Chapter 4

Blurred TextSpotter

In scene text spotting, detection and recognition tasks are highly correlated. Detection is responsible for accurate bounding box over the detected scene text and helps in improvement of the recognition accuracy, while in the recognition text script are identified by eliminating false positives. Traditional text spotting is challenging due to large variations in scale, symbols, text shape, orientations, aspect ratios, font size, font style, and script. In this chapter, we present a text spotter for text detection and recognition in blurry scene images. It is another form of challenge in text detection compared to occlusion, which is mentioned in previous Chapter 3. Nowadays, scene images are captured very often using smartphones, cameras, and webcams. These captured images mostly suffer from camera shake and defocus issues, which makes the text spotting complex in captured images unless the experts click the images, as shown in Fig. 4.1. Blurred text instances occur mainly because of camera motion (motion blur), camera shake artifact, defocus blur, and the surface movement containing scene text instances. In a vision-based application like driver assistance system, drones, and robot navigation, blurred text instances cause serious issues by increasing the ill-localization of text instances in scene images. The perspectively distorted, blurred, and shaky text edges make the process of spotting very complicated and lead to misclassification. In the case of blurry and shaky scene images, we humans try to understand the context related to text instances and then pay attention on transformation invariant features to overcome the effect of perceptual distortion. We model our work based on the aforementioned process by capturing multi-scale context information followed by enhancing the transformation modeling capability of the network to classify scene texts in the presence of camera shake, motion blur, and geometric distortions.

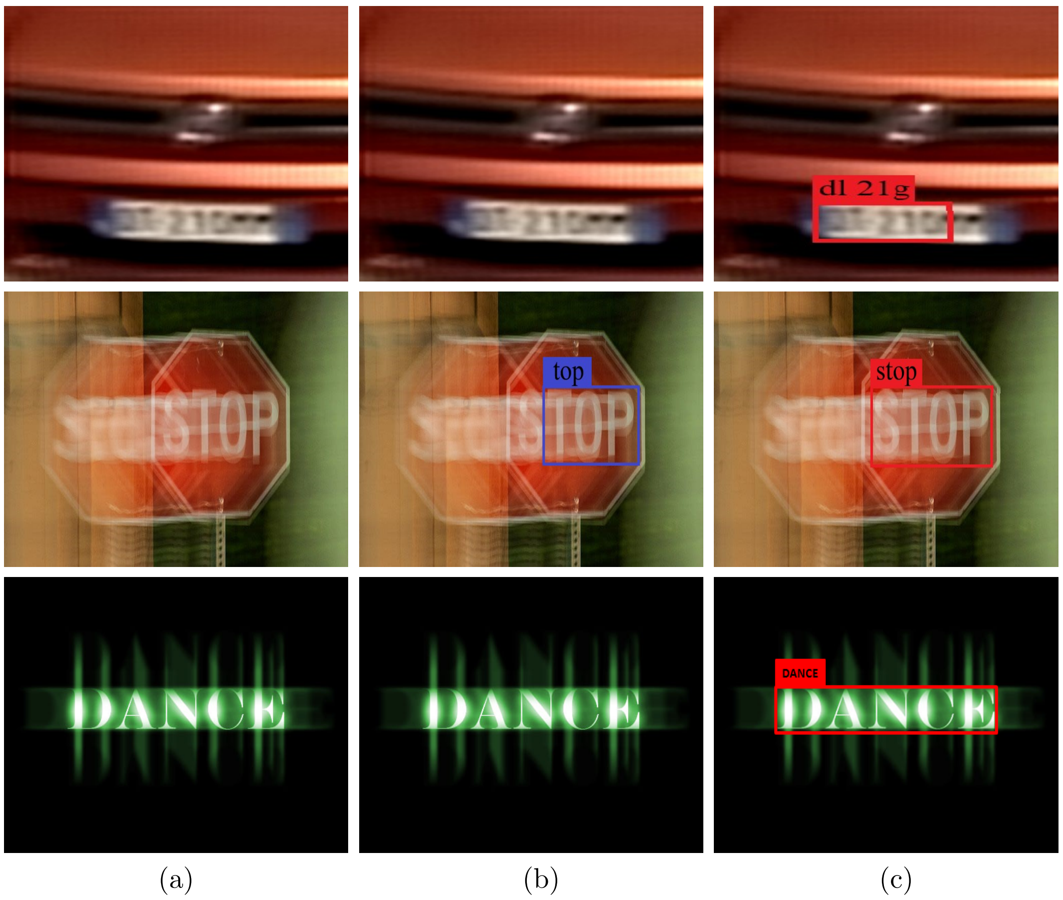


Figure 4.1: Illustration of the necessity of Blurred TextSpotter. Columns (b) and (c) are the recognized text instances in the scene images of column (a) using baseline [1] and our network. It can spot oriented text instances in the blurry scene images.

