train with incomplete pseudo-GTs
- To prevent training inaccurate the pseudo-GTs, measure the quality of each pseudo-GTs generated by the model
- Transcription of words is provided, and the length of the words can be used to evaluate the confidence of the pseudo-GTs.

■ **Confidence score**

$$s_{conf}(w) = \frac{l(w) - \min(l(w), |l(w) - l^c(w)|)}{l(w)}$$

▲ l(w) = True word Length
▲ l^c(w) = Estimated corresponding length of characters

CASE	$\min(l(w), \lvert l(w) - lc(w) \rvert)$	$S_{conf}\ (w)$
$l(w) = l^c(w)$	*0*	$\frac{l(w) - 0}{l(w)} = 1$
$l(w) > \lvert l(w) - l^c(w) \rvert$	$l(w) - lc(w)$	$\frac{l(w) - \{l(w) - lc(w)\}}{l(w)} = \frac{l(w) - l(w) + lc(w)}{l(w)} = \frac{l^c(w)}{l(w)}$
$l(w) < \lvert l(w) - l^c(w) \rvert$	$l(w)$	$\frac{l(w) - l(w)}{l(w)} = 0$

Cf. **If $S_{conf}\ (w)$ is below 0.5**, the estimated character bounding boxes should be neglected since they have **adverse effects when training the model**

Measure the quality of each pseudo-GTs (2)

■ **Pixel-wise confidence map**

- Correctness of its predictions for each pixel in the input image.
- Assigns a confidence score to each pixel in the image, indicating how likely it is that the pixel belongs to a certain class or object.

$$S_c(p) = \begin{cases} s_{conf}(w) & p \in R(w), \\ 1 & \text{otherwise}, \end{cases}$$

▲ R(w) = Bounding box region
▲ p = the pixel in the region R(w)

■ **Objective L**

$$L = \sum_p S_c(p) \cdot \left(||S_r(p) - S_r^*(p)||_2^2 + ||S_a(p) - S_a^*(p)||_2^2 \right),$$

▲ $S_r^* P$ = pseudo−ground truth region score
▲ $S_r P$ = predicted region score
▲ $S_a^* P$ = pseudo−ground truth affinity map
▲ $S_a P$ = predicted affinity score

Cf. When training with synthetic data, we can obtain the real ground truth, so $S_c P$ is set to 1.

Cf. Weakly vs. Strong in supervised learning

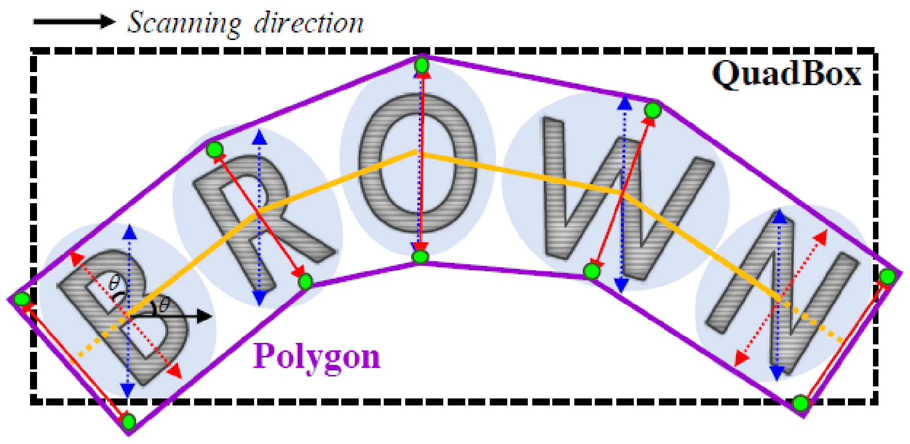
- In the context of **OCR (Optical Character Recognition)**, weakly supervised learning and strong supervised learning refer to different types of training data used to train the OCR model.

	Train	Pros.	Cons.
Strong supervised learning	ground truth text labels are provided for each character or word in the input image	Achieve the high accuracy	time-consuming and expensive to collect
weakly supervised learning	has only partial or incomplete annotations, or no annotations at all	cost-effective solution using less labeled data	Model accuracy is not high as in strong supervised learning

- In summary, weakly supervised learning allows training OCR models using less labeled data and can be a cost-effective solution. However, the tradeoff is that the accuracy of the model may not be as high as in strong supervised learning, which relies on fully annotated training data.

