StrokeNet: Stroke Assisted and Hierarchical Graph Reasoning Networks

Lei Li, Kai Fan, and Chun Yuan, Senior Member, IEEE

Abstract—Scene text detection is still a challenging task, as there may be extremely small or low-resolution strokes and close or arbitrary-shaped texts. In this paper, StrokeNet proposes to effectively detect the texts by capturing the fine-grained strokes and inferring structural relations between the hierarchical representations of each text area in the graph-based network. Different from existing approaches that represent the text area by a series of points or rectangular boxes, we directly localize the strokes of each text instance. We introduce Stroke Assisted Prediction Network (SAPN), which performs hierarchical representation learning of text areas, effectively capturing extremely small or low-resolution texts. We extract a series of text- and stroke-level rectangular boxes on the predicted text areas, which are treated as graph nodes and grouped to form the corresponding local graphs. Hierarchical Relation Graph Network (HRGN) then performs relational reasoning and predicts the likelihood of linkages among graph nodes of different levels. It efficiently splits the close text instances and grouping node classification results into the arbitrary-shaped text area. We introduce a novel dataset with stroke-level annotations, namely SynthStroke, for offline pre-training of widespread text detectors. Experiments on benchmarks verify the State-of-the-Art performance of our method.

Index Terms—Scene text detection, hierarchical representations, graph networks.

I. INTRODUCTION

SCENE text detection in the wild, as a fundamental task in the computer vision field, has been widely applied in numerous applications [1], [2], [3], [4], [5], [6], such as autonomous driving, document analysis and image understanding. The goal of text detection is to label each text instance with a bounding box from input images. Current leading approaches are mainly extended from the object detection [7], [8] or segmentation frameworks [9], which could be summarized into regression-based methods [6], [10], [11], [12], segmentation-based methods [4], [5], [11], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22] and hybrid methods [23], [24], [25], [26], [27], respectively. However, they may suffer in more complicated cases, and the performances of State-of-the-Art methods are still far from the demands of real-world applications.

First, existing text detectors are commonly not adept in capturing the fine-grained strokes, which are the character parts in each text-level bounding box and play an essential role in the representation of the text area. If models could not capture some extremely small or low-resolution strokes well, they may not accurately detect the corresponding text instances for challenging detection scenarios. Second, detecting only from a text-level perspective is challenging to separate the text instances close to each other [16], [28]. In contrast, exploring the interval between strokes of the text area is an effective way to distinguish the close text-level instances from each other. For example, Strokelets [29] adopted a rule-based procedure to generate strokes with multi-scale representation to show its effectiveness for text recognition. Third, regression-based methods [23], [30], [31] typically adapt popular object detectors [32], [33], [34] to localize text-level bounding boxes with location regression, which struggle to localize the text instance with arbitrary shapes. While segmentation-based methods [15], [28] commonly extract text-level bounding boxes directly from the pixel-level segmentation, relying on heavy post-processing to compose the predicted regions into final text instances.

Also, there is a small number of hybrid methods [23], [24], [25] combine the thoughts of two mainstream ideas. However, these methods do not overcome the shortcomings of both regression-based and segmentation-based methods. In recent years, there has been an increasing interest in developing Graph Neural Networks (GNNs) [35], [36] for structured graph data, and a novel solution proposes to use graph networks to boost the performance of hybrid methods for text detection [25], [26]. For instance, DRRG [25] proposed an innovative local graph to bridge a segmentation-based text proposal model and a deep relational reasoning graph network. However, these methods still have room for performance promotion because they have not adequately exploited the abundant information in the text area. For example, they have not explored the fine-grained (i.e., stroke-level) representations, so the learned models tend to confuse adjacent text regions and produce incorrect detection results. Besides, they often directly transfer the typical GNNs [35] to the text detection domain with no specific design and perform a single level of relational reasoning in the text region. They have not fully exploited the reasoning ability of graph models.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
and have not profoundly combined the advantages of two mainstream ideas.

In addition, some real-world applications, such as image text editing and OCR translation \cite{37, 38}, require eliminating the original texts. Therefore, the fine-grained stroke-level representation can accurately define the region that needs the inpainting operation in the task.

To address the mentioned limitations, we first learn hierarchical representations of the text area by detecting both text instances and the corresponding stroke-level representations (strokes), which are experts in detecting extremely small or low-resolution texts. We extract a series of text- and stroke-level rectangular boxes on the predicted text areas. We obtain separated components treated as individual graph nodes and link them to build multiple local graphs. Relational reasoning, hierarchical aggregation, and linkages prediction are then performed on the built graphs, which is beneficial for splitting the close texts and making the arbitrary-shaped text region located precisely. Fig. 1 shows the whole process of our method.

To promote the research of stroke-level representations of the text area, SynthStroke is further introduced which is a multi-regression layer and learnable to promote the research in text quadrangles.

Fig. 1. The process of detecting texts in our StrokeNet.

Regression- or Segmentation-based Methods: Regression-based methods usually relied on general objects, such as Faster R-CNN \cite{32}, SSD \cite{33}, and YOLO \cite{34}, to localize text boxes by directing the offsets from anchors or pixels. Unlike general objects, text instances are often presented in irregular shapes with various scales and aspect ratios. For example, TextBoxes \cite{10} modified the anchor boxes and shape of convolution kernels to capture text with multiple aspect ratios effectively. SSTD \cite{11} proposed an attention mechanism to enhance the text region of the feature map. LOMO \cite{12} tried to refine bounding box proposals iteratively. Wang et al. \cite{6} proposed R-Net to address the large-variance scale problem by mapping the multi-scale features to a scale-invariant space. It introduced a novel Spatial Relationship Module (SPM) to enhance spatial semantics by simultaneously considering long-range dependencies and local independencies. However, regression-based methods often require complex anchor settings and exhaustive tuning, which prevent them from applying in the wild. Besides, these works are limited to representing accurate bounding boxes for arbitrary-shaped texts such as curved or deformed ones, which are widely distributed in real-world scenarios.

