FAST: Faster Arbitrarily-Shaped Text Detector with Minimalist Kernel Representation

Zhe Chen, Jiahao Wang, Wenhai Wang, Guo Chen, Enze Xie, Ping Luo, Tong Lu

***Abstract*—We propose an accurate and efficient scene text detection framework, termed FAST (*i.e.*, faster arbitrarily-shaped text detector). Different from recent advanced text detectors that used complicated post-processing and hand-crafted network architectures, resulting in low inference speed, FAST has two new designs. (1) We design a minimalist kernel representation (only has 1-channel output) to model text with arbitrary shape, as well as a GPU-parallel post-processing to efficiently assemble text lines with a negligible time overhead. (2) We search the network architecture tailored for text detection, leading to more powerful features than most networks that are searched for image classification. Benefiting from these two designs, FAST achieves an excellent trade-off between accuracy and efficiency on several challenging datasets, including Total Text, CTW1500, ICDAR 2015, and MSRA-TD500. For example, FAST-T yields 81.6% F-measure at 152 FPS on Total-Text, outperforming the previous fastest method by 1.7 points and 70 FPS in terms of accuracy and speed. With TensorRT optimization, the inference speed can be further accelerated to over 600 FPS. Code and models will be released at https://github.com/czczup/FAST.**

***Index Terms*—Arbitrarily-Shaped Text Detector, Real-Time Text Detection, Minimalist Kernel Representation, GPU-Parallel Post-Processing.**

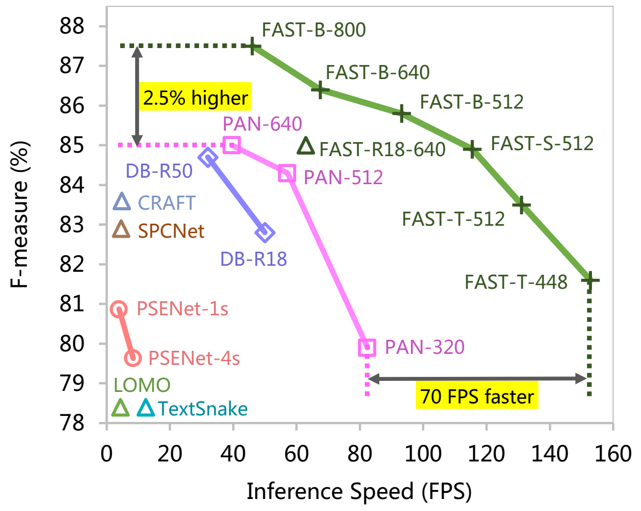


Fig. 1. Text detection F-measure and inference speed of different text detectors on Total-Text [8] dataset. Our FAST models, namely FAST-T/S/B according to the model size, enjoy faster inference speed and better accuracy than state-of-the-art counterparts, such as PAN [4] and DB [1].

I. Introduction

Scene text detection is a fundamental task in computer vision with wide practical applications, such as image understanding, instant translation, and autonomous driving. With the remarkable progress of deep learning, a considerable amount of methods [1]–[6] have been proposed to detect text with arbitrary shape, and the performance on public datasets is constantly being refreshed. However, we argue that the above methods still have room to improve due to two main sub-optimal designs: (1) low-efficient post-processing and (2) hand-crafted network architecture.

First, the post-processing of previous works usually takes about 30% of the whole inference time [1], [3], [4]. Moreover, these post-processing approaches are designed to run on the CPU (see Fig. 2), which are difficult to parallel with GPU resources, resulting in relatively low efficiency. In general, the post-processing is closely related to the text representation method [1]–[4], [7], which determines whether it can be optimized to achieve GPU parallelism. Therefore, it is important to develop a GPU-friendly representation method with a parallelable post-processing for the real-time text detector.

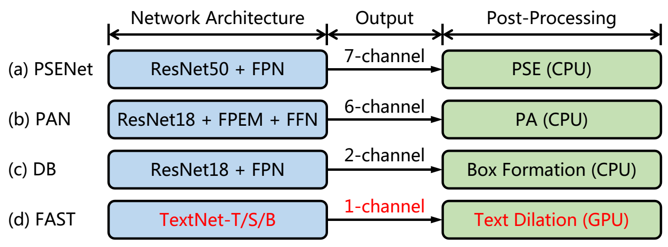
Second, most existing text detectors adopt a heavy hand-crafted backbone (*e.g.*, ResNet50 [9]) to achieve excellent performance, but at the expense of inference speed to some extent. For high efficiency, some methods [1], [4] developed text detectors based on ResNet18 [9], but the backbone is originally designed for image classification and may not be the best choice for text detection. Although many auto-searched lightweight networks [10]–[13] have been presented, they only focus on image classification or general object detection, and the application to text detection is rarely considered. Consequently, how to design an efficient and powerful network specific to text detection, is a topic worth exploring.

In this work, we propose an efficient and powerful text detection framework, termed FAST (**F**aster **A**rbitrary-**S**haped **T**ext detector). As illustrated in Fig. 2, FAST contains the following two main improvements to achieve high efficiency: (1) We propose a minimalist kernel representation (MKR) that formulates a text line as an eroded text region surrounded by peripheral pixels. Compared to existing kernel representations [1], [3], [4], our MKR not only benefits the network to predict a 1-channel output, but also enjoys a GPU-parallel post-processing—text dilation. (2) We carefully

Zhe Chen, Jiahao Wang, Guo Chen, and Tong Lu are with the Department of Computer Science and Technology, Nanjing University, Jiangsu, P. R. China (E-mail: chenzhe98@smail.nju.edu.cn; wangjh@smail.nju.edu.cn; chenguo1177@gmail.com; lutong@nju.edu.cn).

Wenhai Wang is with the Shanghai AI Laboratory, Shanghai, P. R. China (E-mail: wangwenhai@pjlab.org.cn).

Enze Xie and Ping Luo are with the Department of Electronic Engineer, The Chinese University of Hong Kong, Hong Kong, P. R. China (E-mail: Johnny_ez@163.com; pluo@ie.cuhk.edu.hk).



Method	F(%)	Time Cost (ms)		FPS
		Network	Post-Proc.	
PSENet-1s [3]	80.9	118.0	145.0	3.9
PAN-512 [4]	84.3	13.7	3.5	57.1
DB-R18 [1]	82.8	15.3	4.7	50.0
FAST-S-512 (ours)	**84.9**	**7.5**	**1.1**	**115.5**

Fig. 2. Overall pipelines of representative arbitrarily-shaped text detectors. "Post-Proc." is short for post-processing. Our FAST achieves significantly faster inference speed than previous methods [1], [3], [4], benefiting from (1) the minimalist kernel representation (MKR) with a GPU-parallel post-processing method—text deliation, and (2) the efficient TextNet architecture specifically searched for text detection.

design a NAS search space and reward function for the text detection task. The searched efficient backbones are named TextNet, which can provide more powerful features for text detection than the network searched on image classification (*e.g.*, MobileNetV3 [12]). Combining the advantages of these designs, our method achieves an excellent trade-off between accuracy and inference speed.

To demonstrate the effectiveness of our FAST, we conduct extensive experiments on four challenging benchmarks, including Total-Text [8], CTW1500 [14], ICDAR 2015 [15], and MSRA-TD500 [16]. According to the model size, we name our text detectors FAST-Tiny/Small/Base (short for FAST-T/S/B), respectively. As shown in Fig. 1, on the Total-Text dataset, FAST-T-448, which means scaling the shorter side of input images to 448 pixels, achieves 81.6% F-measure at 152.8 FPS, being 1.7% F-measure higher and 70 FPS faster than the previous fastest method PAN-320 [4]. Besides, our best model FAST-B-800 achieves 87.5% F-measure while still keeping a real-time speed (46.0 FPS).

In summary, our contributions are as follows:

(1) We develop an accurate and efficient arbitrarily-shaped text detector, termed FAST, which is completely GPU-parallel, in terms of post-processing and network architecture.

(2) We propose a minimalist kernel representation (MKR) with a GPU-parallel post-processing—text dilation, significantly reducing its time overhead.

(3) We design a NAS search space and reward function specifically for text detection, and search for a series of backbone networks (*i.e.*, TextNet) friendly to text detection with different inference speeds.

(4) Our FAST-T model achieves an astonishing speed of 152.8 FPS while maintaining competitive accuracy on Total-Text. With TensorRT [17] optimization, it can be further accelerated to over 600 FPS.

