DeepSolo: Let Transformer Decoder with Explicit Points Solo for Text Spotting

Maoyuan Ye[1*] Jing Zhang[2*] Shanshan Zhao[3] Juhua Liu[1†]
Tongliang Liu[2] Bo Du[1†] Dacheng Tao[3,2]
[1]Wuhan University, China [2]The University of Sydney, Australia [3]JD Explore Academy, China
{yemaoyuan, liujuhua, dubo}@whu.edu.cn, {jing.zhang1, tongliang.liu}@sydney.edu.au,
{sshan.zhao00, dacheng.tao}@gmail.com

Abstract

End-to-end text spotting aims to integrate scene text detection and recognition into a unified framework. Dealing with the relationship between the two sub-tasks plays a pivotal role in designing effective spotters. Although Transformer-based methods eliminate the heuristic post-processing, they still suffer from the synergy issue between the sub-tasks and low training efficiency. In this paper, we present ***DeepSolo****, a simple DETR-like baseline that lets a single* ***Deep****coder with Explicit* ***P****oints* ***Solo*** *for text detection and recognition simultaneously. Technically, for each text instance, we represent the character sequence as ordered points and model them with learnable explicit point queries. After passing a single decoder, the point queries have encoded requisite text semantics and locations, thus can be further decoded to the center line, boundary, script, and confidence of text via very simple prediction heads in parallel. Besides, we also introduce a text-matching criterion to deliver more accurate supervisory signals, thus enabling more efficient training. Quantitative experiments on public benchmarks demonstrate that DeepSolo outperforms previous state-of-the-art methods and achieves better training efficiency. In addition, DeepSolo is also compatible with line annotations, which require much less annotation cost than polygons. The code is available at* https://github.com/ViTAE-Transformer/DeepSolo.

1. Introduction

Detecting and recognizing text in natural scenes, *a.k.a.* text spotting, has drawn increasing attention due to its wide range of applications [5, 34, 56, 59], such as autonomous driving [57] and intelligent navigation [7]. How to deal with the relationship between detection and recognition is a long-

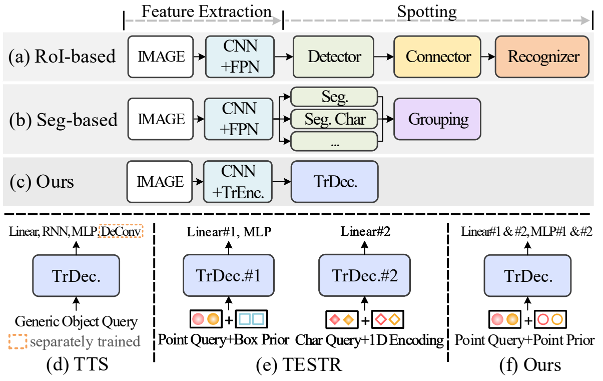

Figure 1. Comparison of pipelines and query designs. TrEnc. (TrDec.): Transformer encoder (decoder). Char: character.

standing problem in designing the text spotting pipeline and has a significant impact on structure design, spotting performance, training efficiency, and annotation cost, etc.

Most pioneering end-to-end spotting methods [16,25,28,29,31,37,43,44,51] follow a detect-then-recognize pipeline, which first detects text instances and then exploits Region-of-Interest (RoI) based connectors to extract features within the detected area, finally feeds them into the following recognizer (Fig. 1a). Although these methods have achieved great progress, there are two main limitations. 1) An extra connector for feature alignment is indispensable. Moreover, some connectors require polygon annotations, which are not applicable when only weak annotations are available. 2) Additional efforts are desired to address the synergy issue [17,62] between the detection and recognition modules. In contrast, the segmentation-based methods [49, 53] try to isolate the two sub-tasks and complete spotting in a parallel multi-task framework with a shared backbone (Fig. 1b). Nevertheless, they are sensitive to noise and require grouping post-processing to gather unstructured components.

Recently, Transformer [47] has improved the performance remarkably for various computer vision tasks [9, 10, 23, 32, 33, 40, 46, 50, 54, 60], including text spotting [17, 21, 39, 61]. Although the spotters [21, 61] based on DETR [3]

[*]Equal contribution. †Corresponding author. This work was done during Maoyuan Ye's internship at JD Explore Academy.

can get rid of the connectors and heuristic post-processing, they lack efficient joint representation to deal with scene text detection and recognition, *e.g.*, requiring an extra RNN module in TTS [21] (Fig. 1d) or exploiting individual Transformer decoder for each sub-task in TESTR [61] (Fig. 1e). The generic object query exploited in TTS fails to consider the unique characteristics of scene text, *e.g.*, location and shape. While TESTR uses point queries with box positional prior that is coarse for point predicting, and the queries are different for detection and recognition, introducing unexpected heterogeneity. Consequently, these designs have a side effect on the performance and training efficiency [55].

In this paper, we propose a novel query form based on explicit point representations of text lines. Built upon it, we present a succinct DETR-like baseline that lets a single **De**coder with **E**xplicit **P**oints **Solo** (dubbed **DeepSolo**) for detection and recognition simultaneously (Fig. 1c and Fig. 1f). Technically, for each instance, we first represent the character sequence as ordered points, where each point has explicit attributes of position, offsets to the top and bottom boundary, and category. Specifically, we devise top-K Bezier center curves to fit scene text instances with arbitrary shape and sample a fixed number of on-curve points covering characters in each text instance. Then, we leverage the sampled points to generate positional queries and guide the learnable content queries with explicit positional prior. Next, we feed the image features from the Transformer encoder and the point queries into a single Transformer decoder, where the output queries are expected to have encoded requisite text semantics and locations. Finally, we adopt several very simple prediction heads (a linear layer or MLP) in parallel to decode the queries into the center line, boundary, script, and confidence of text, thereby solving detection and recognition simultaneously.

In summary, the main contributions are three-fold: **1)** We propose DeepSolo, *i.e.*, a succinct DETR-like baseline with a single Transformer decoder and several simple prediction heads, to solve text spotting efficiently. **2)** We propose a novel query form based on explicit points sampled from the Bezier center curve representation of text instance lines, which can efficiently encode the position, shape, and semantics of text, thus helping simplify the text spotting pipeline. **3)** Experimental results on public datasets demonstrate that DeepSolo is superior to previous representative methods in terms of spotting accuracy, training efficiency, and annotation flexibility.