Segmentation-based methods formulate text detection as a segmentation problem, inspired by fully convolutional networks (FCN) \cite{39}. They usually combine the pixel-level prediction and the post-processing steps to extract text instances from the segmented text area. For example, SSTD \cite{11} proposed an attention mechanism encoded into convolutional features to reduce background interference. Ma et al. \cite{13} incorporated the Rotation Region-of-Interest (RRoI) pooling layer and learning of the rotated proposal into the region-proposal-based architecture \cite{32} to predict the orientation of a text line. Similarly, Mask TextSpotter \cite{14} detected arbitrary-shaped text instances based on an instance segmentation network named Mask R-CNN \cite{9}. TextSnake \cite{15} adopted a stride algorithm to extract the center line and reconstructed the text instances with the estimated geometry attributes. PSENet \cite{16} proposed progressive scale expansion by different scale kernels to position boundaries among texts close to each other. Dai

2) Stroke Assisted Prediction Network (SAPN) constructs hierarchical representations based on regression and segmentation predictions of the text area, making it possible for the end-to-end localization of both text instances and the corresponding stroke-level representations (strokes).

3) Hierarchical Relation Graph Network performs relational reasoning and hierarchical aggregation on the obtained local graphs. It guides the post-processing from linkages prediction to detection results, enabling the strengths of two mainstream ideas (regression/segmentation) to combine deeply to complement each other.

4) SynthStroke is introduced to promote the research in text detection, which includes 800 K synthetic images with text- and stroke-level annotations.

The remainder of the paper is organized as follows: Section II briefly reviews scene text detection. Section III describes the proposed method. Section IV presents a series of experimental results and analyses. Section V presents a concise conclusion summarizing the main points of the work.

II. RELATED WORK

Regression- or Segmentation-based Methods: Regression-based methods usually relied on general objects, such as Faster R-CNN \cite{32}, SSD \cite{33}, and YOLO \cite{34}, to localize text boxes by directing the offsets from anchors or pixels. Unlike general objects, text instances are often presented in irregular shapes with various scales and aspect ratios. For example, TextBoxes \cite{10} modified the anchor boxes and shape of convolution kernels to capture text with multiple aspect ratios effectively. SSTD \cite{11} proposed an attention mechanism to enhance the text region of the feature map. LOMO \cite{12} tried to refine bounding box proposals iteratively. Wang et al. \cite{6} proposed R-Net to address the large-variance scale problem by mapping the multi-scale features to a scale-invariant space. It introduced a novel Spatial Relationship Module (SPM) to enhance spatial semantics by simultaneously considering long-range dependencies and local independencies. However, regression-based methods often require complex anchor settings and exhaustive tuning, which prevent them from applying in the wild. Besides, these works are limited to representing accurate bounding boxes for arbitrary-shaped texts such as curved or deformed ones, which are widely distributed in real-world scenarios.

Segmentation-based methods formulate text detection as a segmentation problem, inspired by fully convolutional networks (FCN) \cite{39}. They usually combine the pixel-level prediction and the post-processing steps to extract text instances from the segmented text area. For example, SSTD \cite{11} proposed an attention mechanism encoded into convolutional features to reduce background interference. Ma et al. \cite{13} incorporated the Rotation Region-of-Interest (RRoI) pooling layer and learning of the rotated proposal into the region-proposal-based architecture \cite{32} to predict the orientation of a text line. Similarly, Mask TextSpotter \cite{14} detected arbitrary-shaped text instances based on an instance segmentation network named Mask R-CNN \cite{9}. TextSnake \cite{15} adopted a stride algorithm to extract the center line and reconstructed the text instances with the estimated geometry attributes. PSENet \cite{16} proposed progressive scale expansion by different scale kernels to position boundaries among texts close to each other.
et al. [4] introduced a text-related feature enhancement module, pyramid region-of-interest (ROI) pooling attention model, and a box-aware context-based text segmentation module for curved text detection. Xue et al. [17] employed a combination of spatial and frequency domain-based features for detecting arbitrarily-oriented text in low-light images. Zhang et al. [5] proposed a two-stage instance segmentation approach with omnidirectional pyramid mask proposal (OPMP) to deal with the stack-omnidirectional text dilemma, under-segmentation of very close text words, and over-segmentation of arbitrary-shape long text lines. Compared to regression-based methods, they localize arbitrary-shaped texts more accurately but struggle with splitting the close texts. Besides, time-consuming post-processing [28], [40] is often involved in grouping pixels into text instances, and this stage is easy to be affected by outliers.

Strokes are the character parts in each text region, and their labels are text segmentation maps. It is worth noting that previous methods such as SegLink [41] and CRAFT [42] segmented or regressed each rectangular box to obtain a single character while still containing background interference. More relevantly, Strokelets [29] adopted a rule-based procedure to generate strokes with multi-scale representation to show its effectiveness for text recognition. In contrast, our StrokeNet could completely segment the characters (strokes) from the complex background by accurately predicting the corresponding segmentation maps, which is more conducive to the subsequent processing of text detection.

**Hybrid Methods:** Hybrid methods combine the idea of two mainstream ideas. They typically perform the pixel-level segmentation to seek text regions and then apply bounding box regression to make the final prediction. For example, EAST [23] predicted offsets from pixels in each text region to perform multi-oriented regression. Zhong et al. [24] developed the anchor-free text detector AF-RPN, which applied segmentation and regression branches to multiple stages of CNN to acquire omnidirectional proposals for further refinement. Hybrid methods can inherit the advantages of both sides to improve detection accuracy. However, these methods do not overcome the shortcomings of both regression-based and segmentation-based methods.

In recent years, Graph Neural Networks (GNNs) [35] based scene text detection approaches [21], [25], [26] have made significant progress and substantially outperformed traditional text detectors in terms of both accuracy and capability. In this branch, DRRG [25] introduced an innovative local graph to bridge a segmentation-based text proposal model and a deep relational reasoning graph network. Ma et al. [26] proposed an arbitrary-shaped text detection approach ReLaText by formulating text detection as a visual relationship detection problem and using a GCN-based visual relationship detection framework to solve the challenging text-line grouping problem. However, these methods have not adequately exploited the abundant information in the fine-grained (i.e., stroke-level) representation, tending to confuse adjacent text regions for incorrect detection. Our proposal also falls into this category: by encoding the hierarchical representations of the text area in a segmentation-based manner and reasoning the relations according to regression-based box proposals, the strengths of two mainstream ideas are combined deeply to complement each other.

