Chapter 3

Cluttered TextSpotter

In this chapter, we present a deep network model to perform scene text detection, recognition, and spotting in natural scene images. Text in scene images are associated with other objects, making the detection of scene text challenging. Traditionally, text instances are smaller in nature in comparison to other objects associated with the scene image. Therefore, a deep network can learn few features relevant to text instances, which further increases the difficulty in the detection task. The presence of background clutters, partial occlusion, and truncation of text instances in scene images enhances complexity in the detection and recognition process. In real-time applications, scene images have a cluttered background, which results in partial occlusion, truncation artifacts, and inter-class interference, as shown in Figure 3.1. The presence of cluttered background noise restricts the dense feature matching process, which eventually leads to a misclassification problem. Although, if we humans have to detect a text instance in a cluttered scene image, no matter how complex the background clutters are, the localization of texts is subserved by a local task sensitive to position-aware features and a global task of retrieving structural context knowledge.

Contextual information is critical for feature representation. In a classification task, focusing on informative regions in images is beneficial to generate discriminative fea-

tures. However, the presence of cluttered background noise severely restricts the feature matching process in real application scenarios. Thus, it is advantageous to identify and focus on the informative regions and their structural context. In generic object detection using deep neural networks (in short deep networks), global structures and context information of an object are used to learn and discriminate salient features [114]. Therefore, it is important to overcome the background clutters and partial occlusion artifacts for accurate spotting of texts in cluttered scene images. Therefore, we extract the local part information and global contextual structural information from the extracted feature map to handle the presence of partial occlusion and truncation in scene images and apply Gaussian softmax for detection and an attention mechanism for text recognition. We consider FOTS [1] as our baseline literature for comparison.

(a) (b) (c)

Figure 3.1: Illustrating the necessity of Cluttered TextSpotter in scene text spotting. Columns (b) and (c) are the recognized text instances in the scene images of column (a) using baseline [1] and our network. Cluttered TextSpotter can spot oriented text instances in the scene images with a cluttered environment.

To the best of our knowledge, local and global structural context information of

text regions, especially in the presence of occluded text parts and truncation artifacts, has still not been fully utilized for text spotting in scene images. Furthermore, in most state-of-the-art literature, the authors mainly consider scenes images with good illuminating conditions. Also, text instances have good resolution and contrast with respect to other objects in the scene images. Also, partial occlusion or truncation artifacts are not considered.

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.

3.1 Proposed Architecture

In this section, we propose a deep network, known as *Cluttered TextSpotter (in short CTS)*, for text spotting in scene images that has higher efficiency and consume less time and memory in the presence of background clutters. The overall architecture of the proposed Cluttered TextSpotter is illustrated in Figure 3.2.

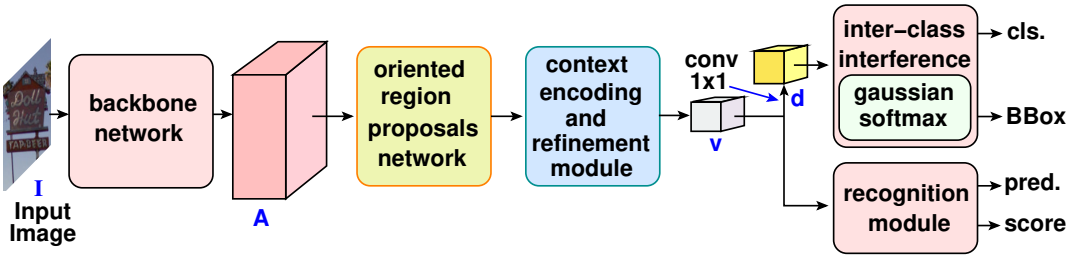

Figure 3.2: The architecture of the proposed Cluttered TextSpotter..

Out network has the following modules:

- First, we introduce MobileNetV2 in our backbone network to make it light-weight in nature. To encode multi-scale contextual information, we integrate a light-weight atrous spatial pyramid pooling with our network. We further include the encoder-decoder model to recover the finer spatial information. This helps to

exploit low-level features along with a high-level and also reduces the degradation problem.

- Second, we design an oriented region proposal network, which is light-head is nature, to obtain oriented region proposals. Bilinear sampling is used for mapping the rotated region proposals into the canonical dimension. It also normalizes the rotation and scaling keeping the aspect ratio and position of individual characters intact.
- Third, we develop a context encoding and refinement module for precisely regressing the oriented bounding boxes of text instances, even in the presence of partially occluded text parts, truncation artifacts, perspective distortion, and inter-class interference. It is being served by both a global process of retrieving structural context and a local process sensitive to regional features.
- Finally, we use Gaussian softmax to mitigate the misclassification problem due to inter-class interference. We also incorporate a light-weight recognition module using C-LSTM within Bi-LSTM for multi-language character-level segmentation and word-level recognition.

3.1.1 Backbone Network

This section aims to develop a backbone network that learns finer spatial information of text regions along with multi-scale contextual information. Our backbone network is light in nature by utilizing the residual bottleneck blocks of MobileNetV2 [115] as a depth separable convolution. It helps to extract the grafted feature map by encoding low-level features with high-level semantics using an encoder-decoder architecture. We assume that the size of the input image I in the backbone network is $2H \times 2W \times 3$. We initialize a convolution layer with a kernel of size 3×3 and 32 filters in our backbone network. We then append 37 residual bottleneck layers followed by another convolution layer with 1×1 kernel and 1280 filters. Each residual bottleneck layer has a kernel of

size 3×3, along with RELU6, dropout, and batch normalization. The architecture of the backbone network is shown in Figure 3.3.

We have taken an output stride of the backbone network as 16 to get a dense feature. We therefore change the stride in the third residual layer of MobileNetV2 from 2 to 1. With the increasing number of stride, the computation complexity decreases, but it leads to failure in dense feature matching. Text in natural scene images is usually small in nature, which requires dense features for precise detection. Thus, there is always a trade-off between running time and high recall. Next, we apply a layer of atrous spatial pyramid pooling (ASPP) [116] after the last layer. ASPP layer consist of four parallel convolutions with same kernel of size 3×3 but different dilation rate $\partial = \{1, 2, 3, 4\}$. Since we are not performing feature pooling in a deeper stage of the network, but feature computation is performed in the initial stage of our network. Therefore our sampling rates are much smaller in comparison with [116]. We incorporate atrous separable convolution, which encodes multi-scale contextual information. Our atrous convolution is divided into a depthwise convolution, followed by a point-wise convolution. This reduces computation complexity while enhancing the convolutional features at multiple scales with four different dilation rates. The output of the four atrous convolutions are concatenated using element-wise sum to obtain the output feature map $\mathcal{O} \in \mathbb{R}^{14 \times 14 \times 1280}$ of the encoder.

