Chapter 5

Fainted TextSpotter

Text instances in natural scene images typically contain immense semantic information that conveys meaningful information to the reader. Scene text instances are hard to classify due to wide variations in underlying background, fonts, textures, and lighting conditions. In the previous Chapter 3 and Chapter 4, we presented two text spotters to fix the challenges obtained by cluttered and blurry scene images for text detection and recognition. In this chapter, we handle the problem of faint text edges in scene images. We propose a deep network architecture, named as *Fainted TextSpotter*. Scene text spotting in an adverse weather outbreak and hostile conditions is a more complicated process due to the presence of fog, rain, and smog during image capturing. This results in an outbreak of faint edges, poor contrast, low illumination, and inter-class interference, as shown in Fig 5.1. This chapter considers the faint edges challenge in the text spotting process. While identifying a text instance in a scene image with fainted edges, we humans first emphasize on all the edges in the image (both fainted and non-fainted) and then focus the attention of edges, which seems to associate with the text instances. We follow the same model by learning semantic edge supervised feature maps followed by focusing attention on pixel-wise spatial features and channel-wise relationships. This helps to resolve the problem faint text edges due to poor contrast.

In this chapter, we propose an end-to-end trainable deep neural network that can address the issue of spotting arbitrary-oriented text instances in scene images, captured in adverse meteorological conditions. It localizes words, predicts script class, and performs word spotting for every rotated bounding box. It is a multilingual fast scene text spotter that utilizes hierarchical spatial context, channel-wise inter-dependencies, and semantic edge supervision to localize and recognize words and predict script class in scene images using smartphones. We explore inter-class interference to reduce the misclassification problem. An efficient recognition module for character segmentation, word-level recognition, and script identification is incorporated. We use benchmark datasets to demonstrate the efficacy of our spotting network.

Figure 5.1: Illustration of natural images (first row) representing the necessity of Fainted TextSpotter. Second and third rows are the recognized scene text instances using FOTS [1] (baseline) and the proposed network, respectively.

The rest of the chapter is organized as follows. In the first section, we describe the proposed architecture of the text spotter. In the next section, we demonstrate the

experimental results. In the last section, we conclude the chapter.

5.1 Proposed Architecture

In this section, we propose a robust and efficient network, denoted by *Fainted TextSpotter*, for arbitrary-oriented text spotting in scene images. It has the following modules:

• First, we incorporate a light-weight backbone network using MobileNetV2. To able to improve text edge categorization for a holistic scene text interpretation, we integrate category-aware semantic edge learning with our network. Semantic edge supervision addresses the problem of an edge pixel being associated with multiple classes due to a hostile environment.

• Second, we design a context encoding module that incorporates class-aware inter-channel relationships to recalibrate adaptively and trilayer dilated feature pyramid to capture hierarchical spatial context. A range of dilation rates and large effective receptive field eliminates the gridding artifacts. It helps in robust and dense localization of text instances.

• Third, we introduce a light-head region proposal network with oriented RoIs to obtain oriented region proposals by the involvement of classification confidence along with localization accuracy. We map the features of a rotated region proposal using bilinear sampling to a canonical dimension. This normalizes the rotation and scale maintaining aspect-ratio and positioning of each character. The localization accuracy score compensates for the missing text region by conserving the entire text region during training.

• Finally, we utilize inter-class distance and intra-class density using Gaussian softmax to address the issue of misclassification due to hostile environment and ambient noises. We also add a recognition module for character segmentation, word-level recognition, and identification of corresponding text script using Bi-CLSTM, self-attention, and circulant matrix.

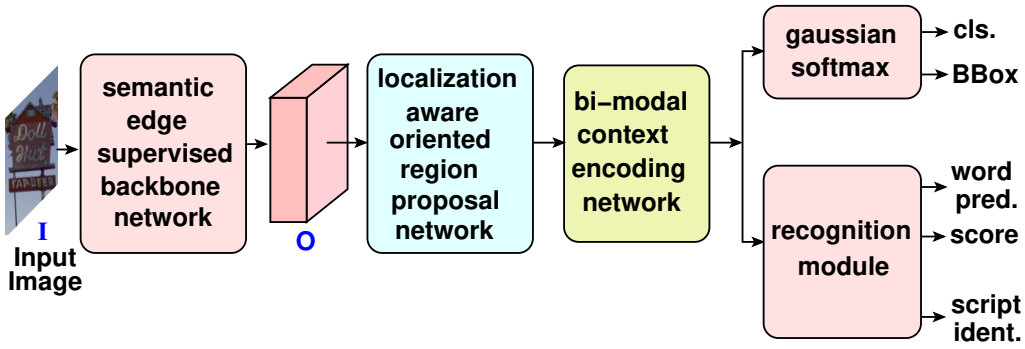


Figure 5.2: Overall architecture of Fainted TextSpotter.

5.1.1 Semantic Edge Supervised Backbone Network

The aim of this section is to obtain a backbone network that encodes a multi-label category-aware semantic edge learning by allowing low-level features to participate and augment high-level semantic classification using a skip-layer architecture. It is light-weight and hardware-efficient in nature. We have not restricted a pixel to be associated with either text or non-text class. Hence, we incorporate the basic blocks of MobileNetV2 [115]. It uses bottleneck depth separable convolutions with residuals. The architecture of the backbone network consists of a convolution layer that has 32 filters with size 3×3, followed by 37 residual bottleneck layers with one 1×1 convolution layer having 1280 filters, as illustrated in Fig. 5.3.

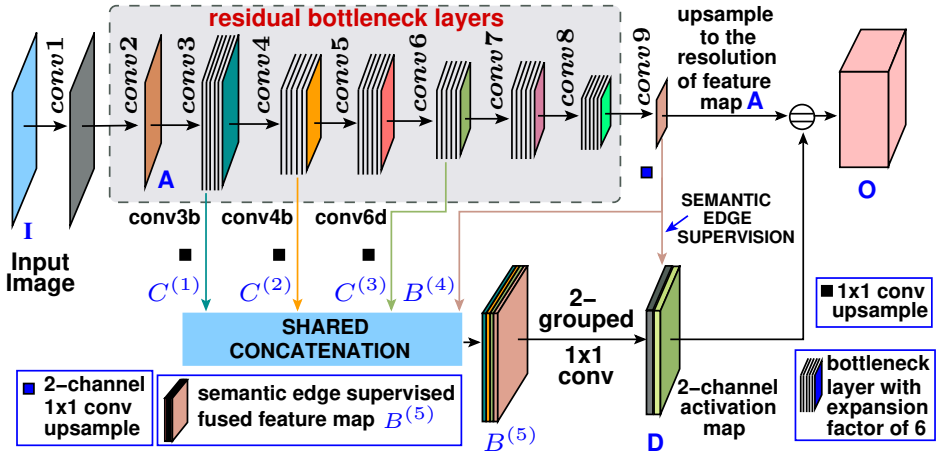


Figure 5.3: Architecture of the backbone network.