**III. PROPOSED METHOD**

The pipeline of the proposed StrokeNet is illustrated in Fig. 2, including two significant modules, SAPN and HRGN. In the first module, we start to apply ResNet-50 equipped with FPN [43] as the backbone to predict the classification and regression confidence of potential text area (text-level prediction block, denoted as TLP). Then, we introduce a stroke-level prediction (SLP) block to detect strokes precisely within the predicted text area. Afterward, box proposals of both text- and stroke-levels are extracted and treated as graph nodes to establish the corresponding local graphs. The isomorphic stroke graph is first built in the second module to update the attention-guided representation among stroke-level nodes. Then, the heterogeneous text graph is further built for relational reasoning and hierarchical aggregation from both levels. Finally, the likelihood of linkages among text-level nodes is inferred and the nodes are grouped into holistic text instances.

A. Stroke Assisted Prediction Network

The backbone adopted in this module is conducive to preserving spatial resolution [43] and taking full advantage of high-level semantic information. After extracting the 32-channel backbone features, two consecutive convolution layers with 16 and 8 output channels are applied to predict the attributes of the text area. Concretely, 4 of the output 8 channels define the classification logits of text area (TA) and text center area (TCA), and the rest 4 channels define the regression logits of $h_1$, $h_2$, $\cos \theta$, and $\sin \theta$. As shown in Fig. 3, TA represents the area where the text is located, where TCA is defined by shrinking TA along the direction perpendicular to the text writing [25]. Besides, $h_1$ and $h_2$ define the distance from current pixel to the upper edge and lower edge of TA respectively. Furthermore, $\theta$ represents the orientation of the text, and naturally indicates an optimization constraint $\cos \theta^2 + \sin \theta^2 = 1$ [15].

We will introduce a stroke-level prediction block: (1) the Text Features Distillation sub-block employs channel-wise attention to distill text representation from the backbone features, and (2) the Stroke Cues Filtration sub-block uses multi-scale orthogonal convolutions as well as spatial attention on rough stroke cues to suppress redundant background details.

**Text Features Distillation:** Since the representation of stroke has different dependencies on the obtained multi-channel features from the backbone, we introduce this sub-block to model the dependencies across channels and distill abstract semantic information of TA by adaptively adjusting the feature response values of each channel. Significantly, we crop out the minimum circumscribed rectangle of the TA (denoted by OTA) to obtain the corresponding representation (indicated as block-1 in Fig. 2) by locating the upper, lower, left, and right boundaries of pixels predicted to be text area. Then we utilize the max-pooling operation combined with consecutive convolution layers to generalize the features of OTA obtained from the backbone, which will be sent to two branches for post-processing. Specifically,
one branch up-samples the generalized features using deconvolution operations (denoted as t_conv) with factor 4. In contrast, the other branch adopts adaptive pooling connected by a shared MLP and a sigmoid layer to compute the channel-wise attention map. Finally, the two branches are multiplied to achieve semantic distillation, obtaining rough stroke cues (e.g., color, texture, and edge representation of strokes), which are sent to the Stroke Cues Filtration sub-block for further filtration.

**Stroke Cues Filtration:** Generally, strokes of each text area can be regarded as the connected region surrounded by a series of edges. Inspired by previous edge detection methods [44], [45], we heuristically model fine-grained stroke-level representation from orthogonal directions. Particularly, a regular convolution and multi-scale orthogonal convolutions [46] with kernel sizes of $3(5, 7) \times 1$ and $1 \times 3(5, 7)$ are introduced to improve the expressive ability of obtained stroke cues. Then we use successive convolution operations to achieve cross-channel integration and information interaction, obtaining the attention coefficients for rough stroke cues. However, the features produced by the previous sub-block cannot provide abundant stroke details, so we take the 3-channel RGB features of OTA as auxiliary stroke cues that include sophisticated texture details. On this basis, spatial attention is performed by multiplying auxiliary stroke cues, and the corresponding attention coefficients to handle rough stroke cues, performing cues filtration, and removing redundant interference from the background. Finally, we aggregate the outputs of two sub-blocks by concatenating the corresponding feature representations and passing them through additional convolution and sigmoid layers to produce high-quality strokes with fine-grained details.

**Advantages Compared with Existing Attention Methods:**

When compared to the general attention method, such as the general SE block [47], the introduced method has two main merits: (1) SE block performs only squeezing and excitation operations on the obtained channel features (by adopting average global pooling operation on the input image features) to get the weights of each channel. However, the input image features are not sufficiently considered to guide more discriminative channel attention weights. In contrast, the Text Features Distillation performs squeezing and excitation operations on the input image features and the channel weight features simultaneously. Significantly, we use both convolution and deconvolution operations to squeeze and excite input features, better capturing inter-channel dependencies based on image features. (2) SE block performs only single-dimensional (along with the channel) attention on the input features, focusing on channel dependencies while ignoring spatial dependencies. We adopt hierarchical
attentions, which first perform channel attention (along the channel) in the Text Features Distillation. It then performs spatial attention (along with the feature map) in the Stroke Cues Filtration, fusing channel and spatial dependencies to extract more informative features. The former achieves semantic distillation and obtains rough stroke cues (high-level image features). The latter uses auxiliary stroke cues (low-level image features) to achieve cues filtration and remove redundant background interference, finally producing high-quality strokes with fine-grained details.

Loss: There are three losses in the SAPN module, which could be formulated as:

\[ L_{SAPN} = L_{cls} + L_{reg} + L_{stroke}, \]

where \( L_{cls} \) can be further decomposed as \( \lambda_1 L_{ta} + \lambda_2 L_{tca} \). \( L_{ta} \) indicates the OHEM loss [48] for TA, and \( L_{tca} \) represents the cross-entropy loss for TCA. Besides:

\[ L_{reg} = \lambda_3 (L_{sin} + L_{cos}) + L_h, \]

where \( L_{sin} \) and \( L_{cos} \) denote the regression loss for the predicted angles. For \( L_h \), we adopt the method in [25] to obtain the loss of height regression.

Furthermore, we employ the hybrid loss to guide stroke detection, defined as \( L_{stroke} = \lambda_4 L_{MSE} + \lambda_5 L_{SSIM} \). \( L_{MSE} \) is adopted to ensure the pixel-wise accuracy, while stroke structure optimization is guided via \( L_{SSIM} \) [49]. During training, \( \lambda_1, \lambda_2, \lambda_3, \lambda_4 \) and \( \lambda_5 \) are tunable but simply set to 1 for all experiments. The detailed equations for \( L_{sin}, L_{cos}, L_h, L_{MSE} \) and \( L_{SSIM} \) can refer to the supplementary materials.