II. Related Work

A. Scene Text Detection

Inspired by general object detection methods [18]–[20], many methods [21]–[27] have been proposed to detect horizontal and multi-oriented text. For instance, Tian *et al.* [27] presented the CTPN, successfully transferred object detection frameworks for horizontal text detection, and obtained promising results. Some researchers [21], [23], [26] have considered the orientations of text lines and designed various methods to detect multi-oriented text. However, most of them fail to locate curved text accurately.

To remedy this defect, recent methods cast the text detection task as a segmentation problem. For example, TextSnake [2] designed a flexible representation for scene text, which described a text instance as a sequence of ordered and overlapping disks centered at symmetric axes. PixelLink [28] separated adjacent text lines by performing text/non-text prediction and link prediction at the pixel level. SPCNet [29] and Mask TextSpotter [30] are designed to detect arbitrarily-shaped text in an instance segmentation manner. SAE [31] introduced a shape-aware loss and new cluster post-processing to distinguish adjacent text lines with various aspect ratios and small gaps. PSENet [3] proposed to present text instances via text kernels, and developed the progressive scale expansion (PSE) algorithm to merge multi-scale text kernels. Although the above methods achieve excellent performance, most of them run at a slow inference speed due to the cumbersome post-processing approaches and complicated network architectures.

B. Real-time Text Detection

With the growing demand of real-time applications, efficient text detection attracts increasing attention. EAST [26] applied a fully convolutional network (FCN) to directly produce rotated rectangles or quadrangles for text regions, which is the first text detector that runs at 20 FPS. PAN [4] and DB [1] are two representative real-time text detectors, both of which adopted a lightweight backbone (*i.e.*, ResNet18 [9]) to speed up inference. For post-processing, PAN developed a learnable post-processing algorithm, namely pixel aggregation (PA), to improve the accuracy by using the predicted similarity vectors. DB proposed the box formation process, which utilized the Vatti clipping algorithm [32] to dilate the predicted text kernels. Recently, PAN++ [33] and DB++ [34] extended their previous methods [1], [4] and obtained improved detection performance. Although these methods have simplified the text detection pipeline compared to previous methods [2], [3], [26], [29], real-time text detection is still room for improvement, due to CPU-based post-processing and sub-optimal hand-crafted network architecture.

C. Neural Architecture Design

Network architecture design is an ongoing research topic in the field of computer vision [35]–[37]. For the text detection task, most of the existing methods [1], [3], [4], [26], [27] adopted the hand-crafted backbone networks, such as VGG [38] and ResNet [9], but these backbones are originally

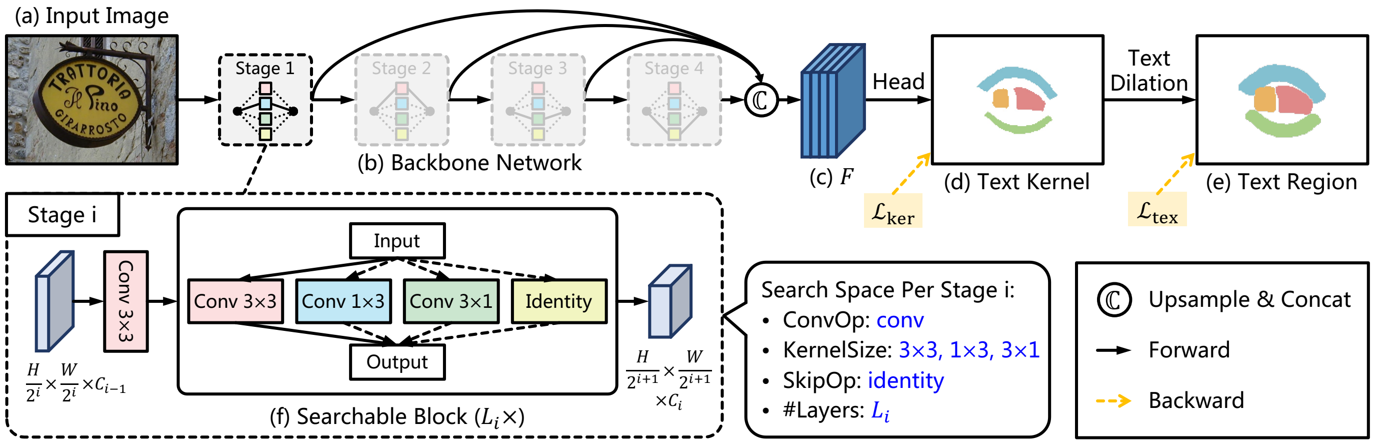


Fig. 3. Overall architecture of FAST. The backbone network is divided into four stages, each of which contains L_i searchable blocks for the architecture search of text detection. The multi-scale features from the backbone are upsampled and concatenated as the final feature map F, which is used to predict text kernels via a lightweight head [4] of 2-layer convolutions. The GPU-parallel post-processing—text dilation, is applied to reconstruct complete text lines.

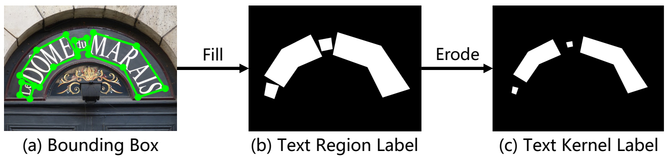


Fig. 4. Label generation of the minimalist kernel representation (MKR). For a given image, the text region label can be generated by filling the bounding boxes. Then, applying an erosion operator to it, we can obtain the text kernel label. We use both these labels to supervise our text detectors.

designed for image classification and may not be the best choice for text detection. Recently, owing to the neural architecture search (NAS) techniques, there has been a significant change in designing neural networks. Many auto-searched efficient networks, such as Proxyless [11], EfficientNet [13], OFA [10], and MobileNetV3 [12], play increasingly important roles in industry and the research community. Despite these developments, these NAS-based models are mainly limited to a few tasks, such as image classification and general object detection, leading to weak generalization ability in other tasks. To compensate for these drawbacks, many researchers explore applying NAS methods to their specific fields, including semantic segmentation [39], pose estimation [40], and scene text recognition [41], [42], etc. However, there is still rare to extend NAS approaches to text detection.

III. Proposed Method

A. Overall Architecture

As illustrated in Fig. 3, the proposed FAST contains (1) a GPU-parallel post-processing—text dilation, to rebuild complete text lines from predicted text kernels; and (2) a backbone network with multiple searchable blocks for architecture search of text detection.

In the inference phase, we first feed the input image of $H\times W\times 3$ into the backbone, and obtain multi-scale features, which are 1/4, 1/8, 1/16, 1/32 of the original image resolution. Then, we reduce the dimension of each feature map to 128 via 3×3 convolution, and these feature maps are upsampled

Algorithm 1 PyTorch-like Pseudo Code of Text Dilation

```
# s: dilation size

# text dilation for post-processing
def text_dilation(text_kernel, s):

    if not training: # in the inference phase
       # binarize text kernel
       text_kernel = text_kernel > 0
       # distinguish text kernels using the connected
           components labeling (CCL) algorithm
       text_kernel = ccl_gpu(text_kernel)

    # implement dilation operation with F.max_pool2d
    # args: input, kernel size, stride, padding
    text = F.max_pool2d(text_kernel, s, 1, s//2)

    return text
```

and concatenated via the function $\mathbb{C}(\cdot)$, to obtain the final feature map F, whose shape is $H/4\times W/4\times 512$ (see Fig. 3(c)). After that, the final feature map F passes through a lightweight head [4] of 2-layer convolutions to perform text kernel segmentation. Finally, we rebuild the complete text regions via the text dilation process with a negligible time overhead, as shown in Fig. 3(d) and Fig. 3(e).

During training, we use loss functions $\mathcal{L}_{\text{ker}}$ and $\mathcal{L}_{\text{tex}}$ to optimize the text kernel predicted by the network (see Fig. 3(d)) and the text region generated by post-processing (see Fig. 3(e)), respectively. During searching, we perform architecture search for text detection based on the widely-used search framework ProxylessNAS [11]. Specifically, we calculate rewards according to the segmentation accuracy and inference speed, and then use the reinforce-based strategy to optimize the network architecture. The processes of training and searching are performed in an alternative manner. When the architecture search is finished, we can prune redundant paths and obtain the final architecture.