Formally, an input image is represented by $I \in \mathbb{R}^{2H \times 2W \times 3}$. We use a filter of size

3×3, dropout, RELU6, and batch normalization during training for residual bottleneck layers. We consider the value of output stride as 16, which can provide a dense feature extraction. We remove the strides at fourth layer and apply category-aware semantic edge supervision [134] using a 1×1 convolution layer and bilinear up-sampling. We impose supervision on top of the last layer ($conv9$) to produce a semantic edge supervised fused feature map B of the same spatial resolution as feature map A (output feature map of $conv2$), *i.e.*, $\{H \times W\}$. We perform side feature extraction to outputs a single channel feature map $C^{(j)}$ and apply shared concatenation to replicate the bottom features on $conv3b$, $conv4b$, and $conv6d$ to separately concatenate with both text and non-text activations as follows:

$$\begin{aligned} B^{(5)} =& \{(C, B^{(4)}_{text}), (C, B^{(4)}_{non-text})\} \\ &\text{where } C = \{C^{(1)}, C^{(2)}, C^{(3)}\}. \end{aligned} \tag{5.1}$$

The output concatenated activation map $B^{(5)}$ is further classified using 2-grouped convolution to generate a bi-channel activation map D. We also classify the non-text edges to keep a check on whether a text edge is wrongly associated with the non-text class. We use 1×1 convolution on D and then apply weighted element-wise **product** with the upscaled feature map $K \in \mathbb{R}^{H \times W \times 1280}$ ($conv9$) to obtain the output feature map $O \in \mathbb{R}^{H \times W \times 1280}$ as follow:

$$O = \{\mathbf{d}'_1 \mathbf{k}_1, \cdots, \mathbf{d}'_{H \times W} \mathbf{k}_{H \times W}\}, \tag{5.2}$$

where $\{\mathbf{d}'_i, \mathbf{k}_i\}$ are the $i - th$ elements of D' and K feature maps.

5.1.2 Bi-modal Context Encoding

The goal of this section is to tackle the fainted text edges, poor contrast, image blurs, perspective distortion, and ambient noises in scene images. We therefore exploit hi-

erarchical spatial features and channel-wise inter-dependencies together to address the issue of unreliable outcomes and ill-localized text edges, as shown in Fig. 5.4. A text edge in the feature map O can be associated with the background also. To reduce the flop count, we compute a feature map $R \in \mathbb{R}^{H \times W \times 128}$ (thin feature map) that has less channels, where channels are decreased from 1280 to 128. We use an adaptive recalibration of inter-channel responses with hierarchical spatial contextual features to model the salient feature maps to intensify its representation explicitly. Hierarchically captured spatial context information is useful for delicate segmentation tasks with complex artifacts. It provides attention to focus on salient spatial regions. The channel dependencies help to rescale the semantic edge supervised feature map to highlight the important channels.

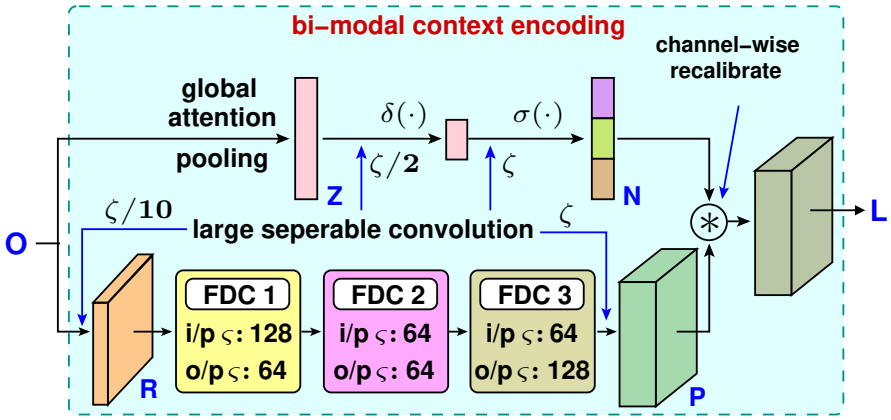


Figure 5.4: Architecture of bi-modal context encoding module.

In the first branch, we consider a large context region with wide spatial variance. To achieve this, we explore a comparatively large receptive field but maintain spatial resolution avoiding striding layers [116]. This prevents overfitting in pixel-wise prediction and an increase in the number of parameters. We incorporate three layers of fused dilated convolutions. We observe the variation in dilation rates can effectively enlarge receptive fields of convolutional kernels. This in turn gathers the surrounding discriminative information and transfers to non-discriminative text regions boosting the presence of

these regions in the text localization maps. This avoids gridding effects. Each layer incorporates three parallel convolutions with 3×3 kernels maintaining the same output dimensions. The dilation rates are $\partial = \{1, 2, 3\}$ and receptive field $RF = 49$. The layers have output channel $\varsigma = \{64, 64, 128\}$. The output feature map P is normalized and channels are enhanced to 1280 (same as O), as shown in Fig. 5.5.

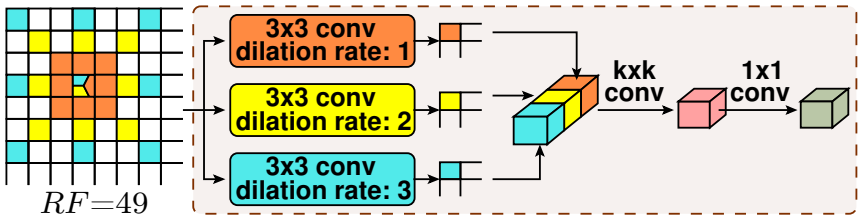


Figure 5.5: Illustration of Fused Dilated Convolutions.

In the second branch, we employ a spatial squeeze and channel excitation, a variant of SENet [135], as shown in Fig. 5.2. We adopt global attention pooling on feature map O to obtain channel-wise statistics. The vector $Z \in \mathbb{R}^{\zeta}$ is obtained by shrinking $\mathbf{e}_l \in \mathbb{R}^{HW \times 1}$ through spatial dimensions $HW = H \times W$ for $l - th$ channel using global attention pooling. The $l - th$ pooled score $\mathbf{z}_{l,k}$ with k class (k ={text, non-text}) is given by:

$$\mathbf{z}_{k,l} = \mathbf{1}^{\tau} \mathbf{e}_l \ \mathbf{w}_k, \tag{5.3}$$

where $\mathbf{1} \in \mathbb{R}^{HW \times 1}$, $\forall$ distinct $l \in \{1, \cdots, \zeta\}$, and $\mathbf{w}_k \in \mathbb{R}^{1 \times 1}$ is the class-specific weights. It helps to embed global spatial knowledge in vector Z, which is transformed by applying sigmoid activation $\sigma(\cdot)$ to obtain the scalar N as follows:

$$N = \sigma(\mathbf{w}_1 \delta(\mathbf{w}_2 Z)), \tag{5.4}$$

such that $\mathbf{w}_1 \in \mathbb{R}^{\zeta \times \beta}$ and $\mathbf{w}_2 \in \mathbb{R}^{\beta \times \zeta}$ are the scores of two fully-connected layers. The parameter β denotes the reduction ratio to encode the channel-wise inter-dependencies, where we choose the value of $\beta = 2$ [135], and the operator $\delta(\cdot)$ indicates the ReLU function.

The output feature map L is obtained by rescaling the transformation output with the activations using channel-wise product between the context enrich feature map P and the channel-excited scalar N as follows:

$$L = [\mathbf{p}_1\mathbf{n}_1, \cdots, \mathbf{p}_{HW}\mathbf{n}_{HW}], \tag{5.5}$$

where $\mathbf{p}_i$ and $\mathbf{n}_i$ are the $i-th$ elements of feature map P and scalar N, respectively.