2. Related Works

2.1. Connector-based Text Spotters

Recent works focus on developing end-to-end spotters. Most of them craft RoI [14] based or Thin-Plate Splines (TPS) [2] based connectors to bridge detection and recognition modules. In [11, 16, 28], given the detection results, RoI-Align or its variant is exploited to extract text features for the following recognizer. Mask TextSpotter series [24, 25, 37] conduct character segmentation in recognition. GLASS [44] devises a plug-in global-to-local attention module with Rotated-RoIAlign for strong recognition. In comparison, [42, 48] use TPS to rectify curve texts with control points. To better rectify curve texts, ABCNet series [29, 31] propose the BezierAlign module using parameterized Bezier curve. Although these methods successfully bridge text detection and recognition, they ignore the synergy issue [17, 62] between the two tasks. To overcome this dilemma, SwinTextSpotter [17] proposes a tailored Recognition Conversion module to back-propagate recognition information to the detector. Although the above methods have achieved remarkable progress, they require an extra RoI-based or TPS-based connector. Since some connectors require polygon annotations, the methods may be not applicable to scenarios with only weak position annotations (*e.g.*, lines). Moreover, a more effective and simpler solution is also expected to address the synergy issue.

2.2. Connector-free Text Spotters

To get rid of the extra connector, segmentation-based methods [49, 53] adopt parallel branches to segment characters and instances. However, they tend to be vulnerable to noise and rely on heuristic grouping. In contrast, SRSTS [52] estimates positive anchors for each instance, then adopts a text shape segmentation branch and a self-reliant sampling recognition branch in parallel to effectively decouple detection and recognition. Nevertheless, it needs NMS for post-processing.

Inspired by DETR family [3, 63], recent works [21, 39, 61] explore the Transformer framework without RoI-Align and complicated post-processing. TESTR [61] adopts two parallel Transformer decoders for detection and recognition. TTS [21] adds an RNN recognizer into Deformable-DETR [63] and shows its potential on weak annotations. SPTS [39] follows the sequence prediction paradigm [4] and demonstrates that using single-point or even script-only annotations is feasible and promising.

Although these methods unlock the potential of Transformer in text spotting, there are still some limitations. For example, the vanilla or individual queries used in TTS and TESTR cannot efficiently encode text features (*e.g.*, location, shape, and semantics), affecting the training efficiency [55] and even increasing the complexity of the pipeline. The box annotations used in TTS might not be ideal [39] for scene text since the box contains a certain portion of background regions and even other texts, thus introducing extra noise. SPTS, TTS, and our work provide more comprehensive but different supervision solutions for the community.

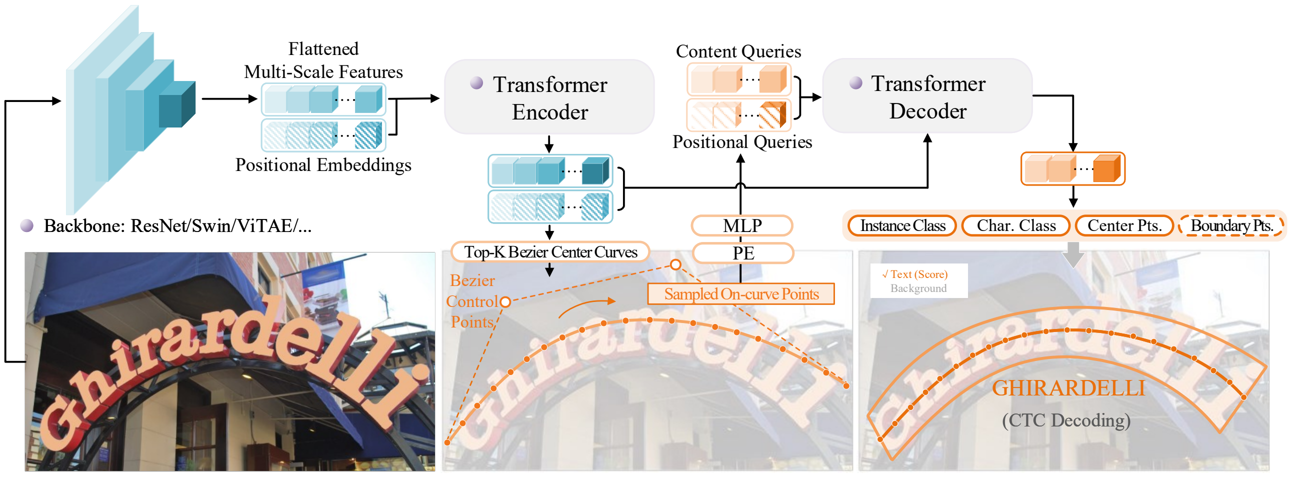

Figure 2. The overall architecture of DeepSolo. We propose an explicit query form based on the points sampled from the Bezier center curve representation of text, solving spotting with a single decoder and simple prediction heads in a unified and concise framework.

3. Methodology

In this paper, we propose **DeepSolo** for text spotting. DeepSolo aims at achieving detection and recognition simultaneously and efficiently with a single Transformer decoder by digging into the close relationship between them.

3.1. Overview

Preliminary. Bezier curve, firstly introduced into text spotting by ABCNet [29], can flexibly fit the shape of scene text with a certain number of control points. Different from ABCNet which crops features with BezierAlign, we explore the distinctive utilization of Bezier curve in the DETR framework. More details of the Bezier curve generation can be referred to [29, 31]. Given four Bezier control points[1] for each side (top and bottom) of a text instance, we simply compute Bezier control points for the center curve by averaging the corresponding control points on top and bottom sides. Then, a fixed number of N points are uniformly sampled on the center, top, and bottom curves, serving as the ground truth. Note that the order of the points is in line with the text reading order.

Model Architecture. The overall architecture is illustrated in Fig. 2. After receiving the image features from the encoder equipped with deformable attention [63], Bezier center curve proposals and their scores are generated. Then, the top-K scored ones are selected. For each selected curve proposal represented by four Bezier control points, N points are uniformly sampled on the curve. The coordinates of these points are encoded as positional queries and added to the learnable content queries, forming the composite queries. Next, the composite queries are fed into the decoder to gather useful text features via deformable cross-attention. Following the decoder, four simple parallel heads are adopted, each of which is responsible for solving a specific task.

3.2. Top-K Bezier Center Curve Proposals

Different from the box proposal adopted in previous works [18,55,58,61,63], which has the drawbacks in representing text with arbitrary shape, we design a simple Bezier center curve proposal scheme from the text center line perspective. It can efficiently fit scene text and distinguish one from others, which also makes the usage of line annotations possible. Specifically, given the image features from the encoder, on each pixel of the feature maps, a 3-layer MLP (8-dim in the last layer) is used to predict offsets to four Bezier control points, determining a curve that represents one text instance. Let i index a pixel from features at level $l \in \{1,2,3,4\}$ with 2D normalized coordinates $\hat{p}_i = (\hat{p}_{ix}, \hat{p}_{iy}) \in [0,1]^2$, its corresponding Bezier control points $BP_i = \{\bar{bp}_{i_0}, \bar{bp}_{i_1}, \bar{bp}_{i_2}, \bar{bp}_{i_3}\}$ are predicted:

$$\bar{bp}_{i_j} = (\sigma(\Delta p_{ix_j} + \sigma^{-1}(\hat{p}_{ix})), \sigma(\Delta p_{iy_j} + \sigma^{-1}(\hat{p}_{iy}))), \quad (1)$$

where $j \in \{0,1,2,3\}$, σ is the sigmoid function. The MLP head only predicts the offsets $\Delta p_{i\{x_0,y_0,\ldots,x_3,y_3\}}$. Moreover, we use a linear layer for text or not text classification. And top-K scored curves are selected as the proposals.