B. Hierarchical Relation Graph Network

Considering it is difficult for a graph model composed of only single-level nodes to achieve grouping node classification results into the arbitrary-shaped text area and efficiently splitting the close text instances simultaneously, whether based on stroke-level nodes or text-level nodes. So we extract both stroke- and text-level nodes to obtain the corresponding graph nodes on both sides. It is noted that each text instance could be divided into a series of ordered quadrilateral components [25] along the direction of the text writing. We built an isomorphic stroke graph and a heterogeneous text graph separately for hierarchical relations reasoning and linkages prediction by extracting candidate bounding boxes from text and stroke levels. As shown in Fig. 4, we extract a series of text-level proposals within the predicted text area by following the method in [15], while shrinking the size of boxes to obtain corresponding stroke-level proposals within the detected region of strokes. In the meantime, NMS [50] and boundary determination [51] are introduced to limit the total number of generated proposals (graph nodes). Please refer to the supplementary materials on graph generation for more details.

We adopt complementary representation for feature initialization of nodes at both levels, i.e., geometric and content embedding. Concretely, circular functions [52] are applied to get geometric embedding by encoding the geometric attributes (including the center coordinates \((x, y)\), width \(w\), height \(h\), sine and cosine values \((\sin \theta, \cos \theta)\) of the orientation of each proposal shown in Fig. 4) into high dimensional spaces, while content embedding is obtained by sending the predicted feature map with the geometric attributes of each proposal to the RoIAlign layer [9].

Based on the built isomorphic stroke graph, which contains only stroke-level nodes, we first adopt the attention mechanism proposed in [36] to model diverse relationships from both aspects of structure and content for attention-guided representation. Concretely, for a stroke node \( s \) and its neighbor \( n \) \((n \in N)\) where \( N \) denotes the neighbor set, an attention coefficient between them can be formulated as:

\[ \alpha_{sn} = \frac{\exp(\text{LeakyReLU}(a^T [W_h s \oplus W_h n]))}{\sum_{k \in N} \exp(\text{LeakyReLU}(a^T [W_h s \oplus W_h k]))}, \]
where $h_s$ and $h_t$ denote the feature vectors of two nodes, $W$ and $a$ are trainable parameters, and $\oplus$ means concatenation. After that, the representation of node $s$ will be updated as $h_s^{\text{updated}} = \text{Sigmoid}((\sum_{k \in \mathcal{N}} a_{sk} \cdot W h_k))$. By learning to increase the attention weight of adjacent nodes that jointly appear in the direction along with the writing while suppressing the weight of adjacent nodes that appear in the direction perpendicular to the writing or in other directions, the built stroke graph performs a distinguishable aggregation to identify the text instances that are close to each other. The updated stroke nodes and text-level nodes are then adopted for hierarchical relations reasoning in the heterogeneous text graph.

Each text node is connected with extra stroke-level nodes apart from its text-level neighbors in the built text graph. For a text node $t$, we filter out the top-3 nearest stroke nodes based on the distance of their centers to $t$. A two-stage information aggregation process is then utilized to update text-level representation. In the first stage, a weighted average aggregator is employed where the weights come from the normalized adjacency matrix $A$ among text-level nodes, defined as:

$$AGG_{\text{text \_ level}} : h_t^{\text{stage1}} = \sum_{m \in \mathcal{N}(t)} \alpha_{t,m} h_m,$$

where $\mathcal{N}(t)$ indicates the 1-hop neighbor set of $t$.

Because various stroke nodes contain information from different parts of the text area, contributing distinctively to the representation of each text node, in the second stage, we perform an expressive information aggregation from stroke nodes to text nodes. A soft mask is first computed as follows:

$$h_t = \text{Meanpooling}(\mathcal{F}(\tilde{N}(t))),\quad m(h_t) = \text{Sigmoid}(\text{MaxPooling}(\mathcal{F}(\tilde{N}(t)) \cdot M \cdot h_t)),$$

where $\tilde{N}(t)$ indicates the stroke-level 1-hop neighbor set of $t$, while $\mathcal{F}(\cdot)$ denotes the corresponding feature vectors. $M$ is a trainable weight matrix, and $\cdot$ represents the matrix multiplication. The obtained $m(h_t)$ serves as the information gatekeeper, which will be multiplied by the feature of stroke-level neighbors in the heterogeneous graph:

$$AGG_{\text{stroke \_ level}} : h_t^{\text{stage2}} = \mathcal{F}(\tilde{N}(t)) \otimes m(h_t),$$

where $\otimes$ denotes the element-wise product. In this way, the stroke-level aggregation for each text node is restricted to a dynamic sub-part of the whole graph. The informative stroke nodes will be encouraged to perform aggregation operations, and the leftovers will be penalized. Besides, this mechanism is conducive to eliminating irrelevant nodes when learning local details, resulting in an efficient learning architecture while stabilizing the training process.

After that, the aggregation from both levels is fused by a gated sum function:

$$h_t^{\text{updated}} = \text{Fuse}(AGG_{\text{text \_ level}}, AGG_{\text{stroke \_ level}}),$$

where $\text{Fuse}(a, b) = p \cdot a + (1 - p) \cdot b$, and $p = \text{Sigmoid}(W_p[a \otimes b] + b_p)$. $W_p$ and $b_p$ are trainable parameters. Finally, all updated representation of text nodes (denoted as $H$) are utilized to predict the linkage relations from each center node to its neighbors, by the modified graph convolution [25]: $P = \text{Softmax}(H \oplus L \cdot H) W_p$, where $L$ denotes symmetric normalized Laplacian of the adjacency matrix $A$, and $W_p$ is the weight parameter. The outputs from the last graph layer are used to predict linkages which are finally grouped for locating arbitrary-shaped text instances. The cross-entropy loss is adopted for training.