5.1.3 Localization-aware Oriented Region Proposal Network

The objective of this section is to provide an oriented R-CNN subnet, which also acquires the confidence of localization for precisely regressing bounding boxes followed by mapping of the RoI features into canonical dimension to normalize rotation and scale, as depicted in Fig. 5.2. We first produce a feature map with a small number of channels (thin feature maps) to perform RoI pooling. In our experiments, we find that an improvement in accuracy measure and memory usage is obtained. This is because the computations are reduced during both training and inference by performing RoI pooling on the thin feature maps. We therefore reduce R-CNN overhead by reducing the channels to 128 from 1280 to compute a thin feature map $J \in \mathbb{R}^{H\times W\times 128}$. We introduce RoI pooling to get fixed shaped feature maps. We also incorporate rotated anchors [23] and IoU-Net learning [136] to obtain rotated region proposals and predict the intersection-over-union (IoU) between each rotated bounding box and the corresponding matched ground-truth, respectively. IoU-Net [136] disentangle classification confidence of being text instance and RoI localization accuracy using IoU-guided non-maximal suppression (NMS) and class-aware **IoU** predictors followed by precise RoI pooling for bounding box refinement.

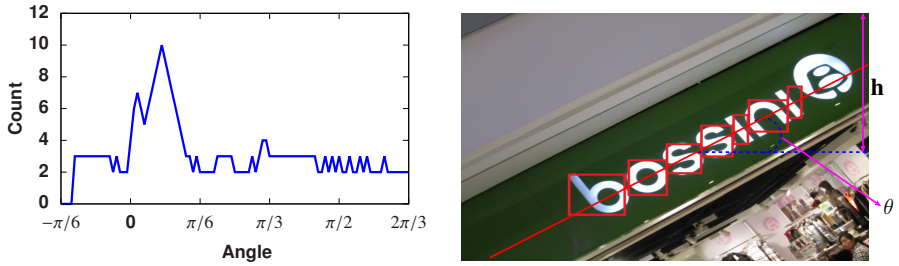
The rotated bounding box of a region proposal $\mathbf{r}$ is defined by five tuples $\mathbf{u} = (x, y, h, w, \theta)$. The geometric center is represented by the coordinate (x, y). The shorter side (height) and the long side (width) are represented by h and w, respectively. We define θ as the orientation angle between the positive direction of the $x-$axis to the

long side of the rotated bounding box in a parallel direction. Rotated anchors make use of aspect-ratio, orientation, and scale parameters. We change the aspect-ratio to {1:2, 1:5, 1:8} for considering a wide range of texts. We kept the scales as 8, 16, and 32.

Orientation Computation. Let us assume that text lines in their neighbourhood have almost similar spatial pattern and the orientation of characters in a text line are closely related. Motivated by skew computation algorithms for projection profile [137], we propose a mechanism for projection for computing orientation of text instances, as shown in Figure 6.8. We further assume that the orientation angle of a text line is φ in a text block. The offset for vertical-coordinate is ρ and draw out a straight line in the text block over the centroid of each character, as depicted in Figure 6.8. We compute the value of counting components $\Omega(\varphi, \rho)$ equals the number of the character components intersect the straight line. We finally compute the feasible orientation φ_i by counting the peak value of the components in the right direction as follows:

$$\varphi_i = \mathbf{arg}\ \underset{\varphi}{\mathbf{max}}\ \underset{\rho}{\mathbf{max}}\ \Omega(\varphi_i, h). \tag{5.6}$$

Assume that we have obtained six orientations, *i.e.*, $\{-\pi/6, 0, \pi/6, \pi/3, \pi/2, 2\pi/3\}$ to maintaining a trade-off between computational efficiency and orientation coverage. This able us to obtain 270 anchors for the proposed R-CNN subnet.



(a) (b)

Figure 5.6: (a) The line chart about the counting number of components in different orientations. (b) The red line correspond to center of the line chart.

Bounding-box Refinement. Any change in input distribution may affect the rotated bounding box regression iteration and consequently may lead to non-monotonic

localization improvement. To handle this problem, the optimization-based bounding box refinement technique [136] is included for robust localization accuracy estimation and as an early-stop indicator with adaptive steps. The problem of bounding box refinement is reformulated to obtain optimal $\mathbf{a}$, such that,

$$\mathbf{a} = \mathbf{arg}\ \min_{\Theta} \{-\mathbf{ln}(\mathbf{IoU})(\Psi(\mathbf{u}, \Theta), \mathbf{u}_{gt})\}, \tag{5.7}$$

where $\mathbf{u}_{gt}$ is the corresponding ground-truth bounding box for region proposal $\mathbf{r}$. Precise RoI Pooling layer ($PrPool$) [136] computes the gradient of **IoU** with reference to $\mathbf{u}$ to find an optimal solution. It has a continuous gradient preventing any quantization of bounding box coordinates. We perform average pooling by computing a two-order integral:

$$PrPool(bin, L) = \frac{\int_{x_1}^{x_2}\int_{y_1}^{y_2} \mathbf{ß}(\widehat{x}, \widehat{y})\ d\widehat{x}d\widehat{y}}{(x_2 - x_1) \times (y_2 - y_1)}, \tag{5.8}$$

where we consider $\mathbf{ß}$ to be continuous at any $(\widehat{x}, \widehat{y})$ using bilinear interpolation and $(\widehat{x}, \widehat{y})$ is continuous coordinates. The feature map $\mathbf{ß}$ is given by:

$$\mathbf{ß}(\widehat{x}, \widehat{y}) = \sum_{i,j} \widehat{\mathbf{w}}_{i,j} \times \mathbf{max}(0, 1 - |\widehat{x} - i|) \times \mathbf{max}(0, 1 - |\widehat{y} - j|). \tag{5.9}$$

The bounding box coordinates are iteratively fine-tuned by the obtained gradient. The predicted **IoU** between the detected bounding box and the corresponding matched ground-truth is maximized, where the predicted **IoU** indicates the localization confidence of each bounding box.

The size and rotation of each detected region may vary and therefore it is essential to map the features into canonical dimensions. We incorporate bilinear sampling of [94] that can map a region $\widehat{\mathbf{o}} \in \mathbb{R}^{h' \times w' \times \varsigma}$ into a tensor $\mathbf{o} \in \mathbb{R}^{h \times w \times \varsigma}$ of fixed-height, as follows:

$$\mathbf{o}_{x,y} = \sum_{x'=1}^{h} \sum_{y'=1}^{w} \widehat{\mathbf{o}}_{x',y'} \Gamma(x' - \Upsilon_{x'}(x)) \Gamma(y' - \Upsilon_{y'}(y)), \tag{5.10}$$

where Γ is the bilinear sampling kernel such that $\Gamma(\phi) = \mathbf{max}(0, 1 - |\phi|)$ and Υ is a point-wise coordinate transformation, which projects coordinates $\{x', y'\}$ of the fixed-sized tensor $\widehat{\mathbf{o}}$ to the coordinates $\{x, y\}$ in the detected region $\mathbf{o}$. The number of

channels remains unchanged. It permits shift and scaling along $x-$ and $y-$axes, whereas rotational parameters are straightaway considered from the region parameters. This helps to normalize rotation and scale persisting aspect-ratio and character position. This is important for persisting accuracy in text spotting task.