Lastly, we introduce the encoder-decoder architecture in our backbone network. The finer spatial information is recovered in this architecture. The integration of MobileNetV2 with ASPP helps to obtain the encoder of the backbone network. This helps to capture high-level semantic features. We further include the decoder to preserve rich spatial information in the backbone network. The decoder of the backbone network helps to exploit both low-level feature with high-level semantics, which helps to prevent from degradation problem. In the encoder, we bilinearly upsampled the spatial resolution of the output feature map $\mathcal{O}$ by a factor of 4. We apply 1×1 convolution on the

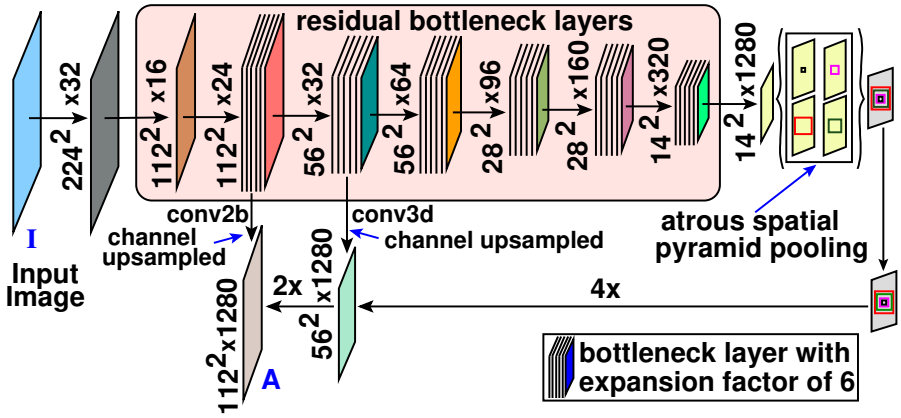

Figure 3.3: Architecture of the backbone network.

low-level features of *residual conv3b* to increase the number of channels to 1280 and concatenated with the corresponding upsampled features using element-wise sum. We then apply a 3×3 convolution to refine the features, which is followed by another bilinear upsampling by a factor of 2. This reduces the number of computations in comparison with one bilinearly upsampling $8\times$ directly. The upscaled feature map is concatenated with the channel reduced feature map of *residual conv2b* using element-wise addition. The output of *residual conv2b* is channel reduced by applying 1×1 convolution to increase the channel to 1280. The final output feature map of the backbone network is $A \in \mathbb{R}^{H \times W \times 1280}$, which is fed as input to the proposed R-CNN subnet.

3.1.2 Oriented Region Proposal Network

The goal of this section is to generated an oriented text region proposal network for detecting arbitrary oriented scene text. It is a subnet of our proposed network, which is light-weight in nature, as depicted in Fig. 3.4. We generate fixed-size feature maps by incorporate RoI warping. We produce thin feature maps (smaller number of channels) accompanied by traditional RoI warping. During the training and inference, we find that RoI warping on thin feature maps will save memory and computation, keeping the accuracy same. We apply position sensitive RoI (PSRoI) pooling of rotated region

proposals on thin feature maps and decrease the R-CNN overhead to improve our performance. The channels are reduced from 1280 to 128 to obtain thin feature maps, as shown in Fig. 3.4.

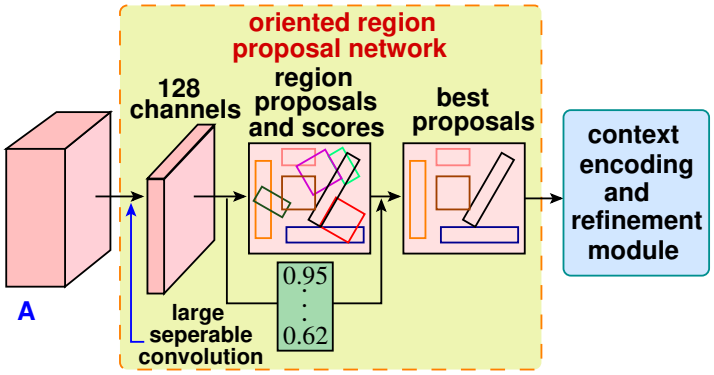

Figure 3.4: Architecture of the oriented region proposal network.

We utilize rotated anchors [23] to obtain rotated region proposals. Intersection-over-union (**IoU**) between ground truth and an anchor is the overlap over skew rectangles. A positive anchor that has an **IoU** > 0.7 with respect to the ground truth or the largest **IoU** overlap with an intersection angle $\mho < \pi/12$ respect to the ground truth. Similarly, the negative anchor has an **IoU** < 0.3 or **IoU** > 0.7 with $\mho > \pi/12$ with a ground truth. The regions which are not selected either by a positive anchor or a negative anchor are not utilized in training. We empirically found that it is reasonable to use **IoU** value as a trade-off between speed and accuracy.

The proposed region proposal $\mathbf{r}$ have bounding box of 5 tuples $\mathbf{u} = (x, y, h, w, \theta)$ and a confidence score $\mathbf{s}$. The center coordinate of a bounding box is denoted by (x, y). The height h and width w represents the small and long sides of the bounding box, respectively. The angle between the $x-$axis (positive direction) and in a direction parallel to the long side of the rotated bounding box is represented by θ. The rotated anchors use scale, aspect ratio, and orientation parameters. The value of $\theta = \{-\pi/6, 0, \pi/6, \pi/3, \pi/2, 2\pi/3\}$ helps to control the runtime complexity and orientation coverage. The aspect ratio has the values {1:2, 1:5, 1:8} to cover a broad range

of text-lines. We keep scales of text instances as 8, 16, and 32. Thus, we obtain 270 anchors [23] for the proposed region-based oriented subnet.

Each detected region has a different shape (aspect ratio and rotation). We therefore map the features of a region into a canonical dimension. We map a region $\widehat{\mathbf{r}} \in \mathbb{R}^{h' \times w' \times \varsigma}$ into a fixed height tensor of $\mathbf{r} \in \mathbb{R}^{h \times w \times \varsigma}$, we incorporate bilinear sampling of [94] and it is define as:

$$\mathbf{r}_{x,y} = \sum_{x'=1}^{h} \sum_{y'=1}^{w} \widehat{\mathbf{r}}_{x',y'} \Gamma(x' - \Upsilon_{x'}(x)) \Gamma(y' - \Upsilon_{y'}(y)), \tag{3.1}$$

where the kernel of bilinear sampling is $\Gamma(\vartheta) = \mathbf{max}(0, 1|\vartheta|)$. Υ is a point-wise coordinate transformation that maps the coordinates $\{x', y'\}$ of $\widehat{\mathbf{r}}$, fixed-size tensor, to the coordinates $\{x, y\}$ of the detected region $\mathbf{r}$, where we keep the number of channels unchanged. The transformation allows the shifting and scaling along the $x-$ and $y-$axes, and the rotational parameters are extracted directly from the region parameters. The standard RoI warping lacks in comparison to feature representation as it not only allows the network to normalize scale and rotation but also persists the aspect ratio and location of each character at the same time. This makes feature representation crucial for persisting accuracy in text spotting task.

3.1.3 Context Encoding and Refinement Module

In this section, we address the challenges caused by the truncation artifacts, partial occlusions, and background clutter. Therefore, we incorporate a local process sensitive to position-sensitive features and a global process of retrieving structural context followed by a context refinement task to address the problem of unreliable results in the text localization process, as shown in Figure 3.5.