It is noted that DRRG [25] adopts a similar operation to generate text components for detection, so we specify the differences between DRRG and our method to highlight the advantages of our approach. First, DRRG obtains only text-level nodes by extracting text-level proposals, while StrokeNet extracts both text- and stroke-level nodes to get the corresponding graph nodes at both levels. Second, DRRG performs relations reasoning among text-level nodes to infer the linkage of text proposals on the built isomorphic graph, while StrokeNet first conducts graph-based reasoning among stroke-level nodes on the constructed isomorphic stroke graph and then performs hierarchical relations reasoning in the heterogeneous text graph (containing both text- and stroke-level nodes). In summary, DRRG performs coarse-grained relations reasoning among only text-level nodes, which can not accurately separate the text instances standing close to each other. In contrast, StrokeNet can fully explore the information of each text instance through hierarchical relations reasoning. Significantly, finer-grained attention-guided reasoning among stroke-level nodes is first performed, which is then utilized to supervise the linkage of text-level nodes by strengthening the ability to separate the close text instances.

During inference, we first apply Stroke Assisted Prediction Network to obtain the multi-level predictions of each text instance and then use the predicted text area and the corresponding stroke segmentation to construct multiple local graphs at both levels.

Next, Hierarchical Relation Graph Network is used to infer the relations at both levels and predicts linkages among text-level nodes. According to the classification results, text nodes are grouped by Breath First Search method [53] and sorted by Min-Path algorithm [54], to obtain the boundary of arbitrary-shaped text by sequentially linking the mid-point of both the top and the bottom in ordered text nodes.

IV. EXPERIMENTS

A. Benchmarks and Implementation Details

We evaluate StrokeNet on six benchmarks: CTW-1500, Total-Text, MSRA-TD500, ICDAR2015, ICDAR 2017 MLT and ICDAR 2019 MLT. Besides, we introduce SynthStroke to pre-train the whole framework, which is helpful for subsequent performing fine-tuning and evaluation on real scene benchmarks. We synthesize the whole dataset based on the 8000 images, which contain no texts and are collected from the open public repository. Precisely, SynthStroke consists of 800 thousand synthetic images with approximately 8 million synthetic word instances. It is noted that another synthetic dataset, namely

\[ \text{[Online]. Available: https://github.com/HCIILAB/Scene-Text-Removal} \]
Fig. 5. An example of SynthStroke with annotations.

Fig. 6. Synthetic examples of SynthStroke.

SynthText [63], which is also synthesized from the mentioned resource and commonly applied for the pre-training of many text detectors in previous research. Since SynthText has maintained a relatively conservative variation of text attributes such as font and rotation angle during the synthesis process, which limits the diversity of text forms. In contrast, we perform augmentations of the dataset to ensure the variety of the whole SynthStroke. Concretely, we apply the different configurations of parameters through the image synthesis, including font, font size, rotation angle, the number of alphabets, and numbers, to control the generation of training samples. A comprehensive comparison of SynthStroke and SynthText is summarized in Table IV-A.

Compared to SynthText, the introduced SynthStroke contains more diverse samples equipped with stroke labels, mainly including some tiny text instances which help train a more powerful text detector. Fig. 5 shows an example of our simulated dataset, while Fig. 6 visualizes several synthetic image examples. A visual comparison of SynthStroke and SynthText is summarized in Fig. 7. In addition, we extract the pseudo stroke labels of evaluated benchmark datasets. Specifically, we first sample 100 thousand images in the introduced SynthStroke and extract the text areas (denoted as $S_t$) of these images according to their text-level bounding boxes. Meanwhile, we also generate the text areas (denoted as $B_t$) of images from benchmarks according to the corresponding text-level bounding boxes. We then utilize $S_t$ to pre-train the introduced Stroke Assisted Prediction Network (SAPN) individually. Afterward, we use the pre-trained SAPN model to extract strokes of $B_t$, which are finally posted in the full-black background images of the exact sizes according to the position of the corresponding images in the benchmark.

For the whole framework, we first pre-train our StrokeNet with the introduced SynthStroke for 5 epochs, and then perform fine-tuning on benchmark datasets for 1000 epochs by extracting their pseudo stroke labels. The model obtained in this way is denoted as StrokeNet (S). To improve fairness, we pre-train another model, namely StrokeNet (T) on SynthText after extracting its pseudo stroke labels and use the same evaluation criteria on benchmarks. All experiments are performed on a single image resolution.

B. Comparison With State-of-The-Art Methods

Close and Arbitrary-shaped Text Detection: We compare StrokeNet with several state-of-the-art methods on two curved benchmarks in Table I, including CTW-1500 and Total-Text. Benefiting the introduced HRGN, our method achieves promising results in representing close and arbitrary-shaped texts, especially with varying degrees of curvature.

Small and Low-resolution Text Detection: We evaluate our method on ICDAR 2015, which contains a lot of small and low-resolution text instances. As shown in Table I, StrokeNet achieves consistent and competitive performance in recall, precision, and H-mean, because the introduced SAPN module plays an essential role in effectively capturing the representation of small and low-resolution strokes.

Multi-language Text Detection: To test the robustness of StrokeNet to multiple languages with long texts, we evaluate our method on MSRA-TD500, ICDAR 2017 MLT, and ICDAR 2019 MLT benchmarks. The quantitative results are listed in Table II. Significantly, the ICDAR 2019 MLT benchmark evaluations verify that StrokeNet achieves superior performance with continuous stability on a large-scale dataset.

Qualitative results shown in Fig. 8 can demonstrate the effectiveness of the proposed method in the above three aspects.
TABLE I
EXPERIMENTAL RESULTS ON CTW-1500, TOTAL-TEXT, AND ICDAR 2015. THE TOP TWO BEST SCORES ARE HIGHLIGHTED IN BOLD. FPS IS FOR REFERENCE ONLY BECAUSE THE EXPERIMENTAL ENVIRONMENTS ARE DIFFERENT. WE REPORT THE FPSs, EACH OF WHICH WAS REPORTED IN THE ORIGINAL PAPER.