B. Minimalist Kernel Representation

1) Definition: To simplify the post-processing, we propose a novel text representation approach termed minimalist kernel

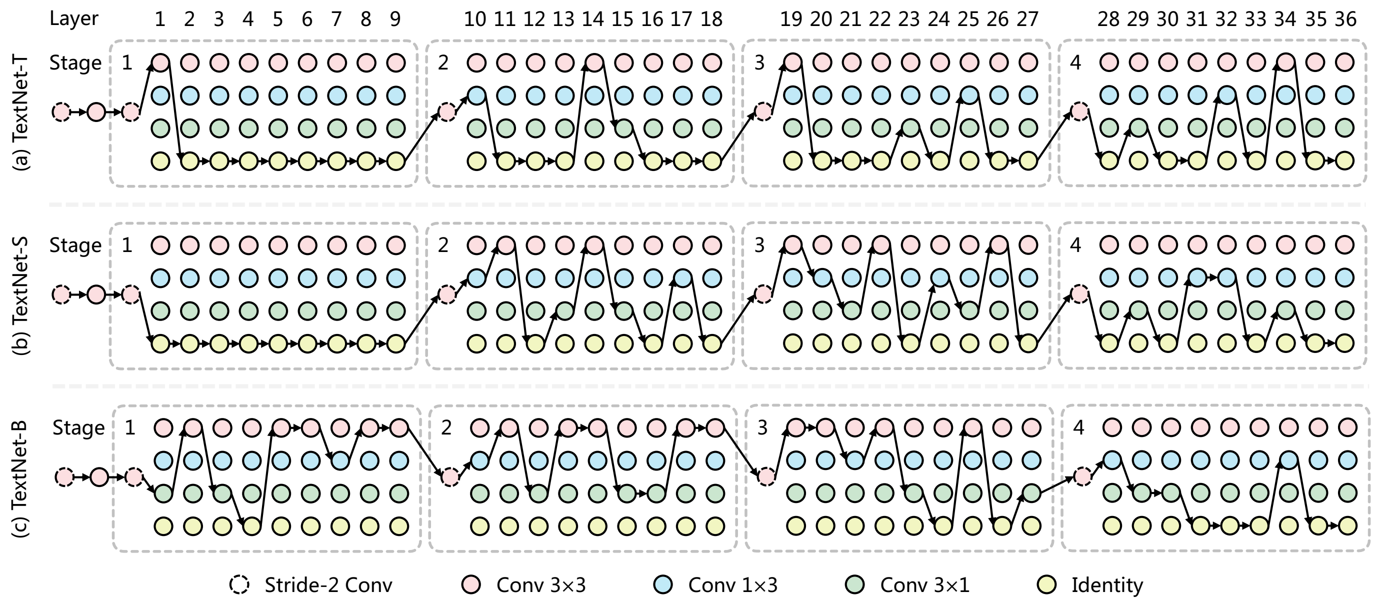


Fig. 5. Searched backbone architecture of TextNet in our proposed FAST. The four nodes in each column represent a searchable block, and the black arrows indicate the selected operations. After the architecture search, we can prune redundant paths and obtain the final architecture.

representation (MKR). As illustrated in Fig. 4, our MKR formulates a given text line as an eroded text region (*i.e.*, text kernel) with peripheral pixels. Compared to the existing kernel representations [1], [3], [4], our MKR has two main differences as follows.

Firstly, because our text kernel label is generated by the morphological erosion operation, it can be approximatively restored to the complete text region by the reverse operation (*i.e.*, dilation). Moreover, both erosion and dilation can be easily implemented in PyTorch with GPU acceleration.

Secondly, our MKR only requires the network to predict a 1-channel output, which is simpler than previous methods that need multi-channel output [1], [3], [4], as illustrated in Fig. 2. To our best knowledge, it may be the simplest kernel representation for arbitrarily-shaped text detection.

2) Label Generation: To learn this representation, we need to generate labels for text kernels and text regions. Specifically, for a given text image, the label of text regions can be directly produced by filling the bounding boxes, which is denoted as G_{tex} (see Fig. 4(b)). Note that G_{tex} is a binary image, applying an erosion operator with $s \times s$ kernel to G_{tex}, the peripheral pixels of text regions will be converted to non-text pixels. To avoid losing text instances due to the erosion operation, we keep at least a minimal text kernel for each text region. We take this result as the label for text kernels and denote it as G_{ker} (see Fig. 4(c)).

3) Post-Processing: Based on the proposed MKR, we develop a GPU-parallel post-processing, termed text dilation, to recover complete text lines with a negligible time overhead. The pseudo code is shown in Algorithm 1, in which we utilize the max-pooling function with $s \times s$ kernel to implement the dilation operator equivalently.

During training, for a given prediction of text kernels, we directly apply the dilation operator to rebuild whole text regions. Since this step is differentiable, we can supervise both text kernels and text regions for more accurate predictions, as shown in Fig. 3. In the inference phase, we first binarize the predicted text kernels, and implement a GPU-accelerated Connected Components Labeling (CCL) algorithm [43] to distinguish different text kernels. Finally, we apply the dilation operator to reconstruct the complete text lines.

C. Efficient TextNet for Text Detection

1) Search Space: Following ProxylessNAS [11], we build a backbone network for the architecture search of text detection. As shown in Fig. 3(f), each stage of the backbone network is comprised of a stride-2 convolution and L_i searchable blocks, where the 3×3 convolution with stride 2 is used to downsample the feature maps, and each searchable block consists of a set of candidate operations, from which the most appropriate one is selected as the final operation after architecture search. In pursuit of extreme speed, we use reparameterable convolutions [36] as the candidate operations, and merge them into plain convolutions without multi-branch topology during inference.

Specifically, we present a layer-level candidate set, defined as {`conv3×3`, `conv1×3`, `conv3×1`, `identity`}. As the 1×3 and 3×1 convolutions have asymmetric kernels and oriented structure priors, they may help to capture the features of extreme aspect-ratio and rotated text lines. In addition, the identity operator indicates that a layer is skipped, which is used to control the depth and inference speed of the network. In summary, because there are a total of $L = L_1+L_2+L_3+L_4$ searchable blocks, and each of them has four candidates, the size of the search space is 4^L.

2) Reward Function: In addition to the search space, we design a customized reward function $\mathcal{R}(\cdot)$, to search network

TABLE I
ABLATION STUDIES OF EACH PROPOSED COMPONENT IN OUR FAST. THE SHORTER SIDES OF IMAGES IN TOTAL-TEXT [8] AND ICDAR 2015 [15] ARE SET TO 640 AND 736 PIXELS, RESPECTIVELY.

Method	#Param	Backbone	Post-Processing	Total-Text		ICDAR 2015	
				F-measure	FPS	F-measure	FPS
Baseline	13.0M	ResNet18 [9]	Pixel Aggregation [4]	85.2	49.0	83.0	34.5
FAST-R18 (ours)	13.0M	ResNet18 [9]	Text Dilation	85.0 (-0.2)	62.9 (+13.9)	82.8 (-0.2)	41.4 (+6.9)
FAST-B (ours)	10.6M	TextNet-B	Text Dilation	**86.4** **(+1.2)**	**67.5** **(+18.5)**	**84.7** **(+1.7)**	**42.7** **(+8.2)**

architectures for real-time text detection. Specifically, given a model m, we define the reward function as:

$$\mathcal{R}(m) = (\mathrm{IoU}_{\mathrm{ker}}(m) + \alpha \mathrm{IoU}_{\mathrm{tex}}(m)) \times \left(\frac{\mathrm{FPS}(m)}{T}\right)^{w}, \quad (1)$$

where $\mathrm{IoU}_{\mathrm{ker}}(m)$ and $\mathrm{IoU}_{\mathrm{tex}}(m)$ denote the intersection-over-union (IoU) metric of the predicted text kernels and text regions, respectively. α is the coefficient of $\mathrm{IoU}_{\mathrm{tex}}(m)$, which is empirically set to 0.5. Besides that, $\mathrm{FPS}(m)$ means the inference speed of the entire text detector measured on the GPU with batch size 1, and T is the target inference speed. w is a hyper-parameter to balance the accuracy and inference speed, which is set to 0.1 following common practice [11].

3) Discussion: Our method is different from existing works on NAS in three main aspects:

- We introduce reparameterable asymmetric convolution [36], [44] into the search space, which has oriented structure priors that may help to capture the features of extreme aspect-ratio and rotated text lines. While most existing NAS methods adopt MBConv [45] as the block, which ignores the geometric characteristics of text lines, and is not efficient enough for GPU.
- We propose a specialized reward function, which considers both the performance of the text kernel and text region, achieving effective architecture search for text detection. While most previous reward functions [12], [13], [46], [47] are designed for image classification or general object detection, and are not suitable for arbitrarily-shaped text detection.
- In this work, we extend the search framework Proxyless-NAS [11] for text detection. We aim to design a faster real-time text detector by compressing the time cost of all components in the text detection pipeline, rather than developing a new search algorithm. We consider that whether the search algorithm needs to be redesigned for text detection, is an interesting topic that can be further explored in the future.