Specific fine-grained parts information of text instances are captured by local part processing, which is important for the classification task. We utilize PSRoI pooling of rotated region proposals to exploit local part information of text instances. PSRoI pooling typically ignores global structure information. If local part information is only

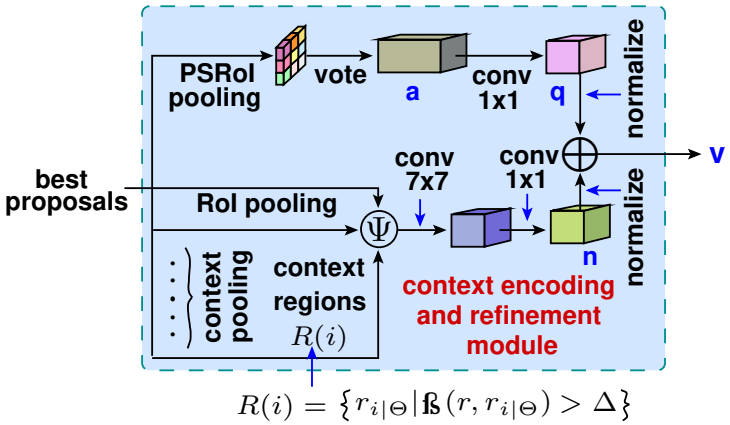

Figure 3.5: Architecture of the context encoding and refinement module.

used, a long word with partially occluded text parts will not be recovered as a single word. As a result, the long word will appear to be a combination of many small words. Global structure information is thus necessary for the detection of texts in scene images, especially in a cluttered environment. In the presence of global structure information, the network will have structural information of a large region. If global structure information is only provided, then the network will have the complete structural information of a large region, but no specific fine-grained part related information will be present, which may lead to misclassification. We therefore utilize both global and local features for detection. Furthermore, we include context information to enhance the representation of global features. Context information also plays an essential role in the classification task. For example, a boat is surrounded by water. The information about the surroundings (or background) for a particular object is the context for that object. Similarly, scene texts mostly occur in regions having a uniform texture and color, such as vehicle number plate, billboards, and door plates.

In the first branch, we used one fully-connected layer with 128 channels and no dropout for PSRoI pooling of each rotated proposal. It extracts text-specific parts followed by a voting task to obtain the resized tensor $\mathbf{q} \in \mathbb{R}^{32\times64\times128}$ using average pooling. It serves as the ensemble of multiple part models. In the second branch, we

introduce global context encoding using the RoI pooling of rotated region proposals. We utilize global context encoding to tackle the partial detection issue, considering that the background regions provide informative clues and auxiliary discriminators. It describes a text region with an accurate status. Due to the wide diversity in size of text instances, we introduce an RoI pooling layer on rotated region proposals to extract a feature vector with fixed-length as the global descriptor. We also obtain a set of context regions R for a given region proposal $\mathbf{r}$ with variations based on $\Theta = \{h, w, \theta\}$ of the original proposal $\mathbf{r}$, such that,

$$R(i) = \{\mathbf{r}_{i|\Theta} | L(\mathbf{r}, \mathbf{r}_{i|\Theta}) > \Delta\}, \tag{3.2}$$

where $\mathbf{r}_{i|\Theta}$ is $i-th$ context region of the original proposal $\mathbf{r}$ by increasing height h or width w by $\delta-$times larger than the size of $\mathbf{r}$ or varying the orientation θ by $\pounds\pi$ ensuring that $\theta + \pounds\pi$ is within the interval $[\frac{\pi}{4}, \frac{3\pi}{4}]$. In our implementation, we consider the threshold Δ as 0.5. We further use the **IoU** score of two regions as a measure of correlation level between them, such that,

$$L(\mathbf{r}, \mathbf{r}_{i|\Theta}) = \mathbf{IoU}(\mathbf{u}, \mathbf{u}_{i|\Theta}). \tag{3.3}$$

RoI pooled features from the original region and a set of context regions are then aggregated together. We adopt an adaptive weighting mechanism for aggregation. The visual features $\mathbf{v}$ extracted from the region $\mathbf{r}$ is bounded by $\mathbf{u}$ and its confidence score $\mathbf{s}$. We use $\mathbf{v}_{i|\Theta}$ and $\mathbf{w}_{i|\Theta}$ to denote the contextual information and the corresponding weight carried by $\mathbf{r}_{i|\Theta}$ for the original proposal $\mathbf{r}$. We implement the aggregation operation Ψ as follows:

$$\Psi(\mathbf{r}, \mathbf{v}_{i|\Theta}, \mathbf{w}_{i|\Theta}) = \frac{\sum_i (\mathbf{w}_{i|\Theta} \times \mathbf{v}_{i|\Theta})}{\sum_i \mathbf{w}_{i|\Theta}}, \tag{3.4}$$

where the value of $i = 0$ represents the original proposal $\mathbf{r}$. Two convolutional layers are incorporated, having a kernel of size 7×7 and 1×1 to further exploit the global representation of oriented RoI and obtain a resized tensor, denoted by $\mathbf{n} \in \mathbb{R}^{16\times32\times128}$. The accuracy of precision is improved by refining the process of regressing the bounding boxes.

We apply 1×1 convolution for normalization separately on local part information enriched tensor $\mathbf{q}$ and global context enhanced feature tensor $\mathbf{n}$ followed by element-wise sum to concatenate and obtain the output tensor $\mathbf{v}$ for each rotated region proposal. The tensor $\mathbf{v}$ is fed into 1×1 convolution layer and a classifier to produce the output vector $\mathbf{d}$, a two-dimensional vector that indicates the probability whether a region is a text instance or not.

3.1.4 Inter-class Interference Problem

The inter-class separability and intra-class compactness play a significant role in quantifying network efficiency. We, therefore, address the inter-class interference by exploring inter-class separability and intra-class compactness of features learned by deep networks using Gaussian softmax [117]. The closeness of features within the same class is denoted by intra-class compactness and how discriminative the features of different classes are from each other is represented by inter-class separability. We assume that the distribution of text-specific features with reference to the background is subject to Gaussian distribution; thus, the classification on $\mathbf{d}$ is performed as follows:

$$\mathfrak{P}(\mathbf{d}_i) = \frac{\mathbf{exp}(\lambda \times \Omega(\mathbf{d}_i, \mu_i, \sigma_i) + \mathbf{d}_i)}{\sum_{j=1} \mathbf{exp}(\lambda \times \Omega(\mathbf{d}_j, \mu_j, \sigma_j) + \mathbf{d}_j)}, \tag{3.5}$$

where μ and σ represent the mean and standard deviation of Gaussian distribution. Ω is the cumulative density function (CDF) and $\mathbf{d}_i$ is the $i - th$ element of $\mathbf{d}$. The use of a softmax function depends on the value of λ. When $\lambda = 0$, traditional softmax function is obtained. Gaussian softmax helps in approximating distributions of text-

specific features with large variations on the training samples. On the other hand, the softmax function learns only from the current observing sample. In addition, the use of distribution parameters μ and σ are used to quantify inter-class separability and intra-class compactness directly. It shows that the average accuracy of detection will be improved with the improvement in inter-class separability and intra-class compactness, even in a cluttered background.