INFERENCE

Post-processing for finding Bounding Box



: Character region
: Local maxima along scanning direction
: Center line of local maxima
: Line of control points (tilted from local maxima)
: **Control points** of text polygon
: **Polygon text instance**

Figure 7. Polygon generation for arbitrarily-shaped texts.

The post-processing for finding bounding boxes is summarized as follows. First, the binary map M covering the image is initialized with 0. $M(p)$ is set to 1 if $S_r(p) > \tau_r$ or $S_a(p) > \tau_a$, where τ_r is the region threshold and τ_a is the affinity threshold. Second, Connected Component Labeling (CCL) on M is performed. Lastly, *QuadBox* is obtained by finding a rotated rectangle with the minimum area enclosing the connected components corresponding to each of the labels. The functions like *connectedComponents* and *minAreaRect* provided by OpenCV can be applied for this purpose.

Polygon generation

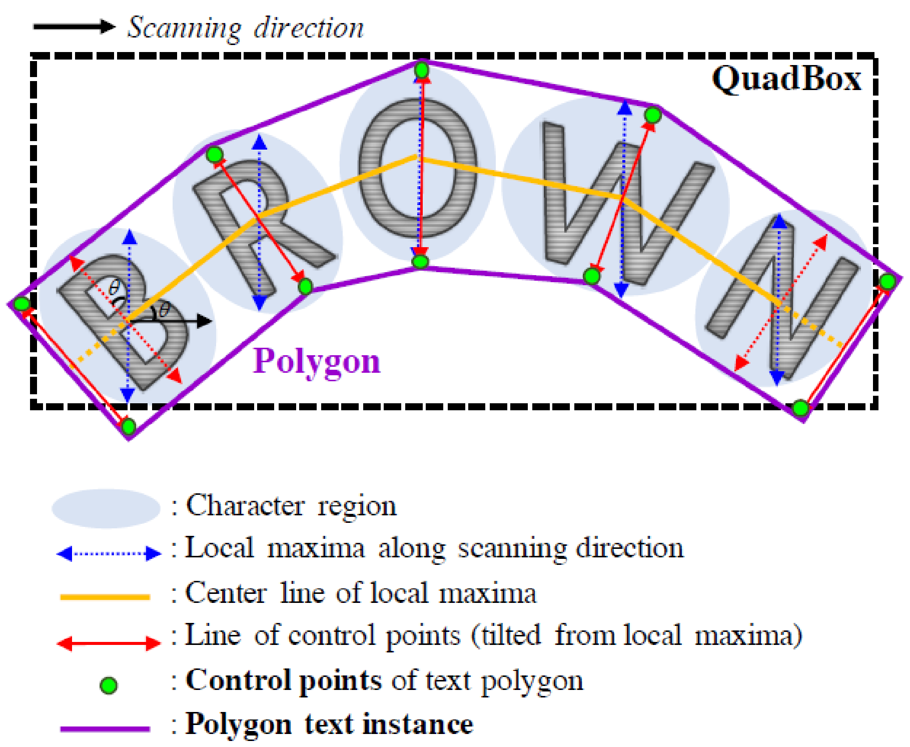

Scanning direction
QuadBox
Polygon
θ
θ
: Character region
: Local maxima along scanning direction
: Center line of local maxima
: Line of control points (tilted from local maxima)
: Control points of text polygon
: Polygon text instance

Figure 7. Polygon generation for arbitrarily-shaped texts.

Can generate a polygon around the entire character region to deal with curved texts effectively.