3.3. Point Query Modeling for Text Spotting

Query Initialization. Given the top-K proposals $\hat{BP}_k (k \in \{0,1,\ldots,K-1\})$, we uniformly sample N points on each curve according to the Bernstein Polynomials [35]. Here, we get normalized point coordinates $Coords$ of shape $K \times N \times 2$ in each image. The point positional queries $\mathbf{P}_q$ of shape $K \times N \times 256$ are generated by:

$$\mathbf{P}_q = MLP(PE(Coords)), \quad (2)$$

[1]Provided by ABCNet [29]

where PE represents the sinusoidal positional encoding function. Following [27], we also adopt a 2-layer MLP head with the $ReLU$ activation for further projection. On the other side, we initialize point content queries $\mathbf{C}_q$ using learnable embeddings. Then, we add $\mathbf{P}_q$ to $\mathbf{C}_q$ to get the composite queries $\mathbf{Q}_q$:

$$\mathbf{Q}_q = \mathbf{C}_q + \mathbf{P}_q. \tag{3}$$

We empirically exploit unshared point embeddings for $\mathbf{C}_q$, *i.e.*, N point content queries of shape $N \times 256$ in one text instance are used for each of the K proposals.

Query Update in the Decoder. After obtaining the composite queries $\mathbf{Q}_q$, we feed them into the Transformer decoder. We follow previous works [8, 55, 61] to firstly mine the relationship between point queries within one text instance using an intra-group self-attention across dimension N. Here, keys are the same with queries while values only contain the content part: $\mathbf{K}_q = \mathbf{Q}_q$, $\mathbf{V}_q = \mathbf{C}_q$. Then, an inter-group self-attention is conducted across K instances to capture the relationship between different instances. The updated composite queries are further sent into the deformable cross-attention to aggregate multi-scale text features from the encoder. The point coordinates $Coords$ are used as the reference points in the deformable attention.

With the explicit point information flowing in the decoder, we adopt a 3-layer MLP head to predict the offsets and update point coordinates after each decoder layer. Then, the updated coordinates will be transformed into new positional queries by Eq. (2).

3.4. Task Prediction

After getting queries of shape $K \times N \times 256$ from decoder for each image, we adopt simple prediction heads to solve sub-tasks. **(1) Instance classification.** We use a linear projection for binary classification (text or background). During inference, we take the mean of N scores as the confidence score for each instance. **(2) Character classification.** As the points are uniformly sampled on the center curve of each text to cover characters, each point query represents a specific class (including background). We adopt a linear projection to perform character classification. **(3) Center curve points.** Given the explicit point coordinates $Coords$, a 3-layer MLP head MLP_{coord} is used to predict the coordinate offsets from reference points to ground truth points on the center curve. **(4) Boundary points.** Similarly, a 3-layer MLP head MLP_{bd} is used to predict the offsets to the ground truth points on the top and bottom curves.

3.5. Optimization

Bipartite Matching. After obtaining the prediction set $\hat{Y}$ from the four heads and the ground truth set Y, we use the Hungarian algorithm [22] to get an optimal injective function $\varphi : [Y] \mapsto [\hat{Y}]$ that minimizes the matching cost $\mathcal{C}$:

$$\arg\min_{\varphi} \sum_{g=0}^{G-1} \mathcal{C}(Y^{(g)}, \hat{Y}^{(\varphi(g))}), \tag{4}$$

where G is the number of ground truth instances per image.

Regarding the cost $\mathcal{C}$, previous work [61] only considers the class and position similarity while ignoring the similarity of the text script. However, matched pairs with similar positions may be quite different in texts, which could increases the optimization difficulty. TTS [21] proposes a cross-entropy-based text matching criterion to address this issue, but it is not applicable to our method since the character predictions are not aligned with the ground truth due to background class and repeated characters. We introduce a text matching criterion based on the typical Connectionist Temporal Classification loss [13] which copes with the length inconsistency issue. For the g-th ground truth and its matched query, the complete cost function is:

$$\begin{aligned}\mathcal{C}(Y^{(g)}, \hat{Y}^{(\varphi(g))}) =& \lambda_{\text{cls}}\text{FL}'(\hat{b}^{(\varphi(g))}) + \lambda_{\text{text}}\text{CTC}(t^{(g)}, \hat{t}^{(\varphi(g))}) \\ &+ \lambda_{\text{coord}} \sum_{n=0}^{N-1} \left\| p_n^{(g)} - \hat{p}_n^{(\varphi(g))} \right\|,\end{aligned} \tag{5}$$

where λ_{cls}, λ_{text}, and λ_{coord} are hyper-parameters to balance different tasks. $\hat{b}^{(\varphi(g))}$ is the probability for the text-only instance class. The same as [61], FL$'$ is defined as the difference between the positive and negative term: $\text{FL}'(x) = -\alpha(1-x)^{\gamma}\log(x) + (1-\alpha)x^{\gamma}\log(1-x)$, which is derived from the focal loss [26]. The second term is the CTC loss between the ground truth text script and predicted N characters. The third term is the L1 distance between the ground truth and predicted point coordinates on the center curve.

Overall Loss. For the k-th query, the focal loss for instance classification is formulated as:

$$\begin{aligned}\mathcal{L}_{\text{cls}}^{(k)} =& -\mathbb{1}_{\{k \in \text{Im}(\varphi)\}}\alpha(1-\hat{b}^{(k)})^{\gamma}\log(\hat{b}^{(k)}) \\ &- \mathbb{1}_{\{k \notin \text{Im}(\varphi)\}}(1-\alpha)(\hat{b}^{(k)})^{\gamma}\log(1-\hat{b}^{(k)}),\end{aligned} \tag{6}$$

where $\mathbb{1}$ is the indicator function, $\text{Im}(\varphi)$ is the image of the mapping φ. As for character classification, we exploit the CTC loss to address the length inconsistency issue between the ground truth text scripts and predictions:

$$\mathcal{L}_{\text{text}}^{(k)} = \mathbb{1}_{\{k \in \text{Im}(\varphi)\}}\text{CTC}(t^{(\varphi^{-1}(k))}, \hat{t}^{(k)}). \tag{7}$$

In addition, L1 distance loss is used for supervising points coordinates on the center curve and the boundaries (*i.e.*, the top and bottom curves):

$$\mathcal{L}_{\text{coord}}^{(k)} = \mathbb{1}_{\{k \in \text{Im}(\varphi)\}} \sum_{n=0}^{N-1} \left\| p_n^{(\varphi^{-1}(k))} - \hat{p}_n^{(k)} \right\|, \tag{8}$$

$$\mathcal{L}_{\text{bd}}^{(k)} = \mathbb{1}_{\{k \in \text{Im}(\varphi)\}} \sum_{n=0}^{N-1} \left(\left\| top_n^{(\varphi^{-1}(k))} - \hat{top}_n^{(k)} \right\| + \left\| bot_n^{(\varphi^{-1}(k))} - \hat{bot}_n^{(k)} \right\| \right). \tag{9}$$