3.1.5 Recognition Module

In this section, the label sequence of each text instance is predicted globally. Occlusion in scene images leads to false label prediction of each character in the word. Inspired by [118], we include a position embedding mechanism to convert each word into a real-value vector and to specify a relative position of each character in a word as a pair of entity. We use position embedding on the input feature tensor $\mathbf{v}$ to obtain the embedded feature tensor P, as shown in Fig. 3.6. Assume the feature tensor $\mathbf{v}$ has ν characters. The tensor $\mathbf{v}$ is divided by a fixed sliding window with the value $\mathfrak{m}$ and the step size of each slide is $\mathfrak{s}$ to obtain a hierarchical sequence $X = \{x_1, x_2, x_3, \cdots, x_\nu\}$. Each word has two annotated entities e_1 and e_2. The aim of relation classification is to identify the semantic relation between entities e_1 and e_2 in a given word. 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.

The position embedding specifies the position of each character in a word as a pair of entity. We thus replace the target position of a character with its corresponding entity pair. We further compute the relative distance $\partial_{i,j}^{\phi}$ between $\phi - th$ character x_i^{ϕ} and target entity e_j^{ϕ} as follows:

$$\partial_{i,j}^{\phi} = x_i^{\phi} - e_j^{\phi}, \quad \forall i \in \{1, 2, \cdots, \nu\}, \text{ and } \quad \forall j \in \{1, 2\}, \tag{3.6}$$

where x_i^{ϕ} and e_j^{ϕ} represent the position of current character x_i and target entity e_j in

a word. For instance, in a given word "HOSPITAL", the relative position from the character "P" to entity "O" (e_1) is +2 ($\partial_{4,1}$) and entity "T" (e_2) is −2 ($\partial_{4,2}$). Each relative distance between current character and target entity is mapped to position vector. In the output feature tensor F contains the characters with relative position vectors. We aggregate the embedding feature tensor F with input feature tensor $\mathbf{v}$ to obtain a feature tensor P having channel $\zeta = 128$.

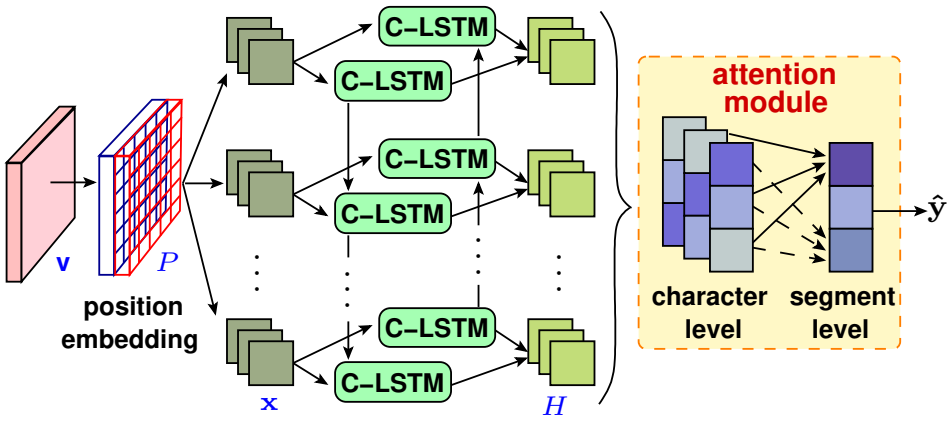

Figure 3.6: Architecture of the recognition module.

Next, we predict a label sequence of character class by incorporate C-LSTM [119], a variant of LSTM, in Bi-LSTM network (BiC-LSTM). Each cell in BiC-LSTM network is a C-LSTM cell. It is capable of capturing both local and global semantic information of a word. We therefore utilize it for label prediction, especially in a cluttered environment. The feature tensor P is fed into the Bi-LSTM network with both a forward and a backward sequence to learn most discriminate spatial information. Let C-LSTM accepts the input sequence $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_t, \cdots, \mathbf{x}_T)$ from P, where each of $\mathbf{x}_t$ is a vector corresponding to step $t \in \{1, 2, \cdots, T\}$. Here, we formulate the equations of C-LSTM

as follows:

$$\mathbf{i}_t = \sigma(\mathfrak{W}_{ix} * \mathbf{x}_t + \mathfrak{W}_{ih} * \mathbf{h}_{t-1} + \mathfrak{W}_{ic} \circ \mathbf{c}_{t-1} + \mathbf{b}_i), \tag{3.7}$$

$$\mathbf{f}_t = \sigma(\mathfrak{W}_{fx} * \mathbf{x}_t + \mathfrak{W}_{fh} * \mathbf{h}_{t-1} + \mathfrak{W}_{fc} \circ \mathbf{c}_{t-1} + \mathbf{b}_f), \tag{3.8}$$

$$\mathbf{m}_t = \mathbf{tanh}(\mathfrak{W}_{mx} * \mathbf{x}_t + \mathfrak{W}_{mh} * \mathbf{h}_{t-1} + \mathbf{b}_m), \tag{3.9}$$

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{m}_t, \tag{3.10}$$

$$\mathbf{o}_t = \sigma(\mathfrak{W}_{ox} * \mathbf{x}_t + \mathfrak{W}_{oh} * \mathbf{h}_{t-1} + \mathfrak{W}_{oc} \circ \mathbf{c}_{t-1} + \mathbf{b}_o), \tag{3.11}$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \mathbf{tanh}(\mathbf{c}_t), \tag{3.12}$$

$$\mathbf{y}_t = \mathfrak{W}_{yh} * \mathbf{h}_t, \tag{3.13}$$

where $\circ$ and $*$ denote the Hadamard product and convolution operation, whereas $\mathbf{b}_\iota$, $\mathfrak{W}_\iota$, $\sigma(\cdot)$, and $\mathbf{tanh}(\cdot)$ represent bias vectors, weight matrices, logistic element-wise sigmoid function, and hyperbolic tangent function, respectively. Also, the symbols $\mathbf{c}$, $\mathbf{f}$, $\mathbf{m}$, $\mathbf{i}$, $\mathbf{o}$, $\mathbf{h}$, and $\mathbf{y}$ present the cell state, forget gate, input modulation gate, input gate, output gate, hidden state, and cell output, respectively. On passing through BiC-LSTM, the feature tensors are transformed into several segments and each segment is associated with multiple characters. As we are dealing with the detection of scene text in a cluttered environment, the identification of semantic relation between the characters in a word is also important. The semantic relation of a word is determined by certain key characters. We therefore apply the attention mechanism to provide weights to the important characters in a word automatically.