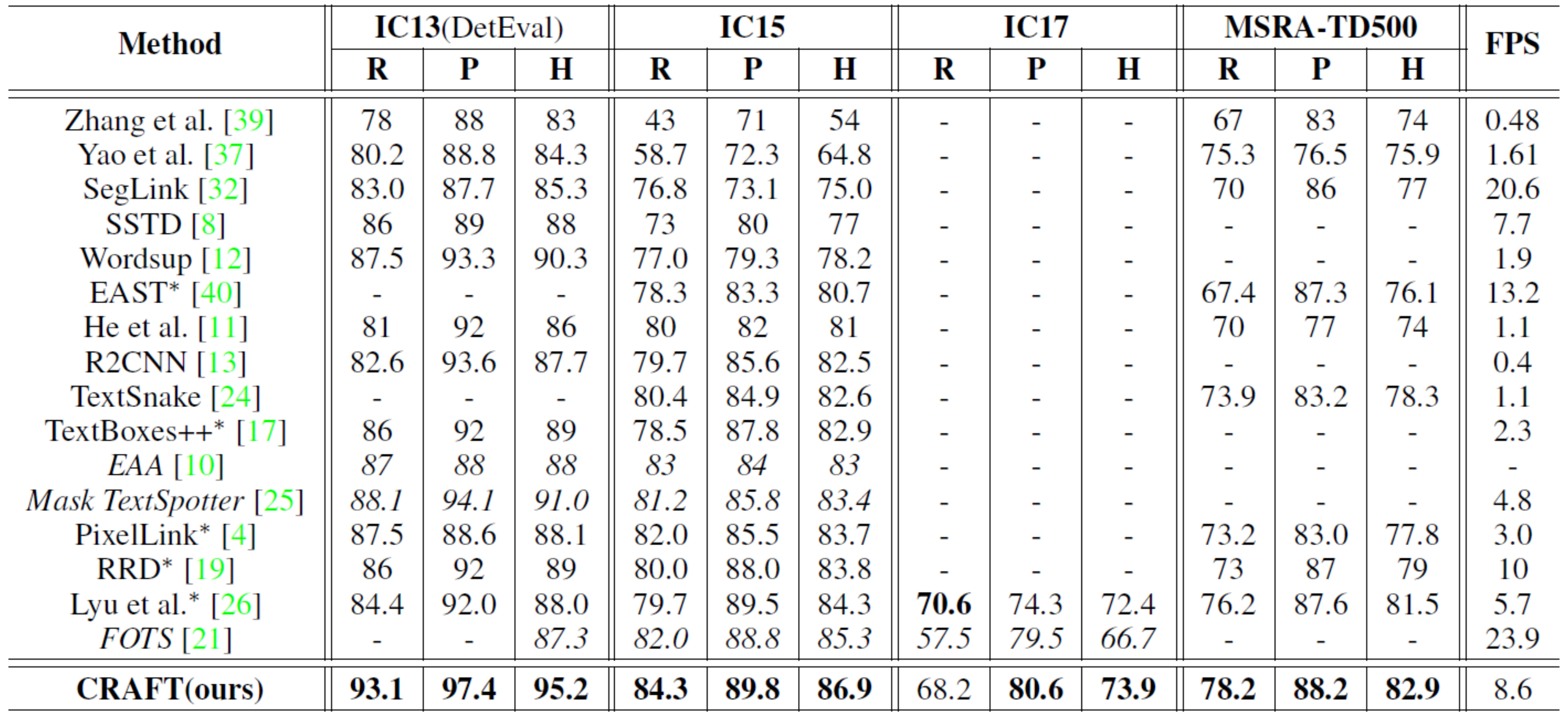
1. The first step is **to find the local maxima line of character regions** along the scanning direction, as shown in the figure with arrows in blue.
   - The lengths of the local maxima lines are **equally set as the maximum length** among them to prevent the final polygon result from becoming uneven.
   - The line connecting all the **center points of the local maxima is called the center line**, shown in yellow.

2. Local maxima lines are rotated to be perpendicular to the **center line** to reflect the tilt angle of characters, as expressed by the red arrows.
3.

   The endpoints of the local maxima lines are the candidates for the control points of the text polygon. To fully cover the text region, we move the **two outer-most tilted local maxima lines** outward along the local maxima center line, making **the final control points (green dots).**