The loss function for the decoder consists of the four aforementioned losses:

$$\mathcal{L}_{\text{dec}} = \sum_{k} \left(\lambda_{\text{cls}} \mathcal{L}_{\text{cls}}^{(k)} + \lambda_{\text{text}} \mathcal{L}_{\text{text}}^{(k)} + \lambda_{\text{coord}} \mathcal{L}_{\text{coord}}^{(k)} + \lambda_{\text{bd}} \mathcal{L}_{\text{bd}}^{(k)} \right), \tag{10}$$

where the hyper-parameters λ_{cls}, λ_{text}, and λ_{coord} are the same as those in Eq. (5). λ_{bd} is the boundary loss weight. In addition, to make the Bezier center curve proposals in Sec. 3.2 more accurate, we resort to adding intermediate supervision on the encoder. As we hope the points sampled on the Bezier center curve proposals are as close to the ground truth as possible, we calculate the L1 loss for the N uniformly sampled points instead of only the four Bezier control points for each instance. This supervision method has been explored by [12]. The loss function for the encoder is formulated as:

$$\mathcal{L}_{\text{enc}} = \sum_{i} \left(\lambda_{\text{cls}} \mathcal{L}_{\text{cls}}^{(i)} + \lambda_{\text{coord}} \mathcal{L}_{\text{coord}}^{(i)} \right), \tag{11}$$

where bipartite matching is also exploited to get one-to-one matching. The overall loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{dec}} + \mathcal{L}_{\text{enc}}. \tag{12}$$

4. Experiments

4.1. Settings

Benchmarks. We evaluate our method on Total-Text [6], ICDAR 2015 (IC15) [19] and CTW1500 [30]. **Total-Text** is an arbitrarily-shaped word-level scene text benchmark, with 1,255 training images and 300 testing images. **IC15** contains 1,000 training images and 500 testing images for quadrilateral scene text. Different from Total-Text and IC15, **CTW1500** is a text-line level benchmark for scene text with arbitrary shape. There are 1,000 training images and 500 testing images. We adopt the following additional datasets for pre-training: 1) Synth150K [29], a synthetic dataset that contains 94,723 images with multi-oriented texts and 54,327 images with curved texts. 2) ICDAR 2017 MLT (MLT17) [38], which is a multi-language scene text dataset. 3) ICDAR 2013 (IC13) [20] that contains 229 training images with horizontal text. We also investigate the influence of leveraging 21,778 training images and 3,124 validation images of TextOCR [45].

Implementation Details. The number of heads and sampling points for deformable attention is 8 and 4, respectively. The number of both encoder and decoder layers is 6. The number of proposals K is 100. The number of sampled points N is set to 25 by default without special instructions. Our models predict 38 character classes on Total-Text and IC15, 97 classes on CTW1500. AdamW [36] is used as the optimizer. The loss weights λ_{cls}, λ_{coord}, λ_{bd}, and λ_{text} are set to 1.0, 1.0, 0.5, and 0.5, respectively. For focal loss, α is 0.25 and γ is 2.0. The image batch size is 8. More details of the experiments can be found in the appendix.

4.2. Ablation Studies

We first conduct ablation experiments on Total-Text. We then investigate the influence of different training datasets and backbone choices. In Tab. 1 and Tab. 2, we pre-train each model on a mixture of Synth150K and Total-Text, then fine-tune it on Total-Text.

Text Loss Weight. We study the influence of the text loss weight, which has a direct impact on recognition performance. The results in Tab. 1 show that our model achieves a better trade-off between detection and end-to-end performance when λ_{text} is set to 0.5. We choose $\lambda_{\text{text}} = 0.5$ for other experiments.

λ_{text}	Detection			E2E	
	P	R	F1	None	Full
0.25	**94.29**	82.07	**87.76**	76.68	85.76
0.5	93.86	**82.11**	87.59	**78.83**	**86.15**
0.75	94.15	79.27	86.07	78.82	85.71
1.0	93.06	81.71	87.01	77.73	85.61

Table 1. Hyper-parameter study of λ_{text}. E2E: the end-to-end spotting results. None (Full) denotes the F1-measure without (with) using the lexicon that includes all words in the test set.

Sharing Point Embeddings. In Tab. 2, when sharing point embeddings for all instances, the results on end-to-end task drop. It indicates that different instances require different point embeddings to encode the instance-specific features.

Text Matching Criterion. To evaluate the effectiveness of the text matching criterion, in Tab. 2, we remove the text matching criterion from Eq. (5). The primary end-to-end results decline. It validates the value of conducting text matching, which provides high-quality one-to-one matching between the predictions and ground truth according to both the position and script similarity.

Training Data. We study the influence of different pre-training data in Tab. 3. For the end-to-end spotting task, with only Synth150K as the pre-training data, our method can achieve 78.83% accuracy without using lexicon. With additional MLT17, IC13, and IC15 real training data, the 'None' and 'Full' scores are improved by 0.82% and 0.85%, respectively. We further show that the performance is improved by a large margin using TextOCR. In TextOCR, the average number of text instances per image is higher than in other datasets, which can provide more positive signals. It demonstrates the value of using real data for pre-training and the scalability of our model on different data.

We further compare DeepSolo with existing open-source Transformer-based methods [17, 39, 61] by only using the training set of Total-Text. For a fair comparison, we apply

Sharing	Matching	Detection P	Detection R	Detection F1	E2E None	E2E Full	#Params	FPS
✗	✓	**93.86**	**82.11**	**87.59**	**78.83**	**86.15**	42.5M	17.0
✓	✓	93.60	81.21	86.96	77.58	85.98	41.8M	17.0
✗	✗	92.90	82.11	87.17	77.85	85.20	42.5M	17.0
✓	✗	93.41	81.98	87.32	77.09	85.13	41.8M	17.0

Table 2. Influence of sharing point embeddings and text matching.

#Row	External Data	Volume	Detection P	Detection R	Detection F1	E2E None	E2E Full
1	Synth150K	150K	**93.86**	82.11	87.59	78.83	86.15
2	Row#1+MLT17+IC13+IC15	160K	93.09	82.11	87.26	79.65	87.00
3	Row#2 +TextOCR	185K	93.19	**84.64**	**88.72**	**82.54**	**88.72**

Table 3. Influence of the training data. Volume: dataset volume.