BiC-LSTM outputs $H = \{\mathbf{h}_1, \cdots, \mathbf{h}_t, \cdots, \mathbf{h}_T\}$, a matrix of hidden states, on which we apply character based attention mechanism to determine the most influential characters in $t-th$ segment $\mathbf{h}_t$. The character-level attention is computed by weighted sum

of characters in $\mathbf{h}_t$ as follows:

$$\mathbf{g}_t = \mathbf{tanh}(\mathbf{h}_t \cdot \alpha_t^T), \tag{3.14}$$
$$\alpha_t = \mathbf{softmax}((\mathfrak{W}_t^{\alpha})^T \cdot \mathbf{tanh}(\mathbf{h}_t)),$$

where $\mathbf{g}_t$ is the character-level attention vector, $\mathfrak{W}_t^{\alpha}$ is the model parameter, and α_t is the weight vector of $\mathbf{h}_t$. Furthermore, we apply segment-level attention mechanism on the output matrix $G = (\mathbf{g}_1, \cdots, \mathbf{g}_t, \cdots, \mathbf{g}_T)$ to weigh the importance of each segment in a word with the semantic relation η, we use bilinear function as follows:

$$\xi_t = (\mathbf{g}_t)^T \cdot \mathfrak{D} \cdot \eta_{emb}, \tag{3.15}$$

where $\mathfrak{D}$ is the weighted diagonal matrix learned during the training process and η_{emb} is the embedding of relation η. The normalized importance weight Φ_t is calculated by:

$$\Phi_t = \frac{\mathbf{exp}(\xi_t)}{\sum_{t'=1}^{T} \mathbf{exp}(\xi_t')}. \tag{3.16}$$

The final attention based vector $\mathbf{z}$ is obtained by a linear weighted combination of all segments as follows:

$$\mathbf{z} = \sum_{t=1}^{T} \Phi_t \mathbf{g}_t. \tag{3.17}$$

Finally, we utilize a softmax function and a linear transformation to calculate the conditional probability ($\mathcal{P}$) as follows:

$$\hat{\mathbf{y}} = \underset{\dot{\mathbf{y}}}{\mathbf{argmax}}\ \mathcal{P}(\dot{\mathbf{y}}), \tag{3.18}$$
$$\mathcal{P}(\dot{\mathbf{y}}|O) = \mathbf{softmax}(\mathfrak{W}_o \times \mathbf{z} + \mathbf{b}_o),$$

where O denotes all parameters of our model. $\mathfrak{W}_o$ and $\mathbf{b}_o$ are learnable weight and bias.

3.2 Experimental Results

The effectiveness of our Cluttered TextSpotter (CTS) is validated by conducting a set of comprehensive experiments on publicly available benchmark datasets for scene text detection, word spotting, and end-to-end recognition. We pretrain our model using synthtext [8] dataset for three epochs and initialize the weights from ImageNet [9] in the detection task, whereas weights are randomly initialized from $\mathcal{N}(0, 1)$ distribution for the recognition process. Data augmentation is also implemented to improve the robustness of our network. We split the training/testing in a three-fold manner for evaluation and use standard metrics to measure the performance in terms of accuracy and training parameters.

•**Training.** We jointly train the detection and recognition module of our proposed network for three epochs on a merged dataset of ICDAR 2015, ICDAR 2013, SVT, COCO-Text, and MSRA-TD500 with a fixed sample ratio 2:2:1:1:1. We randomly cropped up to 30% of height and width each image. We set the mini-batch to 8 for initial experiments. In the case of RPN and Fast R-CNN, the batch sizes are maintained at 256 and 512.

The end-to-end training use curriculum learning [120] technique to train the model from gradual to complex data efficiently. We train multiple tasks jointly in a single model by propagating the generalization capability from synthesized images to real-world data; (1) we randomly pick 600k images from 800k synthetic images. The training of the recognition branch is performed by freezing the detection branch, where 120k iterations are utilized in the training process with a learning rate of 10^{-3}; (2) the next 80k iterations are utilized for detection only. The learning rate is set to 10^{-4}. In the next 20k iterations, we get sampling tensors from the detection task. The network is trained end-to-end in this stage; and (3) in the next 70k iterations, nearly 70k real-world images

from the COCO-Text, ICDAR 2013, MSRA-TD500, SVT, and ICDAR 2015 datasets are chosen. We also conduct data augmentation [17] to enhance generalization ability. Furthermore, we incorporate character-level supervision by making a batch of size 4. The images are taken from the synthetic dataset. We are maintaining the learning rate as 10^{-4}.

Stochastic gradient descent with adam optimizer, a weight decay of 10^{-3}, and a momentum of 0.9 are used. The input is fed in mini-batches of ψ images, where $\psi = 8$. Due to the presence of less number of real samples, we use data augmentation and multi-scale training in the fine-tuning stage. Moreover, we perform a random rotation of input images in a range of $[-30°, 30°]$ angles. Due to the presence of cluttered background noise, some background textures appear similar to the text instances in noisy scene images, which makes it hard for a network to distinguish between text and non-text instances. This compels the training process to be unbalanced and leads to slow convergence. We therefore use hard a negative mining strategy [17] to suppress training unbalance. We perform two-stage training on a dataset. In the first stage, the negative ratio between negatives and positives is set to 3:1, whereas it changed to 6:1 in the second stage. We select a multi-scale training mechanism [121] to make our network robust. Furthermore, we also use several data augmentation techniques [122,123]. Also, we provide an input image with a larger size to achieve better detection of multi-scale text at the third stage of training. For multi-scale training, we randomly resize the shorter sides of the input images to $(600, 800, 1000, 1200, 1400)$ scales randomly.

Interference. In the inference stage, we use a single model to evaluate all datasets, but the scales of the input images depend on the datasets. We obtain a predefined number, *i.e.*, 300 text region proposals from RPN in our experiment through a forward pass and then get the outputs for detection and recognition tasks. We provide *strong*, *weak*, and *generic* dictionaries for testing reference [96]. In the strong lexicon, 100 words per-image are assigned including all words that appear in the image. The weak

lexicon consists of all words that present in the entire test set of the dataset, whereas the generic dataset has 90k words. We kept the words of length greater than three in dictionaries, where signs and numbers are excluded. We prefer to use two models for evaluation, *i.e.*, end-to-end and word-spotting. The end-to-end model recognizes all the words accurately, even if a detected string is not present in the dictionary. While the word-spotting model only inspects about the presence of the word of the dictionary in the images. Therefore, it is less strict than the end-to-end model for avoiding numbers, symbols, and words whose length is less than 3. We evaluate all results in one single scale without referring to any lexicon.

3.2.1 Ablation Study

In this section, we perform extensive analyses and ablation studies to evaluate the detection and recognition accuracy and computational efficiency of the proposed network. We conduct a comprehensive set of experiments to study different aspects of our network. The experiments were undertaken on split 1 of ICDAR 2013, ICDAR 2015, SVT, COCO-Text, and MSRA-TD500 datasets.

• **Impact of backbone network.** We have carried out a large set of experiments to select a backbone network enriched in spatial information with an optimal number of parameters. We study the impact of MobileNetV2, MobileNetV2+ASPP, MobileNetV2 + ASPP + Encoder-Decoder (Ours), SSDLite, ShuffleNetV2 [124], and IGCV2 [125] as backbone network in the evaluation of f-measure of our proposed network. Table 3.1 depicts that our backbone network outperforms other networks in terms of training parameters and computational complexity of the network. Our backbone network is enriched with low-level spatial details and high-level context information. The network is not kept much deeper to restrict the number of training parameters.