5.1.4 Miss-classification Problem

The inter-class distance and intra-class density can play an important role in measuring the effectiveness of a network. We therefore address the issue of reducing inter-class interference by investigating inter-class distance and intra-class density of features learned by deep networks. Intra-class density indicates the proximity of the features within the same class and inter-class distance denotes how distinct the features of different classes are. We assume that the features of a class is more likely to have a Gaussian distribution and therefore compute gaussian softmax of [117], known as $\mathcal{G}-$softmax, for classification on $\boldsymbol{\mathcal{D}}$, such that,

$$\mathbb{P}(\boldsymbol{\mathcal{D}}_i) = \frac{\mathbf{exp}(\boldsymbol{\mathcal{C}} \times \Delta(\boldsymbol{\mathcal{D}}_i, \mu_i, \sigma_i) + \boldsymbol{\mathcal{D}}_i)}{\sum_{j=1} \mathbf{exp}(\boldsymbol{\mathcal{C}} \times \Delta(\boldsymbol{\mathcal{D}}_j, \mu_j, \sigma_j) + \boldsymbol{\mathcal{D}}_j)}, \tag{5.11}$$

where Δ is the cumulative density function (CDF) of a Gaussian distribution, $\boldsymbol{\mathcal{D}}_i$ is the $i-th$ element of $\boldsymbol{\mathcal{D}}$, and μ is the mean and σ represents the standard deviation. If the value of $\boldsymbol{\mathcal{C}} = 0$, then we will get the conventional softmax function. The use of $\mathcal{G}-$softmax helps to approximate many distributions for every class during training, whereas the traditional softmax function learns from the current observation only. Furthermore, it directly quantifies inter-class distance and intra-class density from μ and σ. This implies that the improvement of over inter-class interference improves of mean accuracy, even in the presence of faint edges and cluttered background.

5.1.5 Recognition Module

The aim of a text recognition module is to process an image and recognize the word. We incorporate a location-aware embedding followed by a self-attention mechanism for

recognition of text instances, as shown in Fig. 5.7. We first apply the location-aware embedding mechanism, alike recognition module in Chapter 3, on the original input feature tensor (region proposal) to obtain the position embedded feature tensor E of shape $(G, U, G+U)$, where $\{G, U\}$ is set to $\{16, 32\}$. The location-aware embedding feature tensor E is calculated as $E'(i,j,:) = \textbf{onehot}(i, U)$, $E''(i,j,:) = \textbf{onehot}(j, G)$, and $E = \textbf{concat}(E', E'')$, where $\textbf{onehot}(\imath, k)$ indicates a $k-$length vector containing $\imath-$index element with unit value and rest as zero. We aggregate the location-aware embedding feature tensor with the original tensor to obtain aggregate tensor F is of size $(G, U, \zeta + G + U)$, where ζ is the number of channels of the input feature tensor which is set to 128.

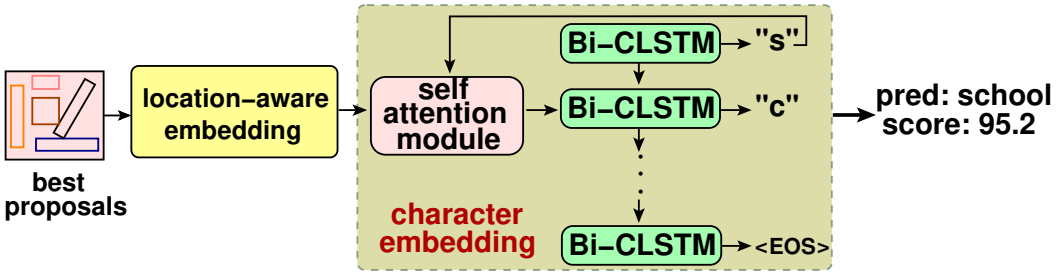


Figure 5.7: Architecture of the proposed recognition module.

Next, we predict a sequence of character classes $\mathbf{y} = (\mathbf{y}_1, \cdots, \mathbf{y}_t, \cdots, \mathbf{y}_T)$ iteratively for T steps. At a step t, the feature tensor is F, the last hidden state is $\mathbf{G}_{t-1}$, and the last predicted character class is $\mathbf{y}_{t-1}$. Unlike [118], we have used Bi-CLSTM [119] to keep the network light-weight. We expand $\mathbf{G}_{t-1}$ from a vector to a feature tensor $\mathbf{h}_{t-1}$ by copying, where $\mathbf{h}_{t-1}$ is of shape (G, U, V) and V is the hidden size of the recurrent neural network that is set to 128. LSTM accepts an input sequence $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_t, \cdots, \mathbf{x}_T)$, where each of $\mathbf{x}_t$ is a vector corresponding to step t. The output sequence $\mathbf{y}$ is computed

by using the following LSTM equations iteratively from $t = \{1, \cdots, T\}$:

$$\mathbf{i}_t = \sigma(\mathfrak{W}_{ix}\mathbf{x}_t + \mathfrak{W}_{iy}\mathbf{y}_{t-1} + \mathfrak{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i), \tag{5.12}$$

$$\mathbf{f}_t = \sigma(\mathfrak{W}_{fx}\mathbf{x}_t + \mathfrak{W}_{fy}\mathbf{y}_{t-1} + \mathfrak{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f),$$

$$\mathbf{m}_t = \sigma(\mathfrak{W}_{mx}\mathbf{x}_t + \mathfrak{W}_{my}\mathbf{y}_{t-1} + \mathbf{b}_m), \tag{5.13}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{m}_t \odot \mathbf{i}_t, \tag{5.14}$$

$$\mathbf{o}_t = \sigma(\mathfrak{W}_{ox}\mathbf{x}_t + \mathfrak{W}_{oy}\mathbf{y}_{t-1} + \mathfrak{W}_{oc}\mathbf{c}_t + \mathbf{b}_o), \tag{5.15}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{tanh}(\mathbf{c}_t), \tag{5.16}$$

$$\mathbf{y}_t = \mathfrak{W}_{yh}\mathbf{h}_t, \tag{5.17}$$

where symbols $\mathbf{i}$, $\mathbf{m}$, $\mathbf{o}$, $\mathbf{f}$, $\mathbf{c}$, $\mathbf{h}$, and $\mathbf{y}$ are respectively input gate, input modulation gate, output gate, forget gate, cell state, cell output, and projected output. The $\odot$ operator represents the element-wise product. The weight matrices are denoted by $\mathfrak{W}$ terms. The $\mathbf{b}$ terms indicate bias vectors. Bi-CLSTM uses a square matrix, known as circulant matrix, where each row (or column) vector is the circulant reformat of the row (or column) vectors. We assume that any matrix can be transformed into a set of circulant matrix blocks to obtain a block-circulant matrix. We compute the attention weight α_t as follows:

$$\alpha_t(i,j) = \frac{\mathbf{exp}(\xi_t(i,j))}{\sum_{i'=1}^{G}\sum_{j'=1}^{U}\mathbf{exp}(\xi_t(i',j'))}, \tag{5.18}$$

$$\xi_t = \mathfrak{W}_t \times \mathbf{tanh}(\mathfrak{W}_s \times \mathbf{H}_{t-1} + \mathfrak{W}_f \times F + \mathbf{b}_\varepsilon), \tag{5.19}$$

where ξ_t and α_t are of shape (G, U). $\mathfrak{W}_t$, $\mathfrak{W}_s$, $\mathfrak{W}_f$, and $\mathbf{b}_\varepsilon$ are trainable weights, and bias, respectively. We then update the glimpse $\mathbf{g}_t$ of step t by imposing the attention weight to the original feature tensor F as follows:

$$\mathbf{g}_t = \sum_{i=1}^{G} \sum_{j=1}^{U} \alpha_t(i,j) \times F(i,j). \tag{5.20}$$