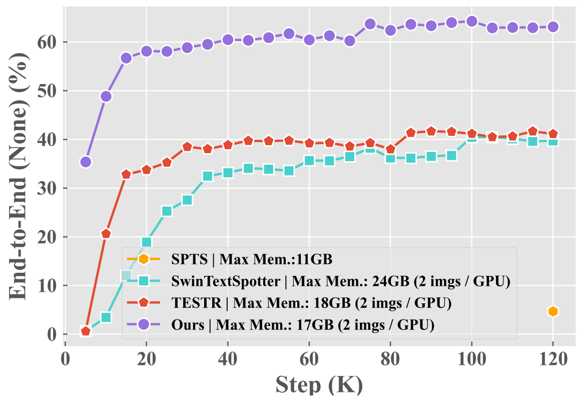

Figure 3. Comparison with open-source Transformer-based methods using only Total-Text training set.

Backbone	Detection P	Detection R	Detection F1	E2E None	E2E Full	#Params	Mem. (MB)
ResNet-50	93.09	82.11	87.26	79.65	87.00	42.5M	17,216
Swin-T	92.77	83.51	87.90	79.66	87.05	43.1M	26,573
ViTAEv2-S	92.57	**85.50**	**88.89**	**81.79**	**88.40**	33.7M	25,332
ResNet-101	93.20	83.51	88.09	80.12	87.14	61.5M	19,541
Swin-S	**93.72**	84.24	88.73	81.27	87.75	64.4M	33,974

Table 4. The influence of different backbones. Mem.: the peak memory of batching two images on one GPU during pre-training.

the same data augmentation. Since both TESTR and our method are based on Deformable-DETR [63], we set the same configuration for the Transformer modules. ResNet-50 [15] is adopted in all experiments. The image batch size is set to 8, while the actual batch size of SPTS doubles due to batch augmentation. In Fig. 3, our method achieves faster convergence and better performance, showing the superiority of DeepSolo over representative ones in training efficiency. In addition, the training GPU memory usage of our model is also less than SwinTextSpotter and TESTR.

Compared with TESTR which adopts dual decoders, the results demonstrate the value of our proposed query form, which contributes a unified representation to detection and recognition, encodes a more accurate and explicit position prior, and facilitates the learning of the single decoder. Consequently, instead of using complex dual decoders, our simpler design could effectively mitigate the synergy issue.

Different Backbones. We also conduct experiments to investigate the influence of different backbones on our model in Tab. 4. We select ResNet, Swin Transformer [33], and ViTAE-v2 [60] for comparison. All the models are pre-trained with the mixture data as listed in Row #2 of Tab. 3. Compared with ResNet-50, ViTAE-v2-S outperforms it by a large margin on the end-to-end spotting task, *i.e.*, 2.14 % on the 'None' setting. Compared with ResNet-101, Swin-S achieves a gain of 1.15% on the 'None' setting. We conjecture that the domain gap between large-scale synthetic data and small-scale real data might impact the performance of different backbones. It deserves further exploration to carefully tune the training configuration and study the training paradigm for text spotting.

4.3. Comparison with State-of-the-art Methods

Results on Total-Text. To evaluate the effectiveness of DeepSolo on scene text with arbitrary shape, we compare our model with state-of-the-art methods in Tab. 5. The metrics in the end-to-end setting are the primary ones for text spotting. **1)** Considering the 'None' results, with only Synth150K as the external data, our method surpasses previous methods except GLASS. Compared with other Transformer-based methods, DeepSolo significantly outperforms TESTR, SwinTextSpotter, and SPTS by 5.5%, 4.5%, and 4.6%, respectively. DeepSolo also outperforms TTS by 0.6% while using far less training data. **2)** With additional MLT17, IC13, and IC15 real data, DeepSolo achieves 79.7% in the 'None' setting, which is comparable with the 79.9% performance of GLASS. Note that DeepSolo runs faster than GLASS and there is no elaborately tailored module for recognition, *e.g.*, the Global to Local Attention Feature Fusion and the external recognizer in GLASS, while we only use a simple linear layer for recognition output. **3)** When using TextOCR, our method achieves very promising spotting performance, *i.e.*, 82.5% and 88.7% at the 'None' and 'Full' settings. With ViTAEv2-S, the results are further promoted.

Results on ICDAR 2015. We conduct experiments on ICDAR 2015 to verify the effectiveness of DeepSolo on multi-oriented scene text, as presented in Tab. 6. The results show that DeepSolo achieves decent performance among the comparing methods. Specifically, compared with Transformer-based methods using the same training datasets, DeepSolo surpasses SwinTextSpotter and SPTS by 6.4% and 11.1% on the E2E setting with the generic lexicon. With TextOCR, DeepSolo (ResNet-50) achieves the

Method	External Data	Detection			None	Full	FPS (report)	FPS (A100)
		P	R	F1				
TextDragon [11]	Synth800K	85.6	75.7	80.3	48.8	74.8	–	–
CharNet [53] *	Synth800K	88.6	81.0	84.6	63.6	–	–	–
TextPerceptron [42]	Synth800K	88.8	81.8	85.2	69.7	78.3	–	–
CRAFTS [1] *	Synth800K+IC13	89.5	85.4	87.4	78.7	–	–	–
Boundary [48]	Synth800K+IC13+IC15	88.9	85.0	87.0	65.0	76.1	–	–
Mask TextSpotter v3 [25]	Synth800K+IC13+IC15+SCUT	–	–	–	71.2	78.4	–	–
PGNet [49]	Synth800K+IC15	85.5	86.8	86.1	63.1	–	35.5	–
MANGO [41] *	Synth800K+Synth150K+COCO-Text+MLT19+IC13+IC15	–	–	–	72.9	83.6	4.3	–
PAN++ [51]	Synth800K+COCO-Text+MLT17+IC15	–	–	–	68.6	78.6	21.1	–
ABCNet v2 [31]	Synth150K+MLT17	90.2	84.1	87.0	70.4	78.1	10.0	14.9
SRSTS [52]	Synth800K+Synth150K+COCO-Text+MLT17+ArT19+IC15	92.0	83.0	87.2	78.8	86.3	18.7	–
GLASS [44]	Synth800K	90.8	85.5	88.1	79.9	86.2	3.0	–
TESTR [61]	Synth150K+MLT17	93.4	81.4	86.9	73.3	83.9	5.3	12.1
SwinTextSpotter [17]	Synth150K+MLT17+IC13+IC15	–	–	88.0	74.3	84.1	–	2.9
SPTS [39]	Synth150K+MLT17+IC13+IC15	–	–	–	74.2	82.4	–	0.6
TTS (poly) [21]	Synth800K+COCO-Text+IC13+IC15+SCUT	–	–	–	78.2	86.3	–	–
DeepSolo (ResNet-50)	Synth150K	93.9	82.1	87.6	78.8	86.2	17.0	17.0
DeepSolo (ResNet-50)	Synth150K+MLT17+IC13+IC15	93.1	82.1	87.3	79.7	87.0	17.0	17.0
DeepSolo (ResNet-50)	Synth150K+MLT17+IC13+IC15+TextOCR	93.2	84.6	<u>88.7</u>	<u>82.5</u>	<u>88.7</u>	17.0	17.0
DeepSolo (ViTAEv2-S)	Synth150K+MLT17+IC13+IC15+TextOCR	92.9	87.4	**90.0**	**83.6**	**89.6**	10.0	10.0

Table 5. Performance on Total-Text. '*': character-level annotations are used. '*' has the same meaning for other tables.