EXPERIMENT

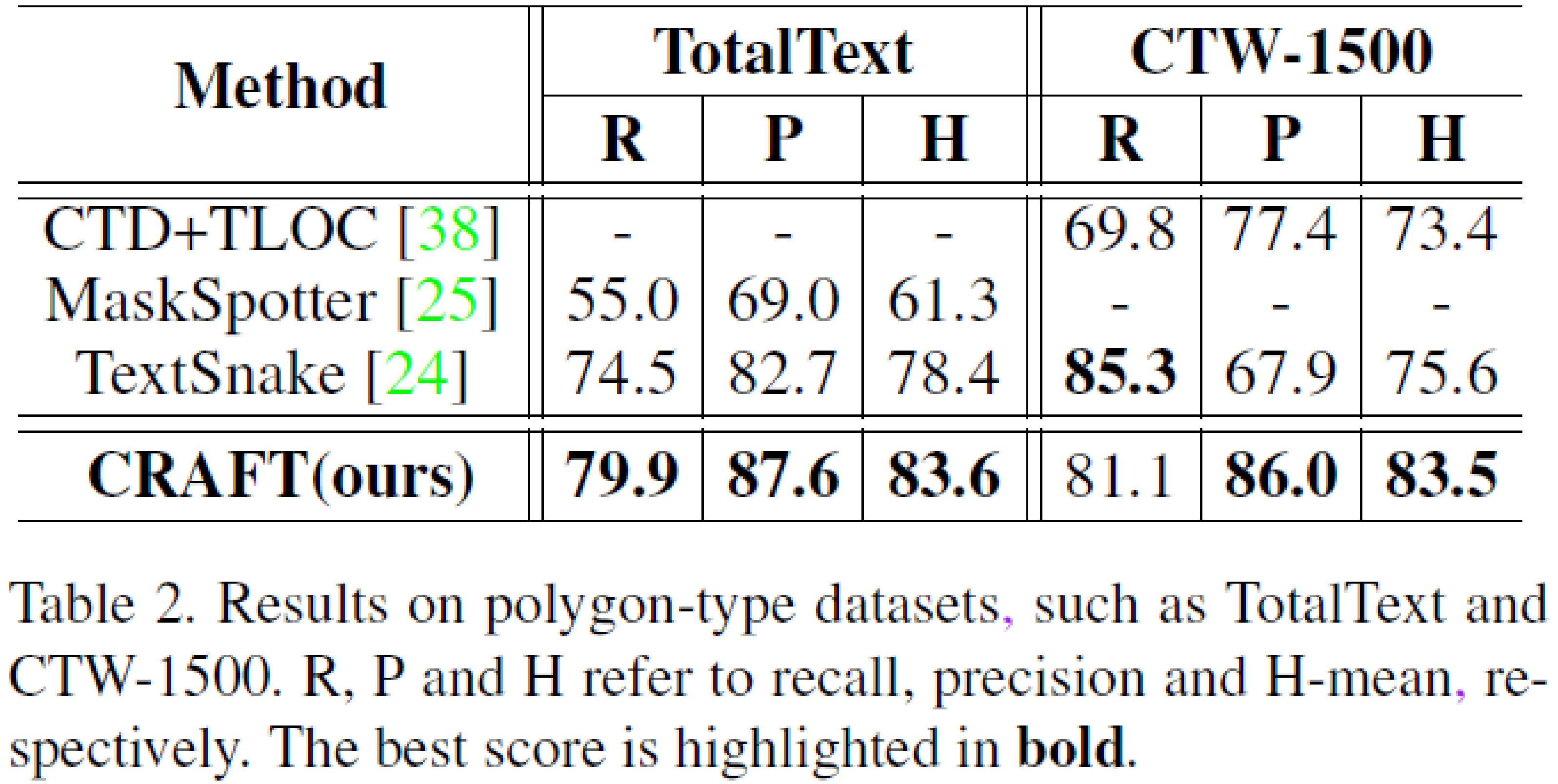
Results on Quadrilateral-type datasets



Method	IC13(DetEval)			IC15			IC17			MSRA-TD500			FPS
	R	**P**	**H**	**R**	**P**	**H**	**R**	**P**	**H**	**R**	**P**	**H**	
Zhang et al. [39]	78	88	83	43	71	54	-	-	-	67	83	74	0.48
Yao et al. [37]	80.2	88.8	84.3	58.7	72.3	64.8	-	-	-	75.3	76.5	75.9	1.61
SegLink [32]	83.0	87.7	85.3	76.8	73.1	75.0	-	-	-	70	86	77	20.6
SSTD [8]	86	89	88	73	80	77	-	-	-	-	-	-	7.7
Wordsup [12]	87.5	93.3	90.3	77.0	79.3	78.2	-	-	-	-	-	-	1.9
EAST* [40]	-	-	-	78.3	83.3	80.7	-	-	-	67.4	87.3	76.1	13.2
He et al. [11]	81	92	86	80	82	81	-	-	-	70	77	74	1.1
R2CNN [13]	82.6	93.6	87.7	79.7	85.6	82.5	-	-	-	-	-	-	0.4
TextSnake [24]	-	-	-	80.4	84.9	82.6	-	-	-	73.9	83.2	78.3	1.1
TextBoxes++* [17]	86	92	89	78.5	87.8	82.9	-	-	-	-	-	-	2.3
EAA [10]	*87*	*88*	*88*	*83*	*84*	*83*	-	-	-	-	-	-	-
Mask TextSpotter [25]	*88.1*	*94.1*	*91.0*	*81.2*	*85.8*	*83.4*	-	-	-	-	-	-	4.8
PixelLink* [4]	87.5	88.6	88.1	82.0	85.5	83.7	-	-	-	73.2	83.0	77.8	3.0
RRD* [19]	86	92	89	80.0	88.0	83.8	-	-	-	73	87	79	10
Lyu et al.* [26]	84.4	92.0	88.0	79.7	89.5	84.3	**70.6**	74.3	72.4	76.2	87.6	81.5	5.7
FOTS [21]	-	-	*87.3*	*82.0*	*88.8*	*85.3*	*57.5*	*79.5*	*66.7*	-	-	-	23.9
CRAFT(ours)	**93.1**	**97.4**	**95.2**	**84.3**	**89.8**	**86.9**	68.2	**80.6**	**73.9**	**78.2**	**88.2**	**82.9**	8.6