D. Loss Function

The loss function of our FAST can be formulated as:

$$\mathcal{L} = \mathcal{L}_{\mathrm{ker}} + \alpha \mathcal{L}_{\mathrm{tex}}, \quad (2)$$

where $\mathcal{L}_{\mathrm{ker}}$ and $\mathcal{L}_{\mathrm{tex}}$ are losses for text kernels and text regions. Following common practices [3], [4], we apply Dice loss [48] to supervise the network. Therefore, $\mathcal{L}_{\mathrm{ker}}$ and $\mathcal{L}_{\mathrm{tex}}$ can be expressed as follows:

$$\mathcal{L}_{\mathrm{ker}} = 1 - \frac{2\sum_{x,y} P_{\mathrm{ker}}(x,y)\, G_{\mathrm{ker}}(x,y)}{\sum_{x,y} P_{\mathrm{ker}}(x,y)^2 + \sum_{x,y} G_{\mathrm{ker}}(x,y)^2}, \quad (3)$$

$$\mathcal{L}_{\mathrm{tex}} = 1 - \frac{2\sum_{x,y} P_{\mathrm{tex}}(x,y)\, G_{\mathrm{tex}}(x,y)}{\sum_{x,y} P_{\mathrm{tex}}(x,y)^2 + \sum_{x,y} G_{\mathrm{tex}}(x,y)^2}, \quad (4)$$

where $P(x,y)$ and $G(x,y)$ represent the value of position (x,y) in the prediction and the ground-truth, respectively. In addition, we apply Online Hard Example Mining (OHEM) [49] to $\mathcal{L}_{\mathrm{tex}}$ to ignore simple non-text regions. α balances the importance of $\mathcal{L}_{\mathrm{ker}}$ and $\mathcal{L}_{\mathrm{tex}}$, which is set to 0.5 in our experiments.

IV. EXPERIMENTS

A. Datasets

Total-Text [8] is a challenging dataset for arbitrarily-shaped text detection, including horizontal, multi-oriented, and curved text lines. It contains 1,255 training and 300 testing images, all of which are labeled with polygons at the word level.
CTW1500 [14] is also a widely used dataset for arbitrarily-shaped text detection. It consists of 1,000 training images and 500 testing images. In this dataset, text lines are labeled with 14 points as polygons.
ICDAR 2015 [15] is one of the challenges of the ICDAR 2015 Robust Reading Competition. It focuses on multi-oriented text in natural scenes and contains 1,000 training images and 500 testing images. The text lines are labeled by quadrangles at the word level.
MSRA-TD500 [16] is a multi-lingual dataset that contains multi-oriented and long text lines. It has 300 training images and 200 testing images. Following the previous works [2], [26], [50], we include the 400 images of HUST-TR400 [51] as training data.
IC17-MLT [52] is a multi-language dataset that consists of 7,200 training images, 1,800 validation images, and 9,000 testing images. In this dataset, text lines are annotated with word-level quadrangles.

B. Implementation Details

1) Training Settings: Following previous methods [3], [29], [53], [54], we pre-train our models on IC17-MLT for 300 epochs, in which images are cropped and resized to 640×640 pixels. We then finetune the models for 300 epochs on Total-Text, and 600 epochs on the other three datasets. The dilation size s is set to 9 when the shorter side is 640 pixels in our experiments. All models are optimized by Adam [55] optimizer with batch size 16 on 4 1080Ti GPUs. We adopt a "poly" learning rate schedule with an initial learning rate of 1×10^{-3}. Training data augmentations include random scale, random crop, random flip, and random rotation.

TABLE II
ABLATION STUDIES OF THE EROSION/DILATION SIZE s. ACCORDING TO THESE RESULTS, WE SET THE EROSION/DILATION SIZE s TO 9 BY DEFAULT IN OUR EXPERIMENTS.

Dataset	$s=3$	$s=5$	$s=7$	$s=9$	$s=11$
Total-Text [8]	82.3	83.4	84.8	**85.0**	85.0
ICDAR 2015 [15]	51.5	70.6	80.1	**82.8**	82.6

TABLE III
TEXT DETECTION F-MEASURE AND INFERENCE SPEED OF DIFFERENT BACKBONES ON TOTAL-TEXT [8] DATASET, WHERE WE SCALE THE SHORTER SIDE OF IMAGES TO 640 PIXELS. OUR TEXTNET MODELS SIGNIFICANTLY OUTPERFORM EXISTING HAND-CRAFTED AND AUTO-SEARCHED NETWORKS. NOTE THAT THE #PARAM HERE DOES NOT INCLUDE THE CLASSIFICATION HEAD.

	Backbone	#Param	P	R	F	FPS
Hand-Crafted	ResNet18 [9]	11.3M	89.3	81.2	85.0	62.9
	ResNet50 [9]	23.6M	89.5	81.5	85.3	32.2
	ResNet101 [9]	42.6M	**90.9**	81.0	85.6	23.0
	VGG16 [38]	14.7M	88.0	81.4	84.6	25.5
	PVTv2-B0 [56]	3.4M	90.4	79.9	84.8	24.0
Auto-Searched	EfficientNet-B0 [13]	3.6M	90.2	81.4	85.6	39.4
	Proxyless-GPU [11]	4.6M	89.6	78.1	83.5	54.7
	OFANet-12ms [10]	3.5M	89.9	79.1	84.2	55.1
	MobileNetV3 [12]	3.0M	90.3	78.4	83.9	56.7
	GENet-Small [57]	6.2M	89.0	77.7	83.0	74.4
	TextNet-T (ours)	6.8M	87.1	81.4	84.2	**95.5**
	TextNet-S (ours)	8.0M	89.1	81.9	85.4	85.3
	TextNet-B (ours)	8.9M	89.9	**83.2**	**86.4**	67.5

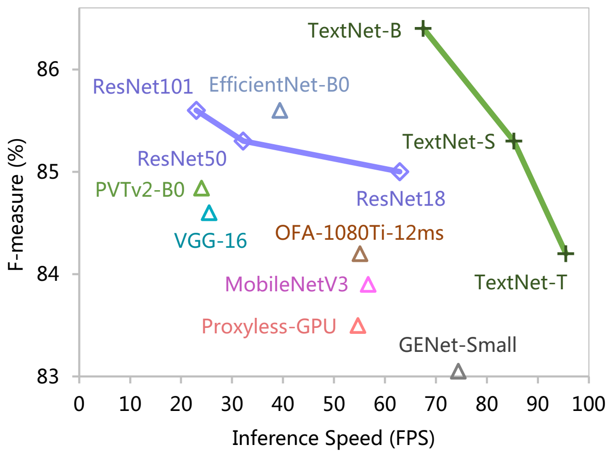


Fig. 6. Text detection F-measure and inference speed of different backbones on Total-Text [8] dataset, where we scale the shorter side of images to 640 pixels. Our TextNet models significantly outperform existing hand-crafted and auto-searched networks.

2) *Inference Settings:* In the inference phase, we scale the shorter side of images to a fixed size and report the performance on each dataset. For a fair comparison, we evaluate all testing images and calculate the average speed. Our main results are tested with a batch size of 1 on one 1080Ti GPU unless explicitly stated. When using TensorRT [17] to speed up inference, we adopt a V100 GPU instead of 1080Ti to deploy our models, because the 1080Ti GPU does not support half-precision (FP16) inference.

3) *NAS Settings:* We extend the widely-used Proxyless-NAS [11] for the architecture search of text detection. During searching, we consider a total of $L = 36$ searchable blocks. Following the common practice [9] of doubling the number of channels when halving the size of feature maps, we set C_1, C_2, C_3, C_4 to 64, 128, 256, and 512, respectively. We set our target inference speed T as 100, 80, and 60 FPS for searching TextNet-T, -S, and -B, respectively. To keep generalization ability, we take IC17-MLT [52] as the training set, and construct a validation set that concludes the training images of ICDAR 2015 [15] and Total-Text [8]. The entire network is trained and searched for 200 epochs, which takes around 200 GPU hours on 1080Ti.