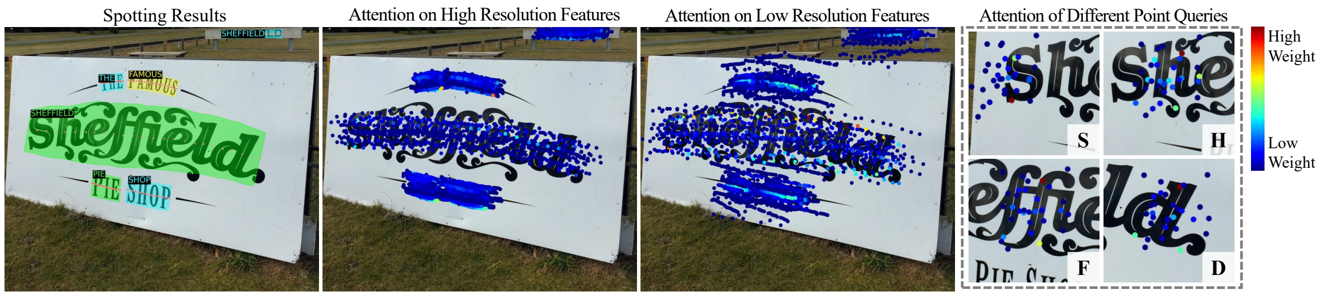
Spotting Results
Attention on High Resolution Features
Attention on Low Resolution Features
Attention of Different Point Queries
SHEFFIELD
THE
FAMOUS
PIE
SHOP
S
H
F
D
High Weight
Low Weight

Figure 4. Visualization of the spotting results, attention on different scale features, and attention of point queries.

'S', 'W', and 'G' metrics of 88.0%, 83.5%, and 79.1%.

Results on CTW1500. In Tab. 7, pre-trained with Synth150K, DeepSolo with the maximum recognition length of 25 already outperforms most of the previous approaches on the 'None' metric. We further increase the number of point queries from 25 to 50 for each text instance, achieving absolute 3.1% improvement on the 'None' result, without obvious sacrifice on inference speed. With MLT17, IC13, IC15 and Total-Text as the additional datasets, DeepSolo presents 64.2% performance without the help of lexicons, being 0.6% better and 25 times faster than SPTS.

4.4. Visual Analysis

Fig. 4 visualizes the spotting results, the attention on different scale features, and the attention of different point queries. It shows that DeepSolo is capable of correctly recognizing scene texts of large size variance. In the rightest figure, it is noteworthy that point queries highly attend to the discriminative extremities of characters, which indicates that point queries can effectively encode the character position, scale, and semantic information. More qualitative results are presented in Fig. 5.

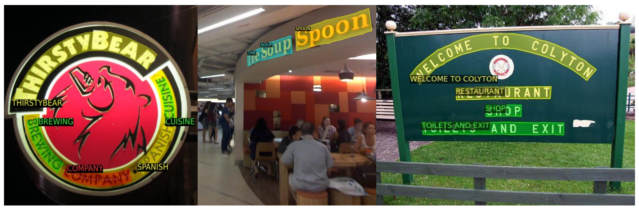

Figure 5. Visualizations on Total-Text, IC15 and CTW1500.

4.5. Compatibility to Line Annotations

DeepSolo can not only adapt to polygon annotations but also line annotations, which are much easier to obtain. We conduct experiments on Total-Text as the target dataset with

Method	External Data	Detection			E2E			Word Spotting		
		P	R	F1	S	W	G	S	W	G
TextDragon [11]	Synth800K	92.5	83.8	87.9	82.5	78.3	65.2	86.2	81.6	68.0
CharNet [53] *	Synth800K	91.2	88.3	89.7	80.1	74.5	62.2	–	–	–
TextPerceptron [42]	Synth800K	92.3	82.5	87.1	80.5	76.6	65.1	84.1	79.4	67.9
CRAFTS [1] *	Synth800K+IC13	89.0	85.3	87.1	83.1	82.1	74.9	–	–	–
Boundary [48]	Synth800K+IC13+Total-Text	89.8	87.5	88.6	79.7	75.2	64.1	–	–	–
Mask TextSpotter v3 [25]	Synth800K+IC13+Total-Text+SCUT	–	–	–	83.3	78.1	74.2	83.1	79.1	75.1
PGNet [49]	Synth800K+Total-Text	91.8	84.8	88.2	83.3	78.3	63.5	–	–	–
MANGO [41] *	Synth800K+Synth150K+COCO-Text+MLT19+IC13+Total-Text	–	–	–	85.4	80.1	73.9	85.2	81.1	74.6
PAN++ [51]	Synth800K+COCO-Text+MLT17+Total-Text	–	–	–	82.7	78.2	69.2	–	–	–
ABCNet v2 [31]	Synth150K+MLT17	90.4	86.0	88.1	82.7	78.5	73.0	–	–	–
SRSTS [52]	Synth800K+Synth150K+COCO-Text+MLT17	96.1	82.0	88.4	85.6	81.7	74.5	85.8	82.6	76.8
GLASS [44]	Synth800K	86.9	84.5	85.7	84.7	80.1	76.3	86.8	82.5	78.8
TESTR [61]	Synth150K+MLT17+Total-Text	90.3	89.7	90.0	85.2	79.4	73.6	–	–	–
SwinTextSpotter [17]	Synth150K+MLT17+IC13+Total-Text	–	–	–	83.9	77.3	70.5	–	–	–
SPTS [39]	Synth150K+MLT17+IC13+Total-Text	–	–	–	77.5	70.2	65.8	–	–	–
TTS [21]	Synth800K+COCO-Text+IC13+Total-Text+SCUT	–	–	–	85.2	81.7	77.4	85.0	81.5	77.3
DeepSolo (ResNet-50)	Synth150K+MLT17+IC13+Total-Text	92.8	87.4	90.0	86.8	81.9	76.9	86.3	82.3	77.3
DeepSolo (ResNet-50)	Synth150K+MLT17+IC13+Total-Text+TextOCR	92.5	87.2	89.8	88.0	83.5	79.1	87.3	83.8	79.5
DeepSolo (ViTAEv2-S)	Synth150K+MLT17+IC13+Total-Text+TextOCR	92.4	87.9	**90.1**	**88.1**	**83.9**	**79.5**	**87.8**	**84.5**	**80.0**

Table 6. Performance on ICDAR2015. ‘S’, ‘W’ and ‘G’ refer to using strong, weak and generic lexicons.