We cascade the glimpse $\mathbf{g}_t$ with the input $\mathbf{x}_t$ of Bi-CLSTM and a character embedding class $\mathbf{y}_{t-1}$ of the last predicted character as follows:

$$\mathbf{\Phi}(\mathbf{y}_{t-1}) = \mathfrak{W}_y \times \mathbf{onehot}(\mathbf{y}_{t-1}, \mathbf{k}) + \mathbf{b}_y, \tag{5.21}$$

$$\mathbf{x}_t = \mathbf{concat}(\mathbf{g}_t, \mathbf{\Phi}(\mathbf{y}_{t-1})), \tag{5.22}$$

where $\mathbf{b}$ and $\mathfrak{W}$ are bias and trainable weights of the linear transformation, respectively. In the sequence decoder, $\mathbf{k}$ is the number of classes that includes 36 classes for alphanumeric characters in English (Latin) and 1 classes for end-of-symbol (EOS) symbol. Then, the conditional probability ($\mathbf{CP}$) at step t is obtained by a linear transformation and a softmax function as follows:

$$\mathbf{CP}(\mathbf{y}_t) = \mathbf{softmax}(\mathfrak{W}_o \times \mathbf{x}_t + \mathbf{b}_o) \tag{5.23}$$

$$\text{and } \mathbf{y}_t \sim \mathbf{CP}(\mathbf{y}_t). \tag{5.24}$$

Greedy decoding of network output is utilized during all the experiments. However, our recognition network is generalized, language independent, and open-dictionary. Furthermore, in order to include real-world examples of rotated and vertical text instances, we preserve the direction of reading at the word-level. Finally, we perform script identification using a majority voting over the predicted characters $\mathbf{y}_t$.

5.2 Experimental Results

To validate the effectiveness of Fainted TextSpotter, we conduct exhaustive experiments for scene text detection and end-to-end word spotting. We have used a pretrained model on SynthText [8] dataset for three epochs with weights for both detection and recognition. The weights are initialized from ImageNet [9] for detection and randomly con-

sidered from $\mathcal{N}(0, 1)$ distribution for recognition. We also adopt three training/testing splits for evaluation. We perform a comprehensive five of experiments on six publicly available benchmark datasets. We consider precision, recall, and f-measure as the metrics for evaluating the accuracy of detection. We report the recognition accuracy results in terms of strong, weak, and generic lexicons. Runtime complexity is measured by the number of flop counts, training parameters, and frames-per-second.

5.2.1 Implementation Details

The overall training process contains two stages, *i.e.*, pre-trained on SynthText dataset and then fine-tuned on the benchmark datasets on which it is to be tested. We set batch sizes of RPN [123] is taken as 256 while Fast R-CNN as 512 per image. Our network is implemented with Intel E5-2670v3 CPU running at 2.30 GHz and NVIDIA Titan X graphic card.

•**Training.** We simultaneously train our detection and recognition models for three epochs on a combined training dataset consisting of ICDAR 2015, MSRA-TD500, RCTW 2017, COCO-Text, and SVT. We randomly crop up to 30% of its width and height of an input image. The training in end-to-end fashion utilizes curriculum learning [120] strategy to train the model gradually to more complex data. We randomly first select 600k images from 800k synthetic images for 120k iterations with a learning rate 10^{-3}, where recognition task is trained by fixing the detection branch. Then, another 80k iterations are utilized for detection only with a learning rate is set to 10^{-4}. In the next 20k iterations, sampling tensors are obtained from detection results and the model is trained end-to-end in this stage. Finally, about 30k real-world images from six benchmark datasets are used in the next 70k iterations that enhance the generalization ability using data augmentation [15, 122]. We randomly rotate the input images in a certain angle range of $[-30°, 30°]$ for data augmentation. To incorporate character-level supervision, we set the batch size to 4 with learning rate 10^{-4}.

We use standard stochastic gradient descent with adam optimiser having a weight decay of 0.001 and a momentum of 0.9. The input images are fed in mini batches of 8 images. In the presence of fainted edges in a scene image, some background textures are very similar to text instance. It is therefore difficult for a network to distinguish. This makes training unbalanced, leading to slow convergence. We use hard a negative mining strategy to suppress them. The training on a dataset is performed into two stages. For the first stage of training, the negative ratio between the negatives and positives is set to 3:1. It is changed to 6:1 in the second stage of the training. To make our network robust, we choose a multi-scale training scheme [121]. Furthermore, we use python script to impose synthetic fog and rain on the training images of benchmark datasets and fed additionally in the network. This is because the images with ambient noises are very less.

Interference. Inference stage evaluates all datasets within a single model, where the scales of the input images depend on the datasets. In case of evaluation of cropped word recognition accuracy of the Fainted TextSpotter, words smaller than two characters are ignored. For each word, a set of hypothesis is formed adding to the ground-truth text a small number of lexicons. In our experimentation, we apply RPN to obtain predefined 300 text proposals in a forward pass followed by detection and recognition tasks. We incorporate *strong*, *weak*, and *generic* dictionaries for testing purpose [96]. The strong lexicon assigns 100 words per-image including all words that appear in the image. In the weak lexicon, all words present in the complete test set of the dataset are considered. The generic dataset consists of 90k words. It is to be noted that the words of length greater than three in dictionaries are kept and excludes signs and numbers. End-to-end and word-spotting models are preferably used for evaluation. In an end-to-end model, all words are recognized accurately, even though a detected string is not present in the dictionary. The word-spotting model, in turn, investigate only about the existence of a word of the dictionary in the images.

5.2.2 Ablation Study

In this section, we explore abundant ablation studies to evaluate detection and recognition accuracy and runtime complexity of the proposed network. We perform extensive experiments to investigate different aspects of our network.

• **Impact of backbone network.** We conduct a comprehensive set of experiments to select a backbone network. We study the impact of MobileNetV2, MobileNetV2+ASPP, MobileNetV2+ Semantic Edge Learning (Ours), SSDLite, ShuffleNetV2 [124], and IGCV2 [125] as a backbone network on the overall performance of the proposed network. Table 5.1 depicts that our backbone network outperforms other networks in terms of accuracy with an optimal number of training parameters. The network is not kept much deeper to restrict the number of training parameters.

Table 5.1: Effect of different variations of MobileNetV2, ShuffleNetV2 and IGCV2 as backbone networks on ICDAR 2015 dataset.