C. Ablation Study and Analysis

1) *Searched Architecture:* As shown in Fig. 5, we plot the searched architectures of TextNet-T/S/B, from which we can make the following observations:

• Asymmetric convolutions are the dominant operators in our TextNet, which facilitate the detection of text lines with promising accuracy and high efficiency.

• TextNet-T and -S tend to stack more convolutions in the deep stages (stage-3 and -4), while TextNet-B prefers shallow stages (stage-1 and -2). It demonstrates that the stacking rules of TextNet are different under three specified speed constraints (*i.e.*, 100, 80, and 60 FPS), and shows that the common strategy derived from hand-crafted network design, *i.e.* stacking most layers in stage-3 [9], [35], [58], may be sub-optimal for real-time text detection.

2) *Effect of Proposed Components:* We show the effect of each proposed component in Table I. For a fair comparison, all models are pre-trained on IC17-MLT [52] and finetuned on Total-Text [8] or ICDAR 2015 [15]. Compared to the baseline that is equipped with ResNet18 [9] and CPU-based post-processing Pixel Aggregation [4], our FAST-R18 replaces the post-processing with the GPU-parallel text dilation, achieving better efficiency with comparable detection performance. Moreover, we substitute the ResNet18 backbone with our TextNet-B, which further boosts performance and efficiency, and reduce the number of parameters.

3) *Effect of Dilation Size:* In this experiment, we study the effect of the dilation size s (equal to the erosion size) based on our FAST-R18 model. We scale the shorter sides of images in Total-Text [8] and ICDAR 2015 [15] to 640 and 736 pixels, respectively. As reported in Table II, the F-measure on both datasets drops when the dilation size is too small. Empirically, we set the dilation size s to 9 by default. Note that if the size of the shorter side (denoted as S) is changed, the dilation size s should be updated proportionally to obtain the best performance:

$$s_{\text{new}} = \text{Round}(S_{\text{new}} \times s_{\text{default}} / S_{\text{default}}). \qquad (5)$$

Here, $\text{Round}(\cdot)$ is a function to round the decimal portion. For example, when the shorter side is fixed to 800 pixels instead

TABLE IV
DETECTION RESULTS ON TOTAL-TEXT [8]. THE SUFFIX OF OUR METHOD MEANS THE SIZE OF THE SHORTER SIDE. "EXT." DENOTES EXTERNAL DATA. "P", "R", AND "F" INDICATE PRECISION, RECALL, AND F-MEASURE, RESPECTIVELY.

Method	Backbone	Ext.	P	R	F	FPS
Non-real-time Methods						
TextSnake [2]	VGG16	✓	82.7	74.5	78.4	12.4
TextField [59]	VGG16	✓	81.2	79.9	80.6	-
PSENet [3]	ResNet50	✓	84.0	78.0	80.9	3.9
LOMO [60]	ResNet50	✓	88.6	75.7	81.6	4.4
SPCNet [29]	ResNet50	✓	83.0	82.8	82.9	4.6
FCENet [6]	ResNet50	-	87.4	79.8	83.4	-
CRAFT [61]	VGG16	✓	87.6	79.9	83.6	4.8
ContourNet [62]	ResNet50	-	86.9	83.9	85.4	3.8
TextFuseNet [5]	ResNet101	✓	89.0	85.3	87.1	3.3
ABPNet [63]	ResNet50	✓	90.7	85.2	87.9	10.7
Real-time Methods						
EAST [26]	PVANet	-	50.0	36.2	42.0	-
PAN [4]	ResNet18	✓	89.3	81.0	85.0	39.6
PAN++ [33]	ResNet18	✓	89.9	81.0	85.3	38.3
DB-R18 [1]	ResNet18	✓	88.3	77.9	82.8	50.0
DB-R50 [1]	ResNet50	✓	87.1	82.5	84.7	32.0
DB++-R18 [34]	ResNet18	✓	87.4	79.6	83.3	48.0
DB++-R50 [34]	ResNet50	✓	88.9	83.2	86.0	28.0
FAST-T-448 (ours)	TextNet-T	✓	86.5	77.2	81.6	152.8
FAST-S-512 (ours)	TextNet-S	✓	88.3	81.7	84.9	115.5
FAST-B-512 (ours)	TextNet-B	✓	89.6	82.4	85.8	93.2
FAST-B-640 (ours)	TextNet-B	✓	89.9	83.2	86.4	67.5
FAST-B-800 (ours)	TextNet-B	✓	90.0	85.2	87.5	46.0

TABLE V
DETECTION RESULTS ON CTW1500 [14]. THE SUFFIX OF OUR METHOD MEANS THE SIZE OF THE SHORTER SIDE. "EXT." DENOTES EXTERNAL DATA. "P", "R", AND "F" INDICATE PRECISION, RECALL, AND F-MEASURE, RESPECTIVELY.

Method	Backbone	Ext.	P	R	F	FPS
Non-real-time Methods						
TextSnake [2]	VGG16	✓	67.9	85.3	75.6	-
LOMO [60]	ResNet50	✓	89.2	69.6	78.4	4.4
SAE [31]	ResNet50	✓	82.7	77.8	80.1	-
TextField [59]	VGG16	✓	83.0	79.8	81.4	-
PSENet [3]	ResNet50	✓	84.8	79.7	82.2	3.9
FCENet [6]	ResNet50	-	85.7	80.7	83.1	-
CRAFT [61]	VGG16	✓	86.0	81.1	83.5	7.6
ContourNet [62]	ResNet50	-	83.7	84.1	83.9	4.5
ReLaText [64]	ResNet50	✓	86.2	83.3	84.8	10.6
ABPNet [63]	ResNet50	✓	86.5	83.6	85.0	12.2
TextFuseNet [5]	ResNet101	✓	87.8	85.4	86.6	3.7
Real-time Methods						
EAST [26]	PVANet	-	78.7	49.1	60.4	21.2
PAN [4]	ResNet18	✓	86.4	81.2	83.7	39.8
PAN++ [33]	ResNet18	✓	87.1	81.1	84.0	36.0
DB-R18 [1]	ResNet18	✓	84.8	77.5	81.0	55.0
DB-R50 [1]	ResNet50	✓	86.9	80.2	83.4	22.0
DB++-R18 [34]	ResNet18	✓	86.7	81.3	83.9	40.0
DB++-R50 [34]	ResNet50	✓	88.5	82.0	85.1	21.0
FAST-T-512 (ours)	TextNet-T	✓	85.5	77.9	81.5	129.1
FAST-S-512 (ours)	TextNet-S	✓	85.6	78.7	82.0	112.9
FAST-B-512 (ours)	TextNet-B	✓	85.7	80.2	82.9	92.6
FAST-B-640 (ours)	TextNet-B	✓	87.8	80.9	84.2	66.5

of 640 pixels for training, we set the dilation size s to 11 according to the Eqn. (5).

4) Comparison with Hand-Crafted Networks: We first compare our TextNet with representative hand-crafted backbones, such as ResNets [9] and VGG16 [38]. For a fair comparison, all models are first pre-trained on IC17-MLT [52] and then finetuned on Total-Text [8]. As shown in Fig. 6, the proposed TextNet models achieve a better trade-off between accuracy and inference speed than previous hand-crafted models by a significant margin. In addition, notably, our TextNet-T, -S, and -B only have 6.8M, 8.0M, and 8.9M parameters respectively, which are more parameter-efficient than ResNets and VGG16. These results demonstrate that TextNet models are effective for text detection on the GPU device.

5) Comparison with Auto-Searched Networks: Here we compare our TextNet with representative auto-searched backbones. For a fair comparison, all models are pre-trained on IC17-MLT [52] and finetuned on Total-Text [8]. As shown in Fig. 6 and Table III, our TextNet models outperform existing searched networks in terms of accuracy and inference speed, including Proxyless-GPU [11], OFANet-12ms [10], MobileNetV3 [12], GENet-Small [57], and EfficientNet-B0 [13]. Specifically, TextNet-S achieves 85.4% F-measure at 85.3 FPS, being 1.2% more accurate and 1.5× faster than OFANet-12ms. TextNet-B yields 86.4% F-measure at 67.5 FPS, which is 1.7× faster than EfficientNet-B0 while improving detection performance by 0.8% F-measure. A major reason is that these networks are mainly searched for image classification, and the generalization ability on other tasks is not robust. Therefore, designing the search space and reward function for text detection is meaningful and necessary.

D. Comparison with State-of-the-Art Methods

1) Curve Text Detection: To show the advantages of FAST in detecting curved text, we compare it with existing state-of-the-art methods on the Total-Text [8] and CTW1500 [65] datasets, and report the results in Table IV and Table V.