| Method            | CTW-1500   | Total-Text | ICDAR 2015 |
|-------------------|------------|------------|------------|
|                   | Recall     | Precision  | Hmean      | FPS | Recall     | Precision  | Hmean      | FPS | Recall     | Precision  | Hmean      | FPS |
| TextSnake [15]    | 83.3       | 67.9       | 75.6       | 1.1 | 74.5       | 82.7       | 78.4       | -  | 84.9       | 80.4       | 82.6       | 1.1 |
| GAPF [20]         | 79.6       | 83.2       | 81.3       | -   | 76.8       | 83.6       | 80.1       | -  | 81.6       | 86.4       | 84.1       | -
| Das et al. [4]    | 83.3       | 66.1       | 84.6       | 1.0 | 78.5       | 84.6       | 81.5       | 1.3 | 82.7       | 86.2       | 84.4       | 2.1 |
| PSNet [16]        | 79.7       | 84.8       | 82.2       | 3.9 | 84.0       | 78.0       | 80.9       | 3.9 | 84.5       | 86.9       | 85.7       | 1.6 |
| TextiMountain [19]| 82.4       | 81.3       | 81.9       | -   | -          | -          | -          | -  | 83.1       | 89.5       | 86.2       | 9.7 |
| CRAFT [42]        | 81.1       | 86.0       | 83.5       | -   | 79.9       | 87.6       | 83.6       | -  | 84.3       | 89.8       | 86.9       | 8.6 |
| DB [55]           | 80.2       | 86.9       | 83.4       | 22.0 | 82.5       | 87.1       | 84.7       | 32.0 | 83.2       | 91.8       | 87.3      | 12.0 |
| ReLaText [26]     | 83.3       | 86.2       | 84.8       | 10.6 | 83.1       | 84.8       | 84.0       | -  | -          | -          | -          | -
| PRPN [22]         | 79.7       | 83.2       | 81.4       | -   | -          | -          | -          | -  | 83.7       | 85.3       | 86.0       | 5.0 |
| DRRG [25]         | 83.0       | 85.9       | 84.5       | -   | 84.9       | 86.5       | 85.7       | -  | 84.7       | 88.5       | 86.6       | -
| ContourNet [45]   | 84.1       | 83.7       | 83.9       | 4.5 | 83.9       | 86.9       | 85.4       | 3.8 | 86.1       | 87.6       | 86.9       | 3.5 |
| ABCNet [56]       | 78.5       | 84.4       | 81.6       | -   | 81.3       | 87.9       | 84.5       | 17.9 | -          | -          | -          | -
| FCENet [57]       | 83.4       | 87.6       | 85.5       | -   | 82.5       | 89.3       | 85.8       | -  | 82.6       | 90.1       | 86.2       | -
| CRTk [18]         | 83.3       | 88.2       | 85.7       | 5.2  | 82.9       | 88.4       | 85.6       | 7.9 | 83.3       | 89.5       | 86.4       | 9.3 |
| Zhang et al. [5]  | 80.8       | 85.1       | 82.9       | 1.4  | 82.7       | 87.6       | 85.1       | 1.4 | 85.5       | 89.1       | 87.3       | 1.4 |
| ABPN [21]         | 81.5       | 87.8       | 84.5       | 12.2 | 84.7       | 90.3       | 87.4       | 10.3 | -          | -          | -          | -
| SDM-ResNet-50 [58]| 84.4       | 88.4       | 86.4       | -   | 86.0       | 90.1       | 88.4       | -  | 89.3       | 92.0       | 90.6       | -
| StrokeNet (T)     | 86.3       | 88.2       | 87.2       | 6.0  | 87.8       | 89.0       | 88.4       | 6.0 | 89.2       | 91.7       | 90.4       | 5.2 |
| StrokeNet (S)     | 86.9       | 88.7       | 87.8       | 6.0  | 88.2       | 89.5       | 88.8       | 6.0 | 89.6       | 92.3       | 90.9       | 5.2 |

Fig. 8. Visualizations on benchmarks. The first column is input images. The second column is the predicted strokes (only sample part of each area by red dotted rectangles for subsequent display). The third column is the text-level box generation. The fourth column is the detected texts.
manner. In comparison, StrokeNet is an end-to-end framework, achieving better results while requiring less inference time. For instance, StrokeNet obtains the 90.4% Hmean rate with the speed of 5.2 FPS on ICDAR 2015 benchmark based on SynthText as the pre-training dataset, while PSENet only gets the 85.7% Hmean rate with 1.6 FPS.

DRRG [25] only extracts text-level proposals and uses a graph convolution network to learn the relations between single-level text nodes, which has a good effect on detecting arbitrary-shape texts but cannot separate the text instances standing close to each other accurately. By contrast, we further extract stroke-level proposals within the segmented strokes. On this basis, finer-grained stroke-level nodes are utilized to supervise the linkage of text-level nodes through hierarchical relations reasoning of the built heterogeneous graph network. Eventually, StrokeNet dramatically improves the performance across all benchmarks and remarkably strengthens the ability to separate the close text instances. Significantly, our approach achieves the 87.2% and 88.4% Hmean rate on CTW-1500 and Total-Text benchmarks, substantially outperforming DRRG with only the 84.5% and 85.7% Hmean rate, respectively. Fig. 9 provides a visualization of two strategies.

**Speed Analysis:** The comparison of detection speed is provided in Table I and Table II. The inference speed of our StrokeNet is competitive compared with most methods, including the latest works ContourNet [45] and Zhang et al. [5]. When compared with other Graph Neural Networks (GNNs) based methods, such as TextFuseNet [64] and ABPN [21], which adapts the original Graph Convolutional Network (GCN) [35] to the detection task, our approach achieves the 87.2% and 88.4% Hmean rate on CTW-1500 and Total-Text benchmarks, substantially outperforming DRRG with only the 84.5% and 85.7% Hmean rate, respectively. Fig. 9 provides a visualization of two strategies.

**Failure Cases:** Furthermore, Fig. 10 indicates some of the failure cases produced by StrokeNet. The examples shown in the first row suggest that a few strokes detected by our method are blurred, but this problem does not tend to have a distinct impact on the detection of the corresponding text area. The second row shows that StrokeNet may mistakenly detect the edges of some background objects in stroke-level detection, leading to the detection results containing undesirable backgrounds. This problem is mainly due to the introduction of orthogonal convolutions, which are sensitive to edge features. In the future, we consider further optimization on detecting more accurate strokes while suppressing background edges.