Method	None	Full	FPS
ABCNet v2 (100 length) [31]	57.5	77.2	10.0
MANGO (25 length) [41]	58.9	78.7	8.4
SwinTextSpotter [17]	51.8	77.0	–
TESTR (100 length) [61]	56.0	81.5	15.9 †
SPTS (100 length) [39]	63.6	**83.8**	0.8 †
DeepSolo (25 length, Synth150K)	60.1	78.4	20.0 †
DeepSolo (50 length, Synth150K)	63.2	80.0	20.0 †
DeepSolo (50 length)	**64.2**	81.4	20.0 †

Table 7. End-to-end recognition performance on CTW1500. ResNet-50 is adopted in DeepSolo. ‘length’: the maximum recognition length. ‘†’: the FPS measured on one A100 GPU.

only line annotations. To take the advantage of existing full annotations, we first pre-train the model on a mixture of Synth150K, MLT17, IC13, IC15, and TextOCR. Then, we simply exclude the boundary head and fine-tune the model on Total-Text with IC13 and IC15 for 6K steps, using only the text center line annotations. During fine-tuning, the random crop augmentation which needs box information is discarded. We use the evaluation protocol provided by SPTS [39]. To further study the sensitivity to the line location, we randomly shift the center line annotations to the boundary and shrink them to the center point at different levels to simulate annotation errors. Results are plotted in Fig. 6. It can achieve 81.6% end-to-end (‘None’) performance, which is comparable with the fully supervised model, *i.e.*, 82.5%. As can be seen in Fig. 6, the model is robust to the shift from 0% to 50% and the shrinkage from 0% to 20%. It indicates that the center line should not be too close to the text boundaries and better cover the complete character area.

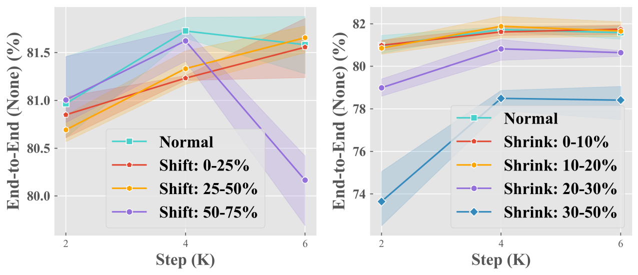

Figure 6. Analysis of the sensitivity to different line locations.

5. Conclusion

In this paper, we propose DeepSolo, a succinct DETR-like baseline, taking the merit of our novel explicit point query form that provides a joint and pivotal representation for detection and recognition. With a single decoder and several simple prediction heads, we present a much simpler text spotter compared with previous related methods. Experiments demonstrate DeepSolo has achieved state-of-the-art performance, high training efficiency, and compatibility with different annotations. We hope DeepSolo can serve as a strong baseline and inspire more follow-up works.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China under Grants 62076186 and 62225113, and in part by the Science and Technology Major Project of Hubei Province (Next-Generation AI Technologies) under Grant 2019AEA170. Dr. Jing Zhang was supported by Australian Research Council Projects in part by FL170100117 and IH180100002. The numerical calculations have been done on the system in the Supercomputing Center of Wuhan University.

References

[1] Youngmin Baek, Seung Shin, Jeonghun Baek, Sungrae Park, Junyeop Lee, Daehyun Nam, and Hwalsuk Lee. Character region attention for text spotting. In *ECCV*, 2020. 7, 8

[2] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *PAMI*, 11(6):567–585, 1989. 2

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1, 2

[4] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. In *ICLR*, 2021. 2

[5] Xiaoxue Chen, Lianwen Jin, Yuanzhi Zhu, Canjie Luo, and Tianwei Wang. Text recognition in the wild: A survey. *ACM Computing Surveys (CSUR)*, 54(2):1–35, 2021. 1

[6] Chee-Kheng Ch'ng, Chee Seng Chan, and Cheng-Lin Liu. Total-text: toward orientation robustness in scene text detection. *IJDAR*, 23(1):31–52, 2020. 5

[7] Guilherme N DeSouza and Avinash C Kak. Vision for mobile robot navigation: A survey. *PAMI*, 24(2):237–267, 2002. 1

[8] Qi Dong, Zhuowen Tu, Haofu Liao, Yuting Zhang, Vijay Mahadevan, and Stefano Soatto. Visual relationship detection using part-and-sum transformers with composite queries. In *ICCV*, 2021. 4

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 1

[10] Bo Du, Jian Ye, Jing Zhang, Juhua Liu, and Dacheng Tao. I3cl: Intra-and inter-instance collaborative learning for arbitrary-shaped scene text detection. *IJCV*, 130(8):1961–1977, 2022. 1

[11] Wei Feng, Wenhao He, Fei Yin, Xu-Yao Zhang, and Cheng-Lin Liu. Textdragon: An end-to-end framework for arbitrary shaped text spotting. In *ICCV*, 2019. 2, 7, 8

[12] Zhengyang Feng, Shaohua Guo, Xin Tan, Ke Xu, Min Wang, and Lizhuang Ma. Rethinking efficient lane detection via curve modeling. In *CVPR*, 2022. 5

[13] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006. 4

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[16] Tong He, Zhi Tian, Weilin Huang, Chunhua Shen, Yu Qiao, and Changming Sun. An end-to-end textspotter with explicit alignment and attention. In *CVPR*, 2018. 1, 2

[17] Mingxin Huang, Yuliang Liu, Zhenghao Peng, Chongyu Liu, Dahua Lin, Shenggao Zhu, Nicholas Yuan, Kai Ding, and Lianwen Jin. Swintextspotter: Scene text spotting via better synergy between text detection and text recognition. In *CVPR*, 2022. 1, 2, 5, 7, 8

[18] Ding Jia, Yuhui Yuan, Haodi He, Xiaopei Wu, Haojun Yu, Weihong Lin, Lei Sun, Chao Zhang, and Han Hu. Detrs with hybrid matching. *arXiv preprint arXiv:2207.13080*, 2022. 3

[19] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *ICDAR*, 2015. 5

[20] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere De Las Heras. Icdar 2013 robust reading competition. In *ICDAR*, 2013. 5

[21] Yair Kittenplon, Inbal Lavi, Sharon Fogel, Yarin Bar, R Manmatha, and Pietro Perona. Towards weakly-supervised text spotting using a multi-task transformer. In *CVPR*, 2022. 1, 2, 4, 7, 8

[22] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 4

[23] Zhiqi Li, Wenhai Wang, Enze Xie, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, Ping Luo, and Tong Lu. Panoptic segformer: Delving deeper into panoptic segmentation with transformers. In *CVPR*, 2022. 1

[24] Minghui Liao, Pengyuan Lyu, Minghang He, Cong Yao, Wenhao Wu, and Xiang Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. *PAMI*, 43(2):532–548, 2021. 2

[25] Minghui Liao, Guan Pang, Jing Huang, Tal Hassner, and Xiang Bai. Mask textspotter v3: Segmentation proposal network for robust scene text spotting. In *ECCV*, 2020. 1, 2, 7, 8

[26] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 4

[27] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. Dab-detr: Dynamic anchor boxes are better queries for detr. In *ICLR*, 2022. 4

[28] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. Fots: Fast oriented text spotting with a unified network. In *CVPR*, 2018. 1, 2

[29] Yuliang Liu, Hao Chen, Chunhua Shen, Tong He, Lianwen Jin, and Liangwei Wang. Abcnet: Real-time scene text spotting with adaptive bezier-curve network. In *CVPR*, 2020. 1, 2, 3, 5

[30] Yuliang Liu, Lianwen Jin, Shuaitao Zhang, Canjie Luo, and Sheng Zhang. Curved scene text detection via transverse and longitudinal sequence connection. *PR*, 90:337–345, 2019. 5

[31] Yuliang Liu, Chunhua Shen, Lianwen Jin, Tong He, Peng Chen, Chongyu Liu, and Hao Chen. Abcnet v2: Adaptive bezier-curve network for real-time end-to-end text spotting. *PAMI*, 2021. 1, 2, 3, 7, 8

[32] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al.