Table 1. Results on quadrilateral-type datasets, such as ICDAR and MSRA-TD500. * denote the results based on multi-scale tests. Methods in *italic* are results solely from the detection of end-to-end models for a fair comparison. R, P, and H refer to recall, precision and H-mean, respectively. The best score is highlighted in **bold**. FPS is for reference only because the experimental environments are different. We report the best FPSs, each of which was reported in the original paper.

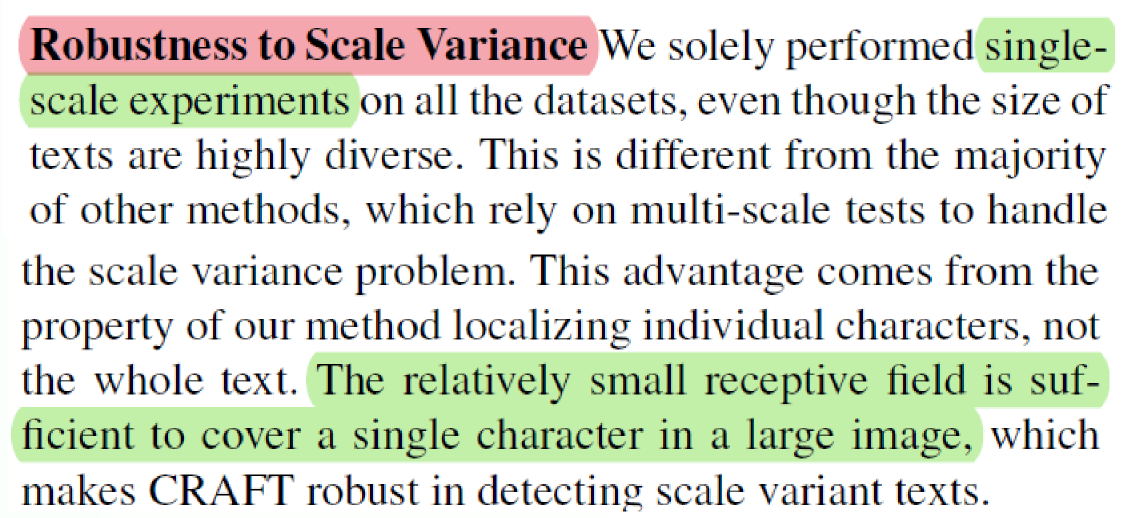
Results on Polygon-type datasets



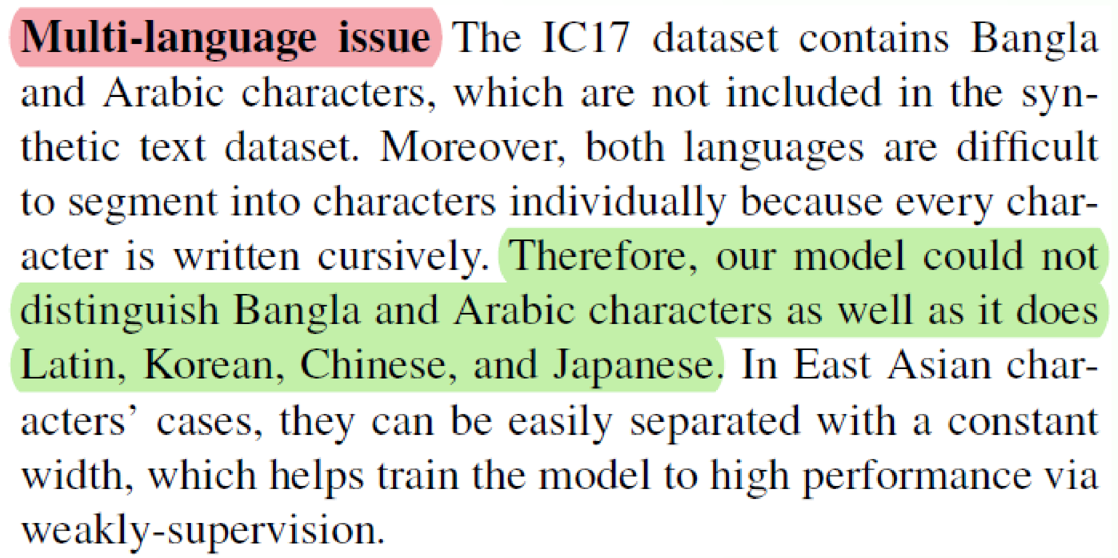
Method	**TotalText**			**CTW-1500**		
	R	**P**	**H**	**R**	**P**	**H**
CTD+TLOC [38]	-	-	-	69.8	77.4	73.4
MaskSpotter [25]	55.0	69.0	61.3	-	-	-
TextSnake [24]	74.5	82.7	78.4	**85.3**	67.9	75.6
CRAFT(ours)	**79.9**	**87.6**	**83.6**	81.1	**86.0**	**83.5**