Backbone	**Fm.**	**Flops** (G)	**Params** (M)	**MAdds**
MobileNetV2 [115]	88.5	0.9	3.5	2.9
MobileNetV2+ASPP [115]	89.7	1.1	5.3	5.9
MobileNetV2+Semantic Edge Learning (Ours)	**90.1**	1.2	5.9	6.1
SSDLite [115]	89.1	**0.8**	**2.5**	**1.4**
ShuffleNetV2 [124]	88.7	2.8	10.9	8.7
IGCV2 [125]	87.6	4.6	23.8	10.7

• **Impact of context encoding module.** In the Fainted TextSpotter, the context encoding module has a spatial context branch and a channel dependency branch. We evaluate the impact of each branch on the overall performance of the proposed network, as shown in Table 5.2. We make ablation studies, where we consider our network as the baseline and create three more models. We include the spatial context branch of context encoding module and name the rest as *SC-TextSpotter*. Only channel dependency branch of the context encoding module is used in this model, which is denoted by *CD-TextSpotter*. The last one excludes both the branches of context encoding. It provides

a means to eradicate the contribution of context encoding branches to the performance of the overall network. This branch is known as *BN-TextSpotter*. It is clear from the results that context encoding module has a significant role in Fainted TextSpotter.

Table 5.2: Effect of context encoding on COCO-Text and SVT datasets.

Model	**COCO-Text**			**SVT**		
	Precision	**Recall**	**F-measure**	**Pr.**	**Re.**	**Fm.**
SC- TextSpotter	73.2	68.5	70.8	84.7	74	78.8
CD- TextSpotter	72.9	67.8	70.3	86.1	75.9	79.6
BN- TextSpotter	70.6	65.2	67.9	83.9	73.1	76.4
Ours	**74.9**	**68.7**	**70.5**	**87.1**	**76.3**	**80.7**

• **Impact of detection module.** We conduct experiments for evaluation of detection results using softmax (vanilla) [138], Large-Margin Softmax (L-softmax) [139] and $\mathcal{G}-$softmax [117] functions. Table 5.3 demonstrates that $\mathcal{G}-$softmax function outperforms the softmax and L-softmax functions on COCO-Text dataset. $\mathcal{G}-$softmax quantifies the density and distance of features. It is evident that with the improvement in intra-class density and inter-class distance, the average f-measure of the detection network also increases.

Table 5.3: Effect of different softmax functions on COCO-Text Dataset.

Function	**Precision**	**Recall**	**F-measure**
Softmax	69.8	61.2	65.5
L-softmax	71.7	60.1	66.3
$\mathcal{G}-$softmax ($\sigma = 0.5$, $\mu = 0$)	73.8	67.2	68.7
$\mathcal{G}-$softmax ($\sigma = 1$, $\mu = 0$)	73.3	66.5	69.9
$\mathcal{G}-$softmax ($\sigma = 5$, $\mu = 0$)	72.3	67.6	68.8
$\mathcal{G}-$softmax ($\sigma = 1$, $\mu =$ -0.1)	74.9	**68.7**	**70.5**
$\mathcal{G}-$softmax ($\sigma = 1$, $\mu = 0.1$)	**75.9**	66.9	68.7
$\mathcal{G}-$softmax ($\sigma = 1$, $\mu = 0.05$)	75.5	66.6	68.1

• **Impact of different devices.** We implement the proposed text spotter on seven different resource-constraint devices, as shown in Fig. 5.8. The technical specifications, such as processing speed and memory, are given in Table 5.4. We measure the power consumption of aforesaid smart devices using a Monsoon power monitor. It is evident

form the Figure 5.8 illustrate that the proposed Fainted TextSpotter is compatible with smart devices.

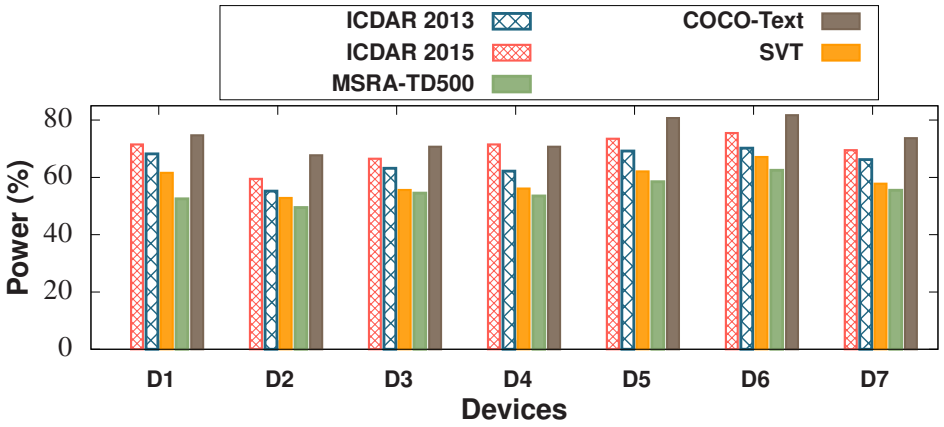


Figure 5.8: Effect of devices on power consumption for different datasets.

Table 5.4: Specifications of the smartphones with Adreno-640 GPU that are used for experimentation.

	Smartphone	**Operating System**	**Internal Memory**	**RAM**
D1	Samsung Galaxy S10+	Android 9	1 TB	12 GB
D2	Asus ROG Phone II	Android 9	1 TB	12 GB
D3	Xiaomi Mi 9 Pro 5G	Android 10	512 GB	12 GB
D4	Oneplus 7 Pro	Android 9	256 GB	12 GB
D5	Google Pixel XL4	Android 10	128 GB	6 GB
D6	LG G8X ThinQ	Android 9	1 TB	6 GB
D7	Sony Xperia 5 Plus	Android 10	1 TB	6 GB

• **Effect of different branches of recognition module on COCO-Text and NAST dataset.** Table 5.5 elaborates that each branches of the recognition module is important and plays a vital role in classification of text instances. Bi-LSTM is an essential part for precise sequential modeling.

• **Impact of size of RoI on detection module for ICDAR 2013 and NAST dataset.** We evaluate the influence of aspect-ratios and scales of the ROIs for the detection of scene text instances. Table 5.6 and Table 5.7 depict that the choice of aspect-ratio and scales in our network shows better f-measure for the detection of text

Table 5.5: Effect of different modules of recognition branch over COCO-Text and NAST dataset.

Location-aware Embedding	Bi-CLSTM	Self-Attention	Word recognition	
			COCO-Text	NAST
✗	✓	✗	62.5	32.5
✗	✓	✓	71.3	40.3
✓	✓	✗	68.4	39.2
✓	✓	✓	82.7	45.7

instances. This is because we capture both at word level and text-line level. Therefore, detected text instances are small but long in nature. Also, the use of spatial and channel-wise attention minimize the false positive in the recall.

Table 5.6: Effect of variation in size of RoI in detection on ICDAR 2013 [3] and NAST dataset.

Aspect-ratio	ICDAR 2013			NAST		
	Precision	Recall	F-measure	Precision	Recall	F-measure
1:2, 1:3, 1:5	89.4	88.6	89.1	52.4	51.2	51.8
1:3, 1:6, 1:9	91.8	91.3	91.5	51.3	50.6	50.9
1:2, 1:6, 1:9	87.8	88.2	88.3	53.6	52.4	53.2
1:3, 1:5, 1:7	89.3	88.6	88.6	58.4	56.9	57.6
1:2, 1:5, 1:8	**94.9**	**95.7**	**94.9**	**59.3**	60.4	**59.8**

Table 5.7: Effect of variation in scale of RoI in detection on ICDAR 2013 [3] and NAST dataset.