On Total-Text, FAST-T-448 yields an F-measure of 81.6% at 152.8 FPS, which is faster than all previous methods. Our FAST-S-512 outperforms the real-time text detector DB++-R18 [34] by 1.6% in F-measure (84.9% *vs.* 83.3%) and runs 2.4× faster. Compared to PAN++ [33], FAST-B-640 is 29.2 FPS faster, while the F-measure is 1.1% better (86.4% *vs.* 85.3%). It is notable that when taking a larger input resolution, FAST-B-800 achieves the best F-measure of 87.5%, surpassing all real-time counterparts in F-measure by at least 1.5% while still keeping a fast inference speed (46.0 FPS).

Similar results are also on CTW1500. For example, the inference speed of FAST-T-512 is 129.1 FPS, which is at least 2.3× faster than prior arts, while the F-measure is still very competitive (81.5%). The best F-measure of our method is 84.2%, which is slightly higher than the strong counterpart DB++-R18 [34] (84.2% *vs.* 83.9%), while our method runs at a faster speed (66.5 FPS *vs.* 40.0 FPS). We show some qualitative curved text detection results in Fig. 7(a)(b), which

TABLE VI
DETECTION RESULTS ON ICDAR 2015 [15]. THE SUFFIX OF OUR METHOD MEANS THE SIZE OF THE SHORTER SIDE. "EXT." DENOTES EXTERNAL DATA. "P", "R", AND "F" INDICATE PRECISION, RECALL, AND F-MEASURE, RESPECTIVELY.

Method	Backbone	Ext.	P	R	F	FPS
Non-real-time Methods						
Corner [50]	VGG16	✓	94.1	70.7	80.7	3.6
PixelLink [28]	VGG16	-	82.9	81.7	82.3	7.3
TextSnake [2]	VGG16	✓	84.9	80.4	82.6	1.1
FCENet [6]	ResNet50	-	85.1	84.2	84.6	-
PolarMask++ [54]	ResNet50	✓	87.3	83.5	85.4	10.0
PSENet [3]	ResNet50	✓	86.9	84.5	85.7	1.6
CRAFT [61]	VGG16	✓	89.8	84.3	86.9	-
LOMO [60]	ResNet50	✓	91.3	83.5	87.2	3.4
SPCNet [29]	ResNet50	✓	88.7	85.8	87.2	4.6
Real-time Methods						
EAST [26]	PVANet	-	83.6	73.5	78.2	13.2
PAN [4]	ResNet18	✓	84.0	81.9	82.9	26.1
PAN++ [33]	ResNet18	✓	85.9	80.4	83.1	28.2
DB-R18 [1]	ResNet18	✓	86.8	78.4	82.3	48.0
DB-R50 [1]	ResNet50	✓	91.8	83.2	87.3	12.0
DB++-R18 [34]	ResNet18	✓	90.1	77.2	83.1	44.0
DB++-R50 [34]	ResNet50	✓	90.9	83.9	87.3	10.0
FAST-T-736 (ours)	TextNet-T	✓	86.0	77.9	81.7	60.9
FAST-S-736 (ours)	TextNet-S	✓	86.3	79.8	82.9	53.9
FAST-B-736 (ours)	TextNet-B	✓	88.0	81.7	84.7	42.7
FAST-B-896 (ours)	TextNet-B	✓	89.2	83.6	86.3	31.8
FAST-B-1280 (ours)	TextNet-B	✓	89.7	84.6	87.1	15.7

TABLE VII
DETECTION RESULTS ON MSRA-TD500 [16]. THE SUFFIX OF OUR METHOD MEANS THE SIZE OF THE SHORTER SIDE. "EXT." DENOTES EXTERNAL DATA. "P", "R", AND "F" INDICATE PRECISION, RECALL, AND F-MEASURE, RESPECTIVELY.

Method	Backbone	Ext.	P	R	F	FPS
Non-real-time Methods						
PixelLink [28]	VGG16	-	83.0	73.2	77.8	3.0
TextSnake [2]	VGG16	✓	83.2	73.9	78.3	1.1
TextField [59]	VGG16	✓	87.4	75.9	81.3	5.2
Corner [50]	VGG16	✓	87.6	76.2	81.5	5.7
CRAFT [61]	VGG16	✓	88.2	78.2	82.9	8.6
SAE [31]	ResNet50	✓	84.2	81.7	82.9	3.0
SBD [66]	ResNet50	✓	89.6	80.5	84.8	3.2
DRRG [67]	VGG16	✓	88.0	82.3	85.1	-
ABPNet [63]	ResNet50	✓	86.6	84.5	85.6	12.3
ReLaText [64]	ResNet50	✓	90.5	83.2	86.7	8.3
Real-time Methods						
EAST [26]	PVANet	-	87.3	67.4	76.1	13.2
PAN [4]	ResNet18	✓	84.4	83.8	84.1	30.2
PAN++ [33]	ResNet18	✓	85.3	84.0	84.7	32.5
DB-R18 [1]	ResNet18	✓	90.4	76.3	82.8	62.0
DB-R50 [1]	ResNet50	✓	91.5	79.2	84.9	32.0
DB++-R18 [34]	ResNet18	✓	87.9	82.5	85.1	55.0
DB++-R50 [34]	ResNet50	✓	91.5	83.3	87.2	29.0
FAST-T-512 (ours)	TextNet-T	✓	91.1	78.8	84.5	137.2
FAST-T-736 (ours)	TextNet-T	✓	88.1	81.9	84.9	79.6
FAST-S-736 (ours)	TextNet-S	✓	91.6	81.7	86.4	72.0
FAST-B-736 (ours)	TextNet-B	✓	92.1	83.0	87.3	56.8

demonstrates that the proposed FAST can accurately locate text lines with complex shapes.

2) Oriented Text Detection: We evaluate the effectiveness of FAST in detecting oriented text lines on the ICDAR 2015 [15] dataset. From Table VI, we can observe that our fastest model FAST-T-736 reaches 60.9 FPS and maintains a competitive F-measure of 81.7%. Compared with PAN++ [33], FAST-B-896 surpasses it by 3.2% in F-measure (86.3% *vs.* 83.1%) and is more efficient (31.8 FPS *vs.* 28.2 FPS). Because ICDAR 2015 contains many small text lines, previous methods [3], [29] always adopt high-resolution images to ensure detection performance. With this setting, FAST-B-1280 achieves an F-measure of 87.1%, which is comparable with DB-R50 [1] and DB++-R50 [34] (87.1% *vs.* 87.3%). Besides, compared with PSENet [3], this model outperforms it by 1.4% F-measure (87.1% *vs.* 85.7%) and runs 9.8× faster. Some qualitative results of oriented text detection are shown in Fig. 7(c).

3) Long Straight Text Detection: FAST is also robust for long straight text detection. As reported in Table VII, on the MSRA-TD500 [16] dataset, FAST-T-736 runs at 137.2 FPS with 84.5% F-measure, which is more efficient than all previous real-time detectors. For example, it is 4.5× and 2.2× faster than PAN [4] and DB-R18 [1] respectively, while keeping higher detection F-measure. Besides, FAST-S-736 achieves 86.4% F-measure at 72.0 FPS, outperforming DB++-R18 [34] by 1.3% (86.4% *vs.* 85.1%) and runs 17 FPS faster (72.0 FPS *vs.* 55.0 FPS). FAST-B-736 yields the F-measure of 87.3%, which is slightly better than DB++-R50 [34] but with significantly higher efficiency (56.8 FPS *vs.* 29.0 FPS). We present some qualitative straight text detection results in Fig. 7(d).