**C. Ablation Study**

The first module SAPN contains two blocks for our proposal, marked as text-level prediction (TLP) block and stroke-level prediction (SLP) block. While the second module HRGN includes two main parts, including the built stroke graph (SG) and text graph (TG). We conduct an ablation study by removing SLP and SG, leading to the variant (TLP + TG\(^\ast\)), which only adopts text-level outputs for the subsequent single-level graph reasoning process. TG\(^\ast\) means that there are no stroke nodes for the built text graph, and only text-level aggregation is performed. Table IV summarizes the results of our models with different settings on MSRA-TD500. We adopt TLP block as our baseline, which performs text-level prediction and only obtains the rectangle result of each text instance. Then we introduce the SLP block to model fine-grained stroke-level representation in the way of multi-task learning, improving the performance by 5.9% in Hmean. When TG\(^\ast\) is further added, single-level relations reasoning and aggregation are performed to localize arbitrary-shaped texts. After that, we further introduce SG to conduct attention-guided aggregation and the feature fusion between nodes of both levels, significantly promoting the final
Some failure cases. Column a: input images with the solid red rectangles as the detection focus. Column b: the corresponding stroke-level outputs, where the dotted rectangular boxes indicate the detection problems. Column c: the final text detection results.

**TABLE III**

|                  | SynthStroke | SynthText |
|------------------|-------------|-----------|
| Image Quantity   | 800 thousand| ~ 800 thousand |
| Text Instance Quantity | ~ 8 million | ~ 8 million |
| Word-level Bounding Box | yes | yes |
| Character-level Bounding Box | no | yes |
| Stroke-level Segmentation | yes | no |
| Font Variation   | yes (100 types) | yes (< 50 types) |
| Orientation Variation | yes (0° ~ 360°) | yes (0° ~ 180°) |
| Font Size Variation | yes (5 ~ 80) | yes (20 ~ 50) |

**TABLE IV**

| Model                    | Module_1 | Module_2 | R   | P   | H   | Gain(%) |
|--------------------------|----------|----------|-----|-----|-----|---------|
| baseline (S)             | TLP      | NONE     | 71.6| 73.8| 72.7| ——      |
| Variant_1 (S)            | TLP+SLP  | NONE     | 74.2| 79.9| 77.0| 5.9     |
| Variant_2 (S)            | TLP      | TG*      | 81.5| 83.9| 82.7| 13.8    |
| StrokeNet (T)            | TLP+SLP  | SG+TG    | 85.2| 87.6| 86.4| 18.8    |
| StrokeNet (T)            | TLP+SLP+1| SG+TG    | 82.3| 84.9| 83.3| 14.6    |
| StrokeNet (T)            | TLP+SLP+2| SG+TG    | 82.8| 85.0| 83.9| 15.4    |
| StrokeNet (T)            | TLP+SLP+3| SG+TG    | 84.0| 85.7| 84.8| 16.6    |
| StrokeNet (S)            | TLP+SLP  | SG+TG    | 85.6| 88.3| 86.9| 19.5    |

**TABLE V**

| L_reg | L_stroke | | | | | |
|-------|----------|---|---|---|---|---|
|       |          | ICDAR 2015 | ICDAR 2017 MLT |
|       |          | R  | P  | H  | R  | P  | H  |
| ✔     | ✔        | 84.2 | 86.1 | 85.1 | 65.6 | 75.9 | 70.4 |
| X     | ✔        | 87.2 | 88.7 | 87.9 | 75.0 | 83.7 | 79.1 |
| X     | ✔        | 69.4 | 65.8 | 67.6 | 49.6 | 45.7 | 47.6 |
| ✔     | ✔        | 89.6 | 92.3 | 90.9 | 78.4 | 87.1 | 82.5 |

Fig. 10. (a): Original image, (b): The coarse stroke maps obtained by Text Features Distillation sub-block, (c) The refined stroke maps obtained by Stroke Cues Filtration sub-block (using the single-scale orthogonal convolution with the kernel size of 3 × 1 and 1 × 3), and (d): The refined stroke maps obtained by Stroke Cues Filtration sub-block (adopting the described multi-scale orthogonal convolutions).

Furthermore, we provide visualizations of ablation on the introduced Stroke-level Prediction block in Fig. 11. Although the coarse stroke maps obtained from the Text Features Distillation sub-block contain some noises unrelated to the strokes, the Stroke Cues Filtration sub-block can further filter out these noises to get the refined stroke maps. Besides, the adopted multi-scale orthogonal convolutions achieve better results than the single-scale orthogonal convolution.

**D. Justification of External Dataset**

We perform a justification of introduced dataset by showing a significant improvement of StrokeNet after pre-training on SynthStroke compared to SynthText in Table I, II and IV. Besides, we apply our SynthStroke to pre-train comparative methods and the results are shown in Table VI. From the table, it is concluded...
that SynthStroke is beneficial for off-line pre-training of widely methods. Moreover, we modify DRRG [25] which is a recently proposed text detector that also adopts the graph networks by adding the SLP block on top of its text region detection module. In the meantime, training is performed in the way of multi-task learning, while both locations and stroke-level masks are output during inference. The corresponding results are revealed in Table VI, demonstrating the applicability of stroke-level representation for related methods in the text detection field. It also shows that exploring the stroke-level representation of text area is essential for further promotion of performance.

Application: We have developed an OCR translation tool based on StrokeNet, shown in Fig. 12. Please refer to supplements for more details. In brief, the StrokeNet is first called to output the stroke- and text-level detection results, which will be separately fed into a text recognition model and an inpainting module to improve the performance on both sides. The machine translation model is run by calling Google translate API, including a language identification API.

V. CONCLUSION

We propose StrokeNet accompanied by SynthStroke dataset with stroke-level annotations for scene text detection. Our method focuses on multi-level representations and hierarchical relation reasoning of text regions, efficiently detecting extremely small or low-resolution strokes and splitting close or arbitrary-shaped texts. Future work should explore the end-to-end fashion of text detection to support various downstream tasks, such as text recognition and text removal.

REFERENCES

[1] X. Liu and W. Wang, “Robustly extracting captions in videos based on stroke-like edges and spatio-temporal analysis,” IEEE Trans. Multimedia, vol. 14, no. 2, pp. 482–489, Apr. 2012.

[2] L. Wu, P. Shivakumara, T. Lu, and C.-L. Tan, “A new technique for multi-oriented scene text line detection and tracking in video,” IEEE Trans. Multimedia, vol. 17, no. 8, pp. 1137–1152, Aug. 2015.

[3] S. Karaoğlu, R. Tao, T. Gevers, and A. W. Smeulders, “Words matter: Scene text for image classification and retrieval,” IEEE Trans. Multimedia, vol. 19, no. 5, pp. 1063–1076, May 2017.