Scale	ICDAR 2013			NAST		
	Precision	Recall	F-measure	Precision	Recall	F-measure
4, 8, 16	91.4	90.6	91.1	52.2	50.3	51.2
8, 16, 24	92.3	91.6	91.6	56.6	55.8	56.2
8, 16, 32	**94.9**	**95.7**	**94.9**	**59.3**	**60.4**	**58.9**
16, 24, 32	94.1	94.5	94.3	57.4	57.2	57.3
16, 32, 64	92.5	90.8	91.6	55.1	53.3	54.2

• **Impact of size of RoIs in Recognition module.** The aspect-ratios, scales, and number of channels of the RoIs affect the recognition process. With the increase in the number of channels, the recognition accuracy increases; however, it also increases the

computational overhead. Therefore, we maintain an accuracy-cost trade-off, as shown in Table 5.8. The choice of aspect-ratios and scales taken in the proposed network helps in precise detection, which enhances the recognition efficiency, as analyzed in Table 5.9 and Table 5.10.

Table 5.8: Effect of variation in the number of channel in text recognition on ICDAR 2015 [2] dataset.

Number of channel	Params (M)	End-to-End			Word-Spotting		
		strong	weak	generic	strong	weak	generic
16	2.83	82.6	79.3	75.7	84.8	81.1	78.2
32	4.54	85.6	82.8	78.3	88.6	85.9	81.4
128	**6.21**	**91.5**	**89.6**	**84.9**	**94.2**	**92.6**	**88.2**
256	9.67	91.6	88.3	84.6	94.6	91.3	86.3
512	12.89	91.8	89.8	85.2	95.1	92.4	89.7

Table 5.9: Effect of variation in size of RoI in text spotting on ICDAR 2015 [2] dataset.

Aspect-ratio	End-to-End			Word-Spotting		
	strong	weak	generic	strong	weak	generic
1:2, 1:3, 1:5	85.8	82.9	79.7	89.7	86.4	82.2
1:3, 1:6, 1:9	88.5	86.7	83.5	92.8	87.8	84.3
1:2, 1:6, 1:9	86.1	82.4	78.5	90.4	87.2	85.9
1:3, 1:5, 1:7	87.4	85.3	82.4	93.6	91.1	86.3
1:2, 1:5, 1:8	**91.5**	**89.6**	**84.9**	**94.2**	**92.6**	**88.2**

Table 5.10: Effect of variation in scale of RoI in text spotting on ICDAR 2015 [2] dataset.

Scale	End-to-End			Word-Spotting		
	strong	weak	generic	strong	weak	generic
4, 8, 16	85.6	83.4	77.5	89.5	85.6	81.5
8, 16, 24	88.2	87.2	82.4	92.7	89.8	86.4
8, 16, 32	**91.5**	**89.6**	**84.9**	**94.2**	**92.6**	**88.2**
16, 24, 32	90.6	88.7	86.1	93.3	91.8	87.3
16, 32, 64	88.2	85.1	81.8	91.7	89.4	85.7

5.2.3 Comparison with State-of-the-Art Results

In this section, we compare our network with the state-of-the-art approaches [1,8,12,15, 19, 44, 46, 48, 50, 53, 68, 92, 93, 95–97, 99, 102, 104, 130, 132, 133, 140–145] on five different publicly available benchmark datasets. We also perform a comparative study of the both parameter count and computational cost of our network with the existing models.

• **Detection results on different datasets.** Fainted TextSpotter is compared with recent approaches in terms of precision, recall, and f-measure. Tables 5.11, 5.12, and 5.13 depict that Fainted TextSpotter outperforms the existing literature on all ICDAR 3013, ICDAR 2015 and MSRA-TD500 datasets in terms of f-measure. Our network perform better in the presence of noises, as shown in Table 5.14 and TableTable 5.15 where images in COCO-Text and SVT datasets have blurring artifacts and perceptual distortion, respectively.

Table 5.11: Performance comparison on ICDAR 2013 [3] dataset for text detection in scene images.

Methods	**ICDAR 2013**		
	Precision	**Recall**	**F-measure**
SegLink [17]	87.7	83	85.3
WordSup [37]	93.3	87.5	90.3
LATD [34]	91	77	83
Raghunandan *et al.* [42]	88.4	66.4	75.8
Tang & Wu [47]	91.1	86.1	88.5
Lyu *et al.* [35]	93.3	79.4	85.8
FOTS [1]	-	-	88.2
He *et. al.* [96]	91	88	90
TextBoxes++ [98]	86	74	80
Mask TextSpotter [104]	**94.8**	89.5	92.1
ASTS (baseline) [107]	-	-	91.7
ASTS (weakly) [107]	-	-	93.5
Text Perceptron (2-stage) [112]	92.7	88.7	90.7
Text Perceptron (end-to-end) [112]	94.7	88.9	91.7
TextNet [128]	93.2	89.3	91.2
Boundary [110]	93.1	87.3	90.1
RRD [36]	88	75	81
Ours	94.9	**95.7**	**94.9**

Table 5.12: Performance comparison on ICDAR 2015 [2] dataset for text detection in scene images.

Methods	**ICDAR 2015**		
	Precision	**Recall**	**F-measure**
EAST [15]	83.3	78.3	80.7
SegLink [17]	73.1	76.8	75.0
WordSup [37]	79.3	77	78.1
LATD [34]	87.8	78.1	82.7
Li *et. al.* [99]	71.6	93.4	81.1
Lyu *et al.* [35]	**94.1**	70.7	80.7
FOTS [1]	91	85.1	87.9
Li *et. al.* [95]	91.4	80.5	85.6
He *et. al.* [96]	87	86	87.0
TextBoxes++ [98]	87.2	76.7	81.7
TextDragon [105]	92.4	83.7	87.8
Mask TextSpotter [104]	86.6	87.3	87.0
ASTS (baseline) [107]	-	-	87.8
ASTS (weakly) [107]	-	-	89.9
Text Perceptron (2-stage) [112]	91.6	81.8	86.4
Text Perceptron (end-to-end) [112]	92.3	82.5	87.1
TextNet [128]	89.4	85.4	87.3
Boundary [110]	89.8	87.5	88.6
RRD [36]	85.6	79	82.2
Ours	91.1	**89.8**	**90.7**

• **Recognition results on different datasets.** Fainted TextSpotter is compared with state-of-the-art literature for word spotting and end-to-end text recognition. Table 5.16 demonstrates that our network is improved by more than 5% for all 3 lexicons on ICDAR 2015 dataset with respect to FOTS [1] (baseline). The proposed network achieves state-of-the-art results for word spotting on SVT dataset, as shown in Table 5.17. We obtain comparable results for COCO-Text dataset in terms of text recognition accuracy.

• **Speed and Model Size.** The use of light-weight backbone and region proposal networks drastically reduces both parameter count and computational cost. Table 5.18 demonstrates the test time speed of the proposed Fainted TextSpotter network when compared with state-of-the-art scene text detection methods. We have reported the flops, number of training parameters (params), and frames-per-second (fps) of our network to evaluate the running time complexity.

Table 5.13: Performance comparison on MSRA-TD500 dataset.