E. Efficiency Analysis

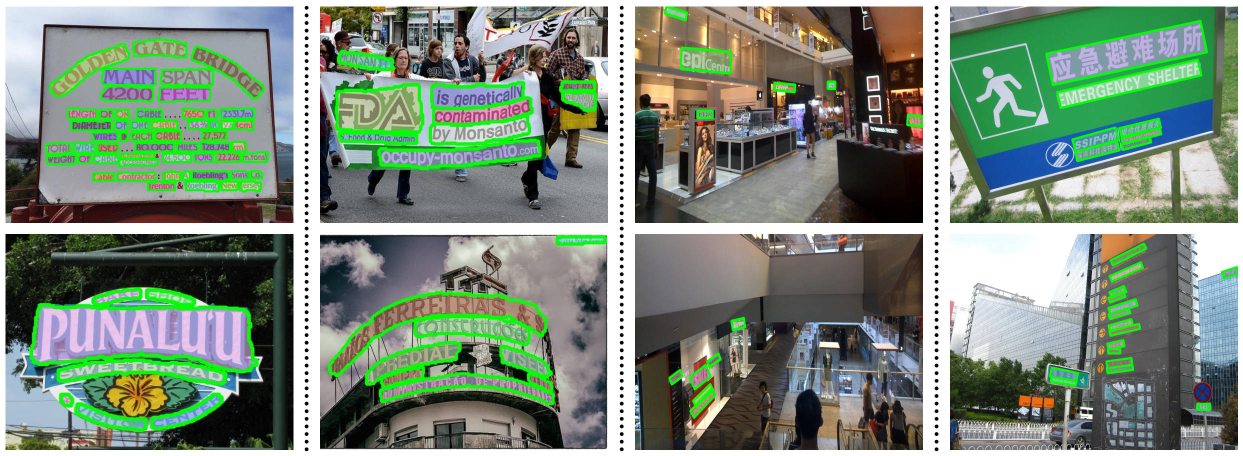
1) Parameter: In Table VIII, we compare the number of parameters of the proposed FAST with representative arbitrarily-shaped text detectors. As shown, our detectors FAST-T, FAST-S, and FAST-B have 8.5M, 9.7M, and 10.6M parameters respectively, which are more parameter-efficient than previous real-time text detectors, such as PAN [4] (12.2M) and DB-R18 [1] (13.8M).

2) FLOPs: We report the total FLOPs of different text detectors in Table VIII. Although our primary concern is inference speed rather than FLOPs, our FAST models also have advantages over previous methods. For example, on the Total-Text dataset [8], with similar FLOPs as DB-R18 [1] (29.5G *vs.* 30.0G), our FAST-T-512 improves the F-measure from 82.8% to 83.5%. Besides, FAST-B-640 achieves an F-measure of 86.4% with 64.0G FLOPs, being 1.4 points better (86.4% *vs.* 85.0%) than PAN-640 [4] that has similar FLOPs (64.0G *vs.* 65.4G). Compared with PSENet [3], our FAST-B-800 achieves the more competitive performance of 87.5% F-measure, but its computational cost is less than a third of the PSENet (100.1G *vs.* 345.9G).

3) Inference Speed: In this experiment, we adopt TensorRT [17], an inference engine specially designed for industrial deployment, to further speed up our FAST models. Because the 1080Ti GPU does not support half-precision (FP16) inference, we consider two additional settings: (1) V100 + PyTorch

TABLE VIII
EFFICIENCY ANALYSIS WITH STATE-OF-THE-ART METHODS ON TOTAL-TEXT [8]. "F" INDICATES F-MEASURE. "SCALE" DENOTES THE SCALE OF THE TESTING IMAGE, WHERE "L:" MEANS THE LONGER SIDE IS FIXED, "S:" MEANS THE SHORTER SIDE IS FIXED, AND "H:" MEANS THE HEIGHT IS FIXED. "BS" IS SHORT FOR BATCH SIZE. "#PARAM" REPRESENTS THE TOTAL NUMBER OF PARAMETERS FOR THE DETECTORS. THE PYTORCH FPS IS MEASURED WITH BATCH SIZE 1.

Method	F-measure	#Param (total)	Scale	FLOPs	PyTorch FPS (FP32)		TensorRT FPS (V100 FP16)			
					1080Ti	V100	BS = 1	BS = 2	BS = 4	BS = 8
PSENet-4s [3]	79.6	28.7M	L: 1280	345.9G	8.4	10.8	59.3	69.2	72.8	77.3
PAN-320 [4]	79.9	12.2M	S: 320	**16.4G**	82.4	110.8	237.5	323.2	363.4	385.2
PAN-512 [4]	84.3		S: 512	41.9G	57.1	85.5	187.6	204.3	215.5	243.8
PAN-640 [4]	85.0		S: 640	65.4G	39.6	64.6	130.4	154.4	162.8	180.7
DB-R18 [1]	82.8	13.8M	H: 640	30.0G	50.0	73.0	105.4	143.7	204.5	227.6
FAST-T-448 (ours)	81.6	**8.5M**	S: 448	22.6G	**152.8**	**187.0**	**385.9**	**524.8**	**628.1**	**634.7**
FAST-T-512 (ours)	83.5		S: 512	29.5G	131.1	176.4	343.4	408.2	498.5	525.6
FAST-T-640 (ours)	84.2		S: 640	46.0G	95.5	136.4	275.6	287.1	345.4	364.0
FAST-S-512 (ours)	84.9	9.7M	S: 512	33.1G	115.5	150.5	289.7	400.2	465.7	481.5
FAST-S-640 (ours)	85.4		S: 640	51.7G	85.3	115.9	249.6	301.7	325.0	332.0
FAST-B-512 (ours)	85.8	10.6M	S: 512	41.0G	93.2	128.6	281.0	358.6	414.3	448.0
FAST-B-640 (ours)	86.4		S: 640	64.0G	67.5	99.8	219.7	261.8	293.3	297.0
FAST-B-800 (ours)	**87.5**		S: 800	100.1G	46.0	69.2	143.0	177.5	189.2	195.3



(a) Total-Text (b) CTW1500 (c) ICDAR 2015 (d) MSRA-TD500

Fig. 7. Qualitative text detection results of FAST on Total-Text [8], CTW1500 [65], ICDAR 2015 [15] and MSRA-TD500 [16]. These results show that our FAST models are suitable for various complicated natural scenes, including arbitrary shapes, multiple languages, and extreme aspect ratios.

(FP32) with batch size 1, and (2) V100 + TensorRT (FP16) with batch size from 1 to 8.

As shown in Table VIII, when using PyTorch and V100 GPU, our fastest model FAST-T-448 reaches a speed of 187.0 FPS. With TensorRT optimization, it can be further accelerated to 634.7 FPS (batch size = 8), being 3.4× faster than its PyTorch implementation (batch size = 1), which demonstrates the efficiency of FAST in practical applications.

F. Qualitative Results

In this section, we show some qualitative text detection results of the proposed FAST on four challenging datasets, including Total-Text (see Fig. 7(a)), CTW1500 (see Fig. 7(b)), ICDAR 2015 (see Fig. 7(c)) and MSRA-TD500 (see Fig. 7(d)). These results demonstrate that our FAST models are suitable for various complicated natural scenes, including arbitrary shapes, multiple languages, extreme aspect ratios, etc.

V. CONCLUSION

In this work, we proposed FAST, a faster arbitrarily-shaped text detector. To achieve high efficiency, we presented a minimalist kernel representation (MKR), as well as a GPU-parallel post-processing—text dilation, making our models can completely run on the GPU. Moreover, we designed a search space and reward function tailored for text detection, and searched for a series of efficient backbone networks (*i.e.*, TextNet) friendly to text detection. Extensive experiments on several challenging datasets show that equipped with these two designs, our FAST achieves a significantly better trade-off between detection performance and inference speed than previous works. We hope our method could serve as a corner-stone for text-related real-time applications.

REFERENCES

[1] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, "Real-time scene text detection with differentiable binarization," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 34, 2020, pp. 11 474–11 481. 1, 2, 4, 7, 8, 9

[2] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao, "Textsnake: A flexible representation for detecting text of arbitrary shapes," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 20–36. 1, 2, 5, 7, 8

[3] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, and S. Shao, "Shape robust text detection with progressive scale expansion network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9336–9345. 1, 2, 4, 5, 7, 8, 9

[4] W. Wang, E. Xie, X. Song, Y. Zang, W. Wang, T. Lu, G. Yu, and C. Shen, "Efficient and accurate arbitrary-shaped text detection with pixel aggregation network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 8440–8449. 1, 2, 3, 4, 5, 6, 7, 8, 9

[5] J. Ye, Z. Chen, J. Liu, and B. Du, "Textfusenet: Scene text detection with richer fused features." in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2020, pp. 516–522. 1, 7

[6] Y. Zhu, J. Chen, L. Liang, Z. Kuang, L. Jin, and W. Zhang, "Fourier contour embedding for arbitrary-shaped text detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3123–3131. 1, 7, 8

[7] Y. Liu, H. Chen, C. Shen, T. He, L. Jin, and L. Wang, "Abcnet: Real-time scene text spotting with adaptive bezier-curve network," in *proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9809–9818. 1

[8] C. K. Ch'ng and C. S. Chan, "Total-text: A comprehensive dataset for scene text detection and recognition," in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 2017, pp. 935–942. 1, 2, 5, 6, 7, 8, 9