• **Impact of ASPP layer and dilation rates.** We evaluate the importance of the ASPP layer in the proposed network. We also show the effect of a different set of dilation

Table 3.1: Effect of different variations of backbone network over ICDAR 2015 dataset.

Backbone	**F-measure**	**Flops** (G)	**Params** (M)	**MAdds** (B)
MobileNetV2 [115]	92.4	0.7	2.4	2.9
MobileNetV2+ASPP [115]	93	1.1	5.2	5.9
MobileNetV2+ASPP +Encoder-Decoder (Ours)	**93.7**	1.2	5.9	6.1
SSDLite [115]	92.8	**0.8**	**3.4**	**1.4**
ShuffleNetV2 [124]	91.7	2.8	10.9	8.7
IGCV2 [125]	91.3	4.6	23.1	10.4

rates. It is observed from Table 3.2 that the presence of the ASPP layer increases the recall measure of the network. Since text instances are relatively small in size with respect to the other objects present in the scene images, therefore, the ASPP layer with dilation rates $\{1, 2, 3, 4\}$ provides a better recall of the network.

Table 3.2: Effect of variation in dilation rate on ICDAR 2015 [2] and NAST dataset.

Methods	**ICDAR 2015**			**NAST**		
	Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
CTS (without ASPP)	80.1	75.3	77.6	47.4	42.8	45.1
CTS-(1,3,5,7)	87.4	82.8	85.1	54.6	49.5	52.0
CTS-(1,2,4,6)	86.3	78.6	82.4	56.5	52.3	54.4
CTS-(1,2,5,7)	87.8	81.5	84.6	53.3	48.6	51.0
CTS-(1,3,6,9)	89.5	84.9	87.2	57.2	53.9	55.5
CTS-(1,3,5,8)	88.2	83.2	85.7	55.8	48.7	52.2
Ours	**91.8**	**96.4**	**93.7**	**59.1**	**60.4**	**59.7**

• **Impact of Context Encoding and Refinement module.** In the Cluttered TextSpotter, the context encoding and refinement module consist of local, global, and context branches. We evaluate the impact that each branch on the overall performance of the CTS network, as shown in Table 3.3. For this evaluation, we make ablation studies, where we consider the CTS network as the baseline and create three more models. We drop the global and context branches and name the network as *CTS-LP* and perform experiments on it using split 1 of ICDAR 2015 and SVT datasets. Likewise, only the global structure information of the global context branch is used in this model is denoted by *CTS-GS*. This model has both global structure information and

Table 3.3: Effect of different branches of context encoding and refinement module.

Models	ICDAR 2013			SVT		
	Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
CTS-LP	96.9	94.4	95.5	84.8	76.1	78.6
CTS-GS	96.2	94.9	95.4	84.2	76.4	78.9
CTS-GC	97.2	95.1	95.9	85.3	76.8	79.1
Ours	**97.6**	**95.4**	**96.1**	**85.8**	**77.1**	**79.6**

context information of the global context branch. It provides a means to evaluate the contribution of context branches to the performance of the overall network. Context information helps to extract text information efficiently, even in a cluttered environment. This branch is known as *CTS-GC*.

• **Impact of inter-class interface.** We perform experiments with softmax (vanilla) and variations of G-softmax functions for the detection of text instances. The G-softmax function consistently outperforms the softmax function on COCO-Text datasets, as shown in Table 3.4. G-softmax function is utilized to quantify the compactness and separability of features. In our analysis, we observe that the improvement of intra-class compactness and inter-class separability improves the average precision of the detection network.

Table 3.4: Effect of different softmax functions on COCO-Text dataset.

Function	Precision	Recall	F-measure
Softmax	68.1	59.2	63.2
G-softmax ($\mu = 0$, $\sigma = 1$)	72.3	58.3	63.7
G-softmax ($\mu = 0$, $\sigma = 5$)	71.3	58.6	62.9
G-softmax ($\mu =$ -0.1, $\sigma = 1$)	74.2	**59.8**	**67.4**
G-softmax ($\mu = 0.1$, $\sigma = 1$)	**75.1**	57.9	66.8

• **Impact of size of RoIs in Detection module.** We evaluate the influence of aspect-ratios and scales of the ROIs for the detection of scene text instances. Table 3.5 and Table 3.6 depict that the choice of aspect-ratio and scales in our network shows better precision for the detection of text instances. This is because we capture both at word level and text-line level. Therefore, detected text instances are small but long in nature.

Table 3.5: 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	89.2	86.4	83.8	55.5	53.1	54.3
1:3, 1:6, 1:9	92.5	89.7	86.1	54.1	50.7	52.4
1:2, 1:6, 1:9	90.1	85.2	81.7	53.7	52.4	53.1
1:3, 1:5, 1:7	91.4	88.5	85.2	56.8	54.3	55.5
1:2, 1:5, 1:8	**97.6**	**95.4**	**96.1**	**59.1**	**60.4**	**59.7**

Table 3.6: 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	90.2	87.4	88.8	48.8	44.2	46.5
8, 16, 24	93.8	90.1	91.9	50.5	47.8	49.1
8, 16, 32	**97.6**	**95.4**	**96.1**	**59.1**	**60.4**	**59.7**
16, 24, 32	95.4	92.8	94.3	53.2	52.3	52.7
16, 32, 64	94.5	93.6	94.1	53.2	51.7	52.4

• **Impact of different branches of Recognition module.** The recognition module has three branches, *i.e.*, position embedding, attention module, and Bi-LSTM layers. Table 3.7 demonstrates that all the branches are important for recognition with high accuracy.

• **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 3.8. 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 3.9 and Table 3.10.

Table 3.7: Effect of different branches of recognition module on ICDAR 2015 and NAST dataset.

Position Embedding	C-LSTM	Attention		Word recognition	
		Character	Segment	ICDAR 2015	NAST
✗	✓	✗	✗	77.5	34.8
✗	✓	✓	✗	80.2	42.8
✗	✓	✗	✓	79.3	39.8
✗	✓	✓	✓	87.7	45.8
✓	✓	✗	✗	76.4	38.8
✓	✓	✓	✗	85.5	52.8
✓	✓	✗	✓	83.1	50.8
✓	✓	✓	✓	**89.2**	**54.8**

Table 3.8: Effect of variation in the number of channel in text spotting on ICDAR 2015 [2] dataset.

Number of channel	Params (M)	End-to-End			Word-Spotting		
		strong	weak	generic	strong	weak	generic
16	2.654	88.2	86.4	81.3	85.4	81.1	63.2
32	3.728	91.4	90.7	85.1	89.2	87.6	64.9
128	**5.941**	**96.3**	**95.4**	**89.2**	**94.3**	**92.5**	**73.8**
256	9.263	96.5	95.2	90.8	94.8	92.6	73.5
512	13.571	96.9	95.8	91.3	95.2	94.8	74.1

• **Impact of different devices.** We implement the proposed text spotter on several smartphones, as shown in Fig. 3.7. The technical specifications, such as processing speed and memory, are shown in Table 3.11. We use a Monsoon power monitor that can measure the power consumption of smart devices, alike literature [126]. It is to be noted from the result that our CTS network is competent with smart devices.