Methods	**Precision**	**Recall**	**F-measure**
GISCA [30]	86.3	77.1	81.4
East [15]	87.2	67.4	76
RefineText [21]	83.2	80.2	81.7
OPMP [56]	86	83.4	84.7
Mask-Most Net [52]	85.5	74.1	79.4
Yao *et. al.* [57]	76.5	75.3	75.9
Lyu *et. al.* [35]	87.6	76.2	81.5
He *et. al.* [16]	77	70	74
RRPN [23]	82	69	75
Raghunandan *et. al.* [42]	67.2	77.2	72.4
Dey *et. al.* [43]	52	85	65
Khare *et. al.* [44]	45	53.3	48.8
TextField [49]	87.4	75.9	81.3
Mask TTD [54]	85.7	81.1	83.3
Tian *et. al.* [24]	84.2	81.7	82.9
Seglink [17]	86.0	70.0	77.0
DSRN [65]	87.6	71.2	78.5
Ours	**87.9**	**86.2**	**86.6**

5.3 Summary

In this paper, an efficient network for arbitrary-oriented text spotting in scene images based on light-weight deep neural network is proposed. Our network detects text instances, signs, and markings with high accuracy and speed using hierarchical spatial context information and inter-channel dependencies of edge supervised feature map of the light-weight backbone network, which is enriched in spatial information. The proposed text spotter can localize and recognize multilingual text instances in scene images, which are captured in adverse weather conditions, in resource-constrained devices, like smart devices. We have reduced misclassification by addressing the problem of inter-class interference using Gaussian softmax. The proposed Fainted TextSpotter also provides text script class. We have evaluated our network with publicly available benchmark datasets. Our Fainted TextSpotter outperforms previous methods in terms of efficiency and performance.

Table 5.14: Performance comparison on COCO-Text dataset for detection of texts in scene images.

Methods	**COCO-Text**		
	Precision	**Recall**	**F-measure**
EAST [15]	36.4	29.5	32.9
SegLink [17]	28.7	39.6	34.1
MaskTextSpotter [118]	66.8	58.3	62.3
Raghunandan *et al.* [42]	55.3	58.1	56.7
LATD [34]	74	51	61
Cheng *et. al.* [45]	60	33	42
Cheng & Wang [130]	48	38	42
Yao *et al.* [57]	43.2	27.1	33.3
TextBoxes++ [98]	60.8	56.7	58.7
Lyu *et al.* [35]	61.9	32.4	42.5
RRD MS [36]	64	57	61
FCRNall + multi-filt [8]	23.6	25.2	24.7
He *et al.* [46]	64.5	48.7	56.6
Dey *et al.* [43]	50.1	62.3	56.2
Tang & Wu [48]	49.6	67.4	58.5
TextBoxes [97]	54.7	46.3	50.5
Khare *et al.* [44]	35.9	30.2	33
Ours	**74.9**	**68.7**	

Table 5.15: Performance comparison on SVT dataset for detection of texts in scene images.

Methods	**SVT**		
	Precision	**Recall**	**F-measure**
EAST [15]	50.3	32.4	39.4
SegLink [17]	30.2	42.4	35.7
MaskTextSpotter [118]	70.4	65.1	67.7
Raghunandan *et al.* [42]	60.4	68.7	64.2
LATD [34]	78.1	56.3	67.3
Cheng *et. al.* [45]	65.6	35.2	50.4
Cheng & Wang [130]	53.4	36.3	44.8
Yao *et al.* [57]	49.5	34.7	42.1
TextBoxes++ [98]	68.7	62.4	65.5
Lyu *et al.* [35]	66.2	37.9	52
RRD MS [36]	69.5	60.2	64.8
FCRNall + multi-filt [8]	26.2	26.7	27.4
He *et al.* [46]	87	73	79
Dey *et al.* [43]	55	68	61
Tang & Wu [48]	54.1	75.8	63.1
TextBoxes [97]	67.2	60.8	63.8
Khare *et al.* [44]	41.6	44.3	42.9
Ours	**87.1**	**76.3**	**80.7**

Table 5.16: Performance comparison on ICDAR 2015 dataset for word spotting and end-to-end recognition of texts in scene images.

Methods	**Word Spotting**			**End-to-End**		
	strong	**weak**	**generic**	**strong**	**weak**	**generic**
Neumann & Matas [131]	35	19.9	15.6	35	19.9	15.6
Deep TextSpotter [94]	58	53	51	54	51	47
Li *et. al.* [95]	**94.2**	92.4	**88.2**	91.1	**89.8**	84.6
TextBoxes [97]	93.9	91.9	85.9	**91.5**	89.6	83.8
Mask TextSpotter [118]	79.3	74.5	64.2	79.3	73	62.4
He *et. al.* [96]	85	80	65	82	77	63
FOTS [1]	84.6	79.3	63.2	81	75.9	60.8
FOTS MS [1]	88	83.3	68.9	84.5	80.1	66.3
TextBoxes++ [98]	76.5	69	54.4	73.3	65.9	51.9
CharNet [78]	85.4	77.6	69.8	85.1	81.3	71.1
TextDragon [105]	86.2	81.6	68	82.5	78.3	65.2
Qin *et. al.* [146]	87.3	80.5	71.6	85.5	81.9	69.9
Boundary [110]	82.5	77.8	65.3	79.7	75.2	64.1
Text Perceptron [112]	84.1	79.4	67.9	80.5	76.6	65.1
Li *et. al.* [99]	89.6	87.1	70	86	83.4	66.9
ASTS [107]	87.7	82.9	68.9	84.8	79.8	66.5
Ours	**94.2**	**92.6**	**88.2**	**91.5**	89.6	**84.9**

Table 5.17: Performance comparison on COCO-Text and SVT datasets for text recognition accuracy and word spotting in scene images.

Methods	**COCO-Text**	**SVT**	
	Recognition	**Word Spotting**	
	Accuracy	**SVT**	**SVT-50**
Baseline A [4]	**82.9**	63.7	77.2
Baseline B [4]	61	42.3	58.1
Baseline C [4]	23.4	32.4	46.7
Mask TextSpotter [118]	75.3	61.5	79.4
FCRNall + multi-filt [8]	66.4	55.7	67.7
TextBoxes [97]	62.1	64	72.4
Jaderberg *et al.* [93]	71.7	53	76
Li *et al.* [95]	69.2	66.2	**84.9**
TextBoxes++ [98]	66.9	64	84
Neumann and Matas [131]	61.6	56.2	68.1
Alsharif *et al.* [132]	49.2	39.5	48
Jaderberg *et al.* [92]	52.5	45.3	56
Wang *et al.* [133]	57.2	46	62.5
Ours	82.7	**67.2**	**84.9**

Table 5.18: Test time speed in terms of on FLOPS, number of training parameters, and frames-per-second (fps) on ICDAR 2015 dataset.

Methods	**FLOPs (G)**	**Params (M)**	**fps**	**D/R/S**
EAST [15]	4.685	24.1	13.2	D
FOTS [1]	9.997	34.9	7.8	S
E2E-MLT [102]	2.946	4.7	-	S
FCRN [8]	-	-	20	S
Jaderberg *et al.* [93]	-	-	2	S
Enhanced EAST [144]	-	-	7.9	D
Li *et al.* [99]	-	-	3.7	S
Mask TextSpotter [118]	-	-	3.1	S
Ours	0.973	5.9	**28.5**	S