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. 1, 2, 5, 6, 7

[10] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 1, 3, 6, 7

[11] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *Proceedings of the International Conference on Learning Representations (ICML)*, 2018. 1, 3, 4, 5, 6, 7

[12] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1314–1324. 1, 2, 3, 5, 6, 7

[13] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2019, pp. 6105–6114. 1, 3, 5, 6, 7

[14] Y. Liu, L. Jin, S. Zhang, C. Luo, and S. Zhang, "Curved scene text detection via transverse and longitudinal sequence connection," *Pattern Recognition*, vol. 90, pp. 337–345, 2019. 2, 5, 7

[15] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu *et al.*, "Icdar 2015 competition on robust reading," in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 2015, pp. 1156–1160. 2, 5, 6, 8, 9

[16] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu, "Detecting texts of arbitrary orientations in natural images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 1083–1090. 2, 5, 8, 9

[17] H. Vanholder, "Efficient inference with tensorrt," in *GPU Technology Conference*, vol. 1, 2016, p. 2. 2, 6, 8

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016, pp. 21–37. 2

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016. 2

[20] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2961–2969. 2

[21] M. Liao, B. Shi, and X. Bai, "Textboxes++: A single-shot oriented scene text detector," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3676–3690, 2018. 2

[22] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "Textboxes: A fast text detector with a single deep neural network," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017. 2

[23] M. Liao, Z. Zhu, B. Shi, G.-s. Xia, and X. Bai, "Rotation-sensitive regression for oriented scene text detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5909–5918. 2

[24] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue, "Arbitrary-oriented scene text detection via rotation proposals," *IEEE Transactions on Multimedia*, vol. 20, pp. 3111–3122, 2018. 2

[25] B. Shi, X. Bai, and S. Belongie, "Detecting oriented text in natural images by linking segments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2550–2558. 2

[26] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "East: an efficient and accurate scene text detector," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5551–5560. 2, 5, 7, 8

[27] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, "Detecting text in natural image with connectionist text proposal network," in *European conference on computer vision*. Springer, 2016, pp. 56–72. 2

[28] D. Deng, H. Liu, X. Li, and D. Cai, "Pixellink: Detecting scene text via instance segmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 32, 2018. 2, 8

[29] E. Xie, Y. Zang, S. Shao, G. Yu, C. Yao, and G. Li, "Scene text detection with supervised pyramid context network," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 33, 2019, pp. 9038–9045. 2, 5, 7, 8

[30] P. Lyu, M. Liao, C. Yao, W. Wu, and X. Bai, "Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 67–83. 2

[31] Z. Tian, M. Shu, P. Lyu, R. Li, C. Zhou, X. Shen, and J. Jia, "Learning shape-aware embedding for scene text detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4234–4243. 2, 7, 8

[32] B. R. Vatti, "A generic solution to polygon clipping," *Communications of the ACM*, vol. 35, no. 7, pp. 56–63, 1992. 2

[33] W. Wang, E. Xie, X. Li, X. Liu, D. Liang, Y. Zhibo, T. Lu, and C. Shen, "Pan++: Towards efficient and accurate end-to-end spotting of arbitrarily-shaped text," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2, 7, 8

[34] M. Liao, Z. Zou, Z. Wan, C. Yao, and X. Bai, "Real-time scene text detection with differentiable binarization and adaptive scale fusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2, 7, 8

[35] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pvtv2: Improved baselines with pyramid vision transformer," *arXiv preprint arXiv:2106.13797*, 2021. 2, 6

[36] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "Repvgg: Making vgg-style convnets great again," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 13 733–13 742. 2, 4, 5

[37] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao, "Vision transformer adapter for dense predictions," *arXiv preprint arXiv:2205.08534*, 2022. 2

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 2, 6, 7

[39] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 82–92. 3

[40] L. Xu, Y. Guan, S. Jin, W. Liu, C. Qian, P. Luo, W. Ouyang, and X. Wang, "Vipnas: Efficient video pose estimation via neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 072–16 081. 3

[41] S. Hong, D. Kim, and M.-K. Choi, "Memory-efficient models for scene text recognition via neural architecture search," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (CACV)*, 2020, pp. 183–191. 3

[42] H. Zhang, Q. Yao, M. Yang, Y. Xu, and X. Bai, "Autostr: Efficient backbone search for scene text recognition," in *Proceedings of the*

European Conference on Computer Vision (ECCV), 2020, pp. 751–767. 3

[43] S. Allegretti, F. Bolelli, and C. Grana, "Optimized block-based algorithms to label connected components on gpus," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 2, pp. 423–438, 2019. 4

[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826. 5

[45] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520. 5

[46] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2820–2828. 5

[47] N. Wang, Y. Gao, H. Chen, P. Wang, Z. Tian, C. Shen, and Y. Zhang, "Nas-fcos: Fast neural architecture search for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 943–11 951. 5

[48] F. Milletari, N. Navab, S. Ahmadi, and V-net, "Fully convolutional neural networks for volumetric medical image segmentation," in *Proceedings of the Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 565–571. 5

[49] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 761–769. 5

[50] P. Lyu, C. Yao, W. Wu, S. Yan, and X. Bai, "Multi-oriented scene text detection via corner localization and region segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7553–7563. 5, 8

[51] C. Yao, X. Bai, and W. Liu, "A unified framework for multioriented text detection and recognition," *IEEE Transactions on Image Processing*, vol. 23, no. 11, pp. 4737–4749, 2014. 5

[52] N. Nayef, F. Yin, I. Bizid, H. Choi, Y. Feng, D. Karatzas, Z. Luo, U. Pal, C. Rigaud, J. Chazalon *et al.*, "Icdar2017 robust reading challenge on multi-lingual scene text detection and script identification-rrc-mlt," in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1, 2017, pp. 1454–1459. 5, 6, 7

[53] W. Feng, W. He, F. Yin, X.-Y. Zhang, and C.-L. Liu, "Textdragon: An end-to-end framework for arbitrary shaped text spotting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9076–9085. 5

[54] E. Xie, W. Wang, M. Ding, R. Zhang, and P. Luo, "Polarmask++: Enhanced polar representation for single-shot instance segmentation and beyond," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 5, 8

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 5

[56] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," *arXiv preprint arXiv:2102.12122*, 2021. 6

[57] M. Lin, H. Chen, X. Sun, Q. Qian, H. Li, and R. Jin, "Neural architecture design for gpu-efficient networks," *arXiv preprint arXiv:2006.14090*, 2020. 6, 7

[58] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li *et al.*, "Internimage: Exploring large-scale vision foundation models with deformable convolutions," *arXiv preprint arXiv:2211.05778*, 2022. 6

[59] Y. Xu, Y. Wang, W. Zhou, Y. Wang, Z. Yang, and X. Bai, "Textfield: Learning a deep direction field for irregular scene text detection," *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5566–5579, 2019. 7, 8

[60] C. Zhang, B. Liang, Z. Huang, M. En, J. Han, E. Ding, and X. Ding, "Look more than once: An accurate detector for text of arbitrary shapes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 552–10 561. 7, 8

[61] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9365–9374. 7, 8

[62] Y. Wang, H. Xie, Z.-J. Zha, M. Xing, Z. Fu, and Y. Zhang, "Contournet: Taking a further step toward accurate arbitrary-shaped scene text detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 753–11 762. 7

[63] S.-X. Zhang, X. Zhu, C. Yang, H. Wang, and X.-C. Yin, "Adaptive boundary proposal network for arbitrary shape text detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 1305–1314. 7, 8

[64] C. Ma, L. Sun, Z. Zhong, and Q. Huo, "Relatext: Exploiting visual relationships for arbitrary-shaped scene text detection with graph convolutional networks," *Pattern Recognition*, vol. 111, p. 107684, 2021. 7, 8

[65] Y. Liu, L. Jin, S. Zhang, and S. Zhang, "Detecting curve text in the wild: New dataset and new solution," *arXiv preprint arXiv:1712.02170*, 2017. 7, 9

[66] Y. Liu, S. Zhang, L. Jin, L. Xie, Y. Wu, and Z. Wang, "Omnidirectional scene text detection with sequential-free box discretization," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 3052–3058. 8

[67] S.-X. Zhang, X. Zhu, J.-B. Hou, C. Liu, C. Yang, H. Wang, and X.-C. Yin, "Deep relational reasoning graph network for arbitrary shape text detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9699–9708. 8