3.2.2 Comparison with State-of-the-Art Results

In this section, we compare our network with the state-of-the-art approaches [1,15,17,21, 22, 24, 25, 28–30, 39, 50, 52, 56, 94–96, 99, 99, 102, 103, 105, 118] on six different benchmark datasets. We consider recall, precision, and f-measure as the metrics for evaluation of accuracy of detection. Alike [1, 94, 99], we conduct experiments to compare our recognition results based on *strong*, *weak*, and *generic* lexicons. We also perform a

Table 3.9: 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	89.2	86.4	83.8	85.5	79.6	69.7
1:3, 1:6, 1:9	92.5	89.7	86.1	88.8	83.5	72.1
1:2, 1:6, 1:9	90.1	85.2	81.7	83.7	80.8	70.3
1:3, 1:5, 1:7	91.4	88.5	85.2	86.8	81.2	68.6
1:2, 1:5, 1:8	**96.3**	**95.4**	**89.2**	**94.3**	**92.5**	**73.8**

Table 3.10: 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	89.9	86.3	81.5	88.8	85.6	66.4
8, 16, 24	92.7	89.5	85.3	90.5	87.4	69.2
8, 16, 32	**96.3**	**95.4**	**89.2**	**94.3**	**92.5**	**73.8**
16, 24, 32	95.8	93.1	89.6	93.2	90.3	72.7
16, 32, 64	94.1	92.7	88.1	92.8	89.2	70.4

comparative study for parameter count of our network with the existing approaches.

• **Detection results on different datasets.** Cluttered TextSpotter is compared with the recent literature for both detection and recognition on standard evaluation metrics, like precision, recall, and f-measure. It is clear from the results tabulated in Table 3.15 that detection accuracy of our network is improved by more than 1% in terms of f-measure on both ICDAR 2013 and ICDAR 2015 datasets, respectively, with respect to the baseline literature FOTS [1]. Table 3.13 and Table 3.14 show that our network outperforms existing approaches on MSRA-TD500 and COCO-Text datasets. CTS performs better in terms of recall on SVT datasets, as depicted in Table 3.12.

• **Recognition results on different datasets.** The proposed network achieves state-of-the-art end-to-end text recognition accuracy for all three lexicons, as shown in Table 3.17, for ICDAR 2013 and Table 3.18 for ICDAR 2015 dataset. CTS performs better existing literature for both strong and generic lexicons for word spotting. Table 3.16 illustrates that our network outperforms recent literature for distorted text instances

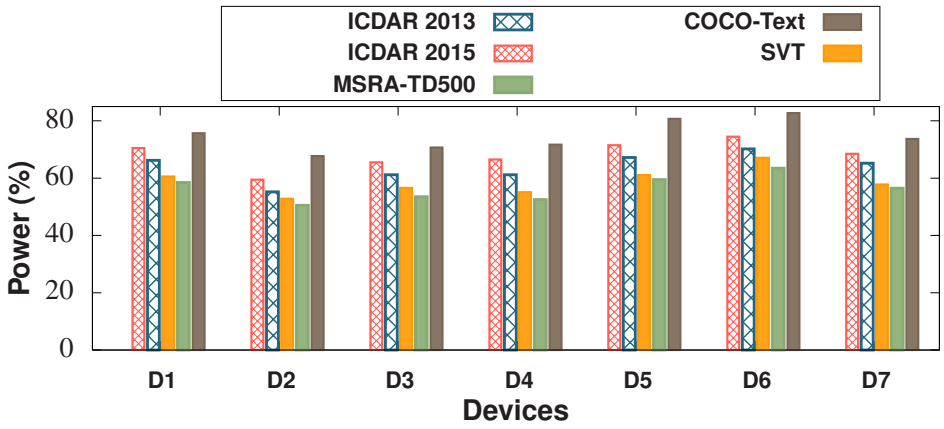

Figure 3.7: Effect of datasets on power consumption for different devices.

Table 3.11: 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

of SVT dataset.

• **Speed and Model Size.** We have reduced both the parameter count and computational cost by using a hardware-efficient backbone network and region proposal network. The test time speed of the proposed CTS network is reported in TABLE 4.19 and compared with state-of-the-art scene text detection methods. To evaluate the run time complexity, we have exhibit the flops, number of training parameters (params), and frames-per-second (fps) of our network.

Table 3.12: Performance comparison on SVT dataset.

Methods	**Precision**	**Recall**	**F-measure**
East [15]	50.3	32.4	39.4
Synthtext [8]	26.2	26.7	27.4
Raghunandan *et. al.* [42]	60.4	68.7	64.2
He *et. al.* [46]	**87**	73	79
Epshtein *et. al.* [47]	54.1	75.8	63.1
Dey *et. al.* [43]	55	68	61
Tang *et. al.* [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	85.8	**77.1**	**79.6**

3.3 Summary

In this paper, we propose a scene text spotter that can address the issue of cluttered environment of scene images. It is an end-to-end trainable deep neural network that uses local part information, global structural features, and context cue information of oriented region proposals for spotting text instances. It helps to localize in scene images with background clutters, where partially occluded text parts, truncation artifacts, and perspective distortions are present. Bilinear sampling is used for mapping the rotated region proposals into the canonical dimension. It also normalizes the rotation and scaling, keeping the aspect ratio and position of individual characters intact. The proposed text spotter works efficiently for the detection of text instances, logos, and symbols with high accuracy and speed. We design a backbone network rich in finer spatial details and multi-scale context information. It helps to reduce the degradation problem. We mitigate the problem of misclassification caused by inter-class interference by exploring inter-class separability and intra-class compactness with the help of Gaussian softmax. We also incorporate character segmentation and word-level classification in the recognition module. It recognizes words and text-line with high accuracy, even in the presence of partial occlusions and truncation artifacts. We conduct an exhaustive set of experimentation to show the efficacy of our network. We evaluated our spotter

Table 3.13: Performance comparison on MSRA-TD500 dataset.

Methods	**Precision**	**Recall**	**F-measure**
GISCA *et. al.* [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
Ours	**87.8**	**86.1**	**86.5**

on publicly available benchmark datasets for different smart devices. Our text spotter has outperformed previous methods in terms of efficiency and performance. It is also evident from the results that our network is light-weight, hardware-efficient, and robust in nature.

Table 3.14: Performance comparison on COCO-Text dataset.

Methods	**Precision**	**Recall**	**F-measure**
Cheng *et. al.* [45]	60	33	42
Yao *et. al.* [57]	43.2	27.1	33.3
Sheng *et. al.* [34]	74	51	61
TextBoxes++ [98]	60.8	56.7	58.7
Lyu *et. al.* [35]	61.9	32.4	42.5
RRD [36]	64	57	61
WordSup [37]	45.2	30.9	36.8
Mask TextSpotter [118]	66.8	58.3	62.3
Ours	**74.2**	**59.8**	**67.4**

Table 3.15: Performance comparison on ICDAR 2013 [3] and ICDAR 2015 [2] dataset.