Table 2. Results on polygon-type datasets, such as TotalText and CTW-1500. R, P and H refer to recall, precision and H-mean, respectively. The best score is highlighted in **bold**.

Discussion



Robustness to Scale Variance We solely performed single-scale experiments on all the datasets, even though the size of texts are highly diverse. This is different from the majority of other methods, which rely on multi-scale tests to handle the scale variance problem. This advantage comes from the property of our method localizing individual characters, not the whole text. The relatively small receptive field is sufficient to cover a single character in a large image, which makes CRAFT robust in detecting scale variant texts.



Multi-language issue The IC17 dataset contains Bangla and Arabic characters, which are not included in the synthetic text dataset. Moreover, both languages are difficult to segment into characters individually because every character is written cursively. Therefore, our model could not distinguish Bangla and Arabic characters as well as it does Latin, Korean, Chinese, and Japanese. In East Asian characters' cases, they can be easily separated with a constant width, which helps train the model to high performance via weakly-supervision.

Discussion

Comparison with End-to-end methods Our method is trained with the ground truth boxes only for detection, but it is comparable with other end-to-end methods, as shown in Table. 3. From the analysis of failure cases, we expect our model to benefit from the recognition results, especially when the ground truth words are separated by semantics, rather than visual cues.

Method	IC13	IC15	IC17
Mask TextSpotter [25]	91.7	86.0	-
EAA [10]	90	87	-
FOTS [21]	92.8	**89.8**	70.8
CRAFT(ours)	**95.2**	86.9	**73.9**

Table 3. H-mean comparison with end-to-end methods. Our method is not trained in an end-to-end manner, yet shows comparable results, or even outperforms popular methods.

Generalization ability Our method achieved state-of-the-art performances on 3 different datasets without additional fine-tuning. This demonstrates that our model is capable of capturing general characteristics of texts, rather than over-fitting to a particular dataset.

Cf. H-mean comparison

- H-mean comparison is a method used in Optical Character Recognition (OCR) to evaluate the accuracy of the OCR system. In OCR, the system tries to recognize text characters from an image or a scanned document.

- H-mean comparison involves comparing the recognized characters with the actual characters present in the document. **The H-mean value is a metric that combines the precision and recall of the OCR system**. It is calculated as the **harmonic mean of the precision and recall values.**
  - *Same with F1-score, both are calculated as the harmonic mean of precision and recall*

- **Precision** is the ratio of the number of **correctly recognized characters** to the total number of **recognized characters.**
- **Recall** is the ratio of the number of **correctly recognized character**s to the total number of **actual characters.**

- The H-mean value is a good measure of the OCR system's performance because it considers both precision and recall. A high H-mean value indicates that the OCR system is accurate and can detect a high percentage of characters correctly.

APPEDNIX

APPENDIX

HYUNHP

LIST

1. NAVER Clova OCR -English version-
   https://youtu.be/NQeaLc2X8vk

2. CRAFT – pytorch official paper
   https://github.com/clovaai/CRAFT-pytorch

3. https://medium.com/@msmapark2/craft-%EB%84%A4%EC%9D%B4%EB%B2%84%EC%9D%98-%EA%B8%80%EC%9E%90-%EC%B0%BE%EB%8A%94-ai-dca5e8aff007

4. https://paper-cat.github.io/2020-10/project2