[4] P. Dai, H. Zhang, and X. Cao, “Deep multi-scale context aware feature aggregation for curved scene text detection,” IEEE Trans. Multimedia, vol. 22, no. 8, pp. 1969–1984, Aug. 2020.

[5] S. Zhang, Y. Liu, L. Jin, Z. Wei, and C. Shen, “OPMP: An omnidirectional pyramid mask proposal network for arbitrary-shape scene text detection,” IEEE Trans. Multimedia, vol. 23, pp. 454–467, 2021.

[6] Y. Wang et al., “R-Net: A relationship network for efficient and accurate scene text detection,” IEEE Trans. Multimedia, vol. 23, pp. 1316–1329, 2021.

[7] J. Li et al., “Attentive contexts for object detection,” IEEE Trans. Multimedia, vol. 19, no. 5, pp. 944–954, May 2017.

[8] C. Chen and Q. Ling, “Adaptive convolution for object detection,” IEEE Trans. Multimedia, vol. 21, no. 12, pp. 3205–3217, Dec. 2019.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 2961–2969.

[10] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, “TextBox: A fast text detector with a single deep neural network,” in Proc. AAAI Conf. Artif. Intell., 2017, vol. 31, no. 1, pp. 4161–4167.

[11] P. He et al., “Single shot text detector with regional attention,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 3047–3055.

[12] C. Zhang et al., “Look more than once: An accurate detector for text of arbitrary shapes,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 10552–10561.

[13] J. Ma et al., “Arbitrary-oriented scene text detection via rotation proposals,” IEEE Trans. Multimedia, vol. 20, no. 11, pp. 3111–3122, Nov. 2018.

[14] P. Lya, 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 Proc. Eur. Conf. Comput. Vis., 2018, pp. 67–83.

[15] S. Long et al., “TextSnake: A flexible representation for detecting text of arbitrary shapes,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 20–36.

[16] W. Wang et al., “Shape robust text detection with progressive scale expansion network,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 9336–9345.

[17] M. Xue et al., “Arbitrarily-oriented text detection in low light natural scene images,” IEEE Trans. Multimedia, vol. 23, pp. 2706–2720, 2021.

[18] C. Cui, L. Lu, Z. Tan, and A. Hussain, “Conceptual text region network: Cognition-inspired accurate scene text detection,” Neurocomputing, vol. 464, pp. 252–264, 2021.

[19] Y. Zhu and J. Du, “TextMountain: Accurate scene text detection via instance segmentation,” Pattern Recognit., vol. 110, 2021, Art. no. 107336.

[20] J. Yang, Z. Zhang, and X. Wang, “GAPF: Curve text detection based on generative adversarial networks and pixel fluctuations,” in Proc. Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl., 2021, pp. 545–552.

[21] S.-X. Zhang, X. Zhu, C. Yang, H. Wang, and X.-C. Yin, “Adaptive boundary proposal network for arbitrary shape text detection,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 1305–1314.

[22] Y. Zhong et al., “PRPN: Progressive region prediction network for natural scene text detection,” Knowl.-Based Syst., vol. 236, 2022, Art. no. 107767.

[23] X. Zhou et al., “EAST: An efficient and accurate scene text detector,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 5551–5560.

[24] Z. Zhong, L. Sun, and Q. Huo, “An anchor-free region proposal network for faster R-CNN-based text detection approaches,” Int. J. Document Anal. Recognit., vol. 22, no. 3, pp. 315–327, 2019.

[25] S.-X. Zhang et al., “Deep relational reasoning graph network for arbitrary shape text detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 9699–9708.

[26] C. Ma, L. Sun, Z. Zhong, and Q. Huo, “RelaText: Exploiting visual relationships for arbitrary-shaped scene text detection with graph convolutional networks,” Pattern Recognit., vol. 111, 2021, Art. no. 107684.

[27] T. Guan et al., “Adaptively selective scene text detection with refined feature-attentive network,” IEEE Trans. Circuits Syst. Video Technol., early access, Mar. 2, 2022, doi: 10.1109/TCSVT.2022.3165390.

[28] D. Deng, H. Liu, X. Li, and D. Cai, “PixelLink: Detecting scene text via instance segmentation,” in Proc. AAAI Conf. Artif. Intell., 2018, vol. 32, no. 1, pp. 6773–6780.

[29] C. Yao, X. Bui, B. Shi, and W. Liu, “Strokelets: A learned multi-scale representation for scene text recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2014, pp. 4042–4049.

[30] S. Tian, S. Lu, and C. Li, “WeText: Scene text detection under weak supervision,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 1492–1500.
[31] P. Lyu, C. Yao, W. Wu, S. Yan, and X. Bai, “Multi-oriented scene text detection via corner localization and region segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 7553–7563.

[32] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[33] W. Liu et al., “SSD: Single shot multibox detector,” in Proc. Eur. Conf. Comput. Vis., Springer, 2016, pp. 21–37.

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 779–788.

[35] M. Welling and T. N. Kipf, “Semi-supervised classification with graph convolutional networks,” in Proc. Int. Conf. Learn. Representations, 2016.

[36] P. Veličković et al., “Graph attention networks,” in ICLR, 2018, pp. 11753–11762.

[37] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 3431–3440.

[38] T. He, W. Huang, Y. Qiao, and J. Yao, “Text-attentional convolutional neural network for scene text detection,” in Proc. IEEE Trans. Image Process., vol. 25, no. 6, pp. 2529–2541, Jun. 2016.

[39] B. Shi, X. Bai, and S. Belongie, “Detecting oriented text in natural images by linking segments,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2550–2558.

[40] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, “Character region awareness for text detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 9365–9374.

[41] T.-Y. Lin et al., “Feature pyramid networks for object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2117–2125.

[42] H. A. Patel, K. S. Shekotkar, and M. Student, “Detecting text or character in natural scenes with stroke width transform,” Int. J. Innov. Res. Comput. Commun. Eng., vol. 3297, no. 5, pp. 1–8, 2007.

[43] Y. Wang et al., “ContourNet: Taking a further step toward accurate arbitrary-shaped scene text detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 11753–11762.

[44] J. Wang, Y. Chen, R. Chakraborty, and S. X. Yu, “Orthogonal convolutional neural networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 11505–11515.

[45] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 7132–7141.

[46] Y. Wang et al., “ContourNet: Taking a further step toward accurate arbitrary-shaped scene text detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 516–522.