Methods	**ICDAR 2013**			**ICDAR 2015**		
	Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
SegLink [17]	92.4	83.8	87.9	88	76.8	82
Zhong *et. al.* [28]	94.4	90.1	92.2	89.6	84.9	87.2
GISCA *et. al.* [30]	92.8	91.4	92.1	89.1	85.5	87.3
TextEdge [22]	88	84	86	88	84	86
He *et. al.* [50]	93	85.2	88.9	86.1	79.9	82.9
He *et. al.* [96]	91	88	90	87	86	87
Qin *et. al.* [39]	90	83	86	61	40	48
RefineText [21]	91.2	85.5	88.3	83.9	80.7	82.3
Mask-Most Net [52]	90.3	87.1	88.7	88.2	85.7	86.9
Yao *et. al.* [57]	88.8	80.2	84.3	72.2	58.6	64.7
Sheng *et. al.* [34]	93	86	90	89.9	80.4	84.9
Bagi *et. al.* [127]	94	91	93	**93**	90	91
TextBoxes++ [98]	84	91	88	87.8	78.5	82.9
Lyu *et. al.* [35]	92	84.4	88	89.5	79.7	84.3
RRD [36]	92	86	89	88	80	83.8
WordSup [37]	93.3	87.5	90.3	79.3	77	78.2
He *et. al.* [16]	92	81	86	82	80	81
RRPN [23]	95	88	91	84	77	80
Synthtext [8]	92	83	75.5	-	-	-
Raghunandan *et. al.* [42]	84.5	87.7	86	-	-	-
He *et. al.* [46]	90	75	81	-	-	-
Epshtein *et. al.* [47]	91.1	86.1	88.5	-	-	-
Dey *et. al.* [43]	67	87	75	-	-	-
Tang *et. al.* [48]	91.9	87.1	89.5	91.9	87.1	89.5
TextBoxes [97]	89	83	86	-	-	-
Khare *et. al.* [44]	57.3	63.4	60.2	-	-	-
CRAFT [38]	97.4	93.1	95.2	89.8	84.3	86.9
Mask TextSpotter [118]	94.8	89.5	92.1	86.6	87.3	87
Li *et. al.* [99]	-	-	-	83.7	96.1	89.5
Fots [1]	-	-	-	91.5	87.9	89.8
Duan *et. al.* [25]	-	-	-	90.4	86.7	88.5
Li *et. al.* [95]	-	-	-	91.4	80.5	85.6
Yang *et. al.* [29]	-	-	-	87	86.2	86.6
Tian *et. al.* [24]	-	-	-	88.3	85	86.6
Xiao *et. al.* [33]	-	-	-	87.9	81.6	84.6
Mohanty *et. al.* [40]	-	-	-	83	81	82
Mohanty *et. al.* [41]	-	-	-	82	83	82
Huang *et al.* [51]	-	-	-	90.8	81.5	85.9
TextField [49]	-	-	-	84.3	83.9	84.1
Mask TTD [54]	-	-	-	86.6	87.6	87.1
Dai *et. al.* [55]	-	-	-	86.2	82.7	84.4
Ours	**97.6**	**95.4**	**96.1**	**91.8**	**96.4**	**93.7**

Table 3.16: Performance comparison on SVT dataset.

Methods	**Word-Spotting**	
	strong	**generic**
TextBoxes [97]	84	64
Jaderberg *et. al.* [93]	76	53
Synthtext [8]	67.7	55.7
Li *et. al.* [95]	84.9	66.1
Ours	**85.8**	**67.6**

Table 3.17: Performance comparison on ICDAR 2013 dataset for the recognition.

Methods	**ICDAR 2013**					
	End-to-End			**Word-Spotting**		
	strong	**weak**	**generic**	**strong**	**weak**	**generic**
Li *et. al.* [99]	91.4	90.3	84.8	86	83.4	66.9
Fots [1]	91.9	90.1	84.7	83.5	79.1	65.3
Li *et. al.* [95]	-	-	-	91	89.8	84.5
MLTS [101]	87	84	70	73	67	59
He *et. al.* [96]	91	89	86	82	77	63
Deep TextSpotter [94]	89	86	77	54	51	47
TextBoxes++ [98]	93	**92**	85	73.3	65.8	51.9
TextBoxes [97]	91	89	84	-	-	-
TextDragon [105]	-	-	-	82.5	78.3	65.1
Mask TextSpotter [118]	93.3	91.3	88.2	92.7	**91.7**	87.7
ASTS [107]	92.8	91.5	85.9	84.8	79.8	66.5
Ours	**93.8**	**92**	**88.5**	**93.1**	**91.7**	**88.1**

Table 3.18: Performance comparison on ICDAR 2015 dataset for the recognition.

Methods	**ICDAR 2015**					
	End-to-End			**Word-Spotting**		
	strong	**weak**	**generic**	**strong**	**weak**	**generic**
Li *et. al.* [99]	96.1	95.1	89	89.6	87.1	70
Fots [1]	95.9	93.9	87.7	87	82.3	67.9
Li *et. al.* [95]	-	-	-	94.1	92.4	88.2
MLTS (spatial attention) [101]	88	85	72	-	-	-
MLTS (channel-wise attention) [101]	87	84	70	-	-	-
He *et. al.* [96]	93	92	87	85	80	65
Deep TextSpotter [94]	92	89	81	58	53	51
TextBoxes++ [98]	96	95	87	76.4	69	54.3
TextBoxes [97]	94	92	87	-	-	-
TextDragon [105]	-	-	-	86.2	81.6	68
Mask TextSpotter [118]	83	77.7	73.5	82.4	78.1	73.6
ASTS [107]	95.9	94.7	88.5	87.7	82.9	68.9
Boundary [110]	88.2	87.7	84.1	-	-	-
Text Perceptron (2-stage) [112]	90.8	90	84.4	93.7	93.1	86.2
Text Perceptron (end-to-end) [112]	91.4	90.7	85.8	94.9	94	88.5
TextNet [128]	89.7	88.8	82.9	94.5	93.4	86.9
Ours	**96.3**	**95.4**	**89.2**	**94.3**	**92.5**	**73.8**

Table 3.19: Test time speed in terms of FLOPS, number of training parameters, and frames per second (FPS) on ICDAR 2015 dataset for detection (D), recognition (R), or spotting (S).

Methods	**Flops** (G)	**Params** (M)	**fps**	**D/R/S**
Fots [1]	9.997	34.98	9.0	S
East [15]	4.685	24.1	13.2	D
E2E-MLT [102]	2.946	4.7	-	R
OctShuffleMLT [103]	1.525	4.8	-	R
OctShuffleMLT [103]+E2E-MLT [102]	**0.829**	**1.6**	-	R
Craft [38]	10.239	20.8	-	D
Li *et. al.* [99]	-	-	3.7	S
Deep TextSpotter [94]	-	-	9	S
Ours	1.139	5.914	**24.8**	S