Swin transformer v2: Scaling up capacity and resolution. In *CVPR*, 2022. 1

[33] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 1, 6

[34] Shangbang Long, Xin He, and Cong Yao. Scene text detection and recognition: The deep learning era. *IJCV*, 129(1):161–184, 2021. 1

[35] George G Lorentz. *Bernstein polynomials*. American Mathematical Soc., 2013. 3

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5

[37] Pengyuan Lyu, Minghui Liao, Cong Yao, Wenhao Wu, and Xiang Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. In *ECCV*, 2018. 1, 2

[38] Nibal Nayef, Fei Yin, Imen Bizid, Hyunsoo Choi, Yuan Feng, Dimosthenis Karatzas, Zhenbo Luo, Umapada Pal, Christophe Rigaud, Joseph Chazalon, et al. Icdar2017 robust reading challenge on multi-lingual scene text detection and script identification-rrc-mlt. In *ICDAR*, 2017. 5

[39] Dezhi Peng, Xinyu Wang, Yuliang Liu, Jiaxin Zhang, Mingxin Huang, Songxuan Lai, Jing Li, Shenggao Zhu, Dahua Lin, Chunhua Shen, et al. Spts: Single-point text spotting. In *ACM MM*, 2022. 1, 2, 5, 7, 8

[40] Zhiliang Peng, Wei Huang, Shanzhi Gu, Lingxi Xie, Yaowei Wang, Jianbin Jiao, and Qixiang Ye. Conformer: Local features coupling global representations for visual recognition. In *ICCV*, 2021. 1

[41] Liang Qiao, Ying Chen, Zhanzhan Cheng, Yunlu Xu, Yi Niu, Shiliang Pu, and Fei Wu. Mango: A mask attention guided one-stage scene text spotter. In *AAAI*, 2021. 7, 8

[42] Liang Qiao, Sanli Tang, Zhanzhan Cheng, Yunlu Xu, Yi Niu, Shiliang Pu, and Fei Wu. Text perceptron: Towards end-to-end arbitrary-shaped text spotting. In *AAAI*, 2020. 2, 7, 8

[43] Siyang Qin, Alessandro Bissacco, Michalis Raptis, Yasuhisa Fujii, and Ying Xiao. Towards unconstrained end-to-end text spotting. In *ICCV*, 2019. 1

[44] Roi Ronen, Shahar Tsiper, Oron Anschel, Inbal Lavi, Amir Markovitz, and R Manmatha. Glass: global to local attention for scene-text spotting. In *ECCV*, 2022. 1, 2, 7, 8

[45] Amanpreet Singh, Guan Pang, Mandy Toh, Jing Huang, Wojciech Galuba, and Tal Hassner. Textocr: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text. In *CVPR*, 2021. 5

[46] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 1

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1

[48] Hao Wang, Pu Lu, Hui Zhang, Mingkun Yang, Xiang Bai, Yongchao Xu, Mengchao He, Yongpan Wang, and Wenyu Liu. All you need is boundary: Toward arbitrary-shaped text spotting. In *AAAI*, 2020. 2, 7, 8

[49] Pengfei Wang, Chengquan Zhang, Fei Qi, Shanshan Liu, Xiaoqiang Zhang, Pengyuan Lyu, Junyu Han, Jingtuo Liu, Errui Ding, and Guangming Shi. Pgnet: Real-time arbitrarily-shaped text spotting with point gathering network. In *AAAI*, 2021. 1, 2, 7, 8

[50] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021. 1

[51] Wenhai Wang, Enze Xie, Xiang Li, Xuebo Liu, Ding Liang, Zhibo Yang, Tong Lu, and Chunhua Shen. Pan++: Towards efficient and accurate end-to-end spotting of arbitrarily-shaped text. *PAMI*, 44(9):5349–5367, 2021. 1, 7, 8

[52] Jingjing Wu, Pengyuan Lyu, Guangming Lu, Chengquan Zhang, Kun Yao, and Wenjie Pei. Decoupling recognition from detection: Single shot self-reliant scene text spotter. In *ACM MM*, 2022. 2, 7, 8

[53] Linjie Xing, Zhi Tian, Weilin Huang, and Matthew R Scott. Convolutional character networks. In *ICCV*, 2019. 1, 2, 7, 8

[54] Yufei Xu, Qiming Zhang, Jing Zhang, and Dacheng Tao. Vitae: Vision transformer advanced by exploring intrinsic inductive bias. In *NeurIPS*, 2021. 1

[55] Maoyuan Ye, Jing Zhang, Shanshan Zhao, Juhua Liu, Bo Du, and Dacheng Tao. Dptext-detr: Towards better scene text detection with dynamic points in transformer. In *AAAI*, 2023. 2, 3, 4

[56] Qixiang Ye and David Doermann. Text detection and recognition in imagery: A survey. *PAMI*, 37(7):1480–1500, 2014. 1

[57] Chongsheng Zhang, Yuefeng Tao, Kai Du, Weiping Ding, Bin Wang, Ji Liu, and Wei Wang. Character-level street view text spotting based on deep multisegmentation network for smarter autonomous driving. *IEEE Transactions on Artificial Intelligence*, 3(2):297–308, 2021. 1

[58] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *ICLR*, 2023. 3

[59] Jing Zhang and Dacheng Tao. Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things. *IEEE Internet of Things Journal*, 8(10):7789–7817, 2020. 1

[60] Qiming Zhang, Yufei Xu, Jing Zhang, and Dacheng Tao. Vitaev2: Vision transformer advanced by exploring inductive bias for image recognition and beyond. *IJCV*, 2022. 1, 6

[61] Xiang Zhang, Yongwen Su, Subarna Tripathi, and Zhuowen Tu. Text spotting transformers. In *CVPR*, 2022. 1, 2, 3, 4, 5, 7, 8

[62] Humen Zhong, Jun Tang, Wenhai Wang, Zhibo Yang, Cong Yao, and Tong Lu. Arts: Eliminating inconsistency between text detection and recognition with auto-rectification text spotter. *arXiv preprint arXiv:2110.10405*, 2021. 1, 2

[63] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 2, 3, 6