We therefore propose a deep network architecture in this chapter to handle blurred texts in the natural scene image. The overall architecture of the proposed blurred text spotter text spotting is shown in Fig. 4.2. The backbone network of the proposed system is composed of MobileNetV2, dilated pyramid pooling, and encoder-decoder architecture to capture multi-scale contextual information. We append spatial and channel-wise activations with the backbone network. We also use a region proposal network to obtain text proposals and an efficient recognition module using Bi-LSTM and attention-GRU for end-to-end spotting and word-level recognition.

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.

4.1 Proposed Architecture

In this section, we propose a text spotter named as Blurred TextSpotter that is robust to blur noise and effectively reduces the misclassification problem in text detection and spotting in blurry scene images. It is robust in nature. The overall architecture of the proposed text spotter is shown in Fig. 4.2.

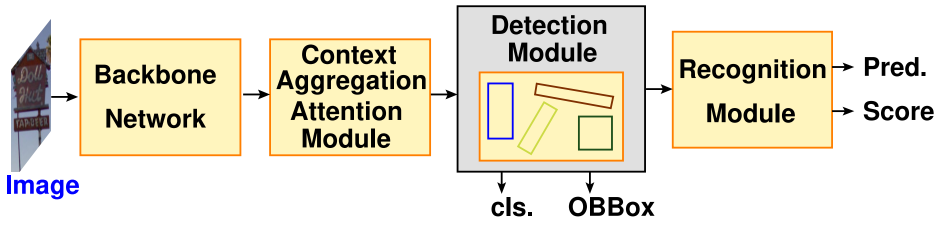


Figure 4.2: Overall Architecture of Proposed Network.

Blurred TextSpotter has the following modules:

- Firstly, we incorporate MobileNetV2 in the proposed backbone network, which makes it light-weight in nature. We also integrate stacked feature pooling and encoder-decoder architecture to capture multi-scale contextual information. This helps to prevent degradation problems. It helps to capture low-level fine details with high-level semantics.
- Next, we develop an attention module for precisely detect the text instances even in the presence of blur, perspective distortion, and inter-class interference. It

provides both spatial and channel-wise attention.

- Then, we design an oriented region proposal network to obtain text proposals. It is light-weight in nature.
- Lastly, we include a recognition module using C-LSTM within Bi-LSTM and attention-GRU for end-to-end spotting and word-level recognition.

4.1.1 Backbone Network

In this section, we propose a robust backbone network to extract multi-scale context enriched feature maps while preserving low, mid, and high-level spatial cues. We have incorporated basic blocks of MobileNetV2 [115]. It is used as a bottleneck depth separable convolutional layer with residuals. We include an encoder-decoder architecture to extract finer spatial information and prevent degradation problems in the backbone network. We have integrated MobileNetV2 with stacked feature pooling (SFP) blocks in the decoder of the backbone network to extract multi-scale contextual information without increasing the complexity of the model.

An input image I is fed into the encoder branch with a spatial resolution of $\mathcal{H}\times\mathcal{W}\times3$, where $\{\mathcal{H},\mathcal{W}\}$ is height and width of the image and 3 represents RGB channel. Initially, we have used a 3×3 convolution layer with 32 number of filters to produce the feature map with size $\mathcal{H}\times\mathcal{W}\times32$. Next, we feed these feature maps into 37 residual bottleneck layers (conv2 to conv8) followed by one 1×1 convolution layer with 1280 number of filters that produce the output feature map with size $\mathcal{H}/16\times\mathcal{W}/16\times1280$, as shown in Fig. 4.3. The stride value at the conv4 layer is changed from 2 to 1 to capture dense features. ReLU6 and batch normalization layers are accumulated on the top of each layer in the residual block to maintain the learning process and decrease the training iterations required to train the backbone network.

In the decoder branch, first, we apply the SFP block on the output feature map of conv9. An SFP block consists of three dilated convolutions with a kernel of size 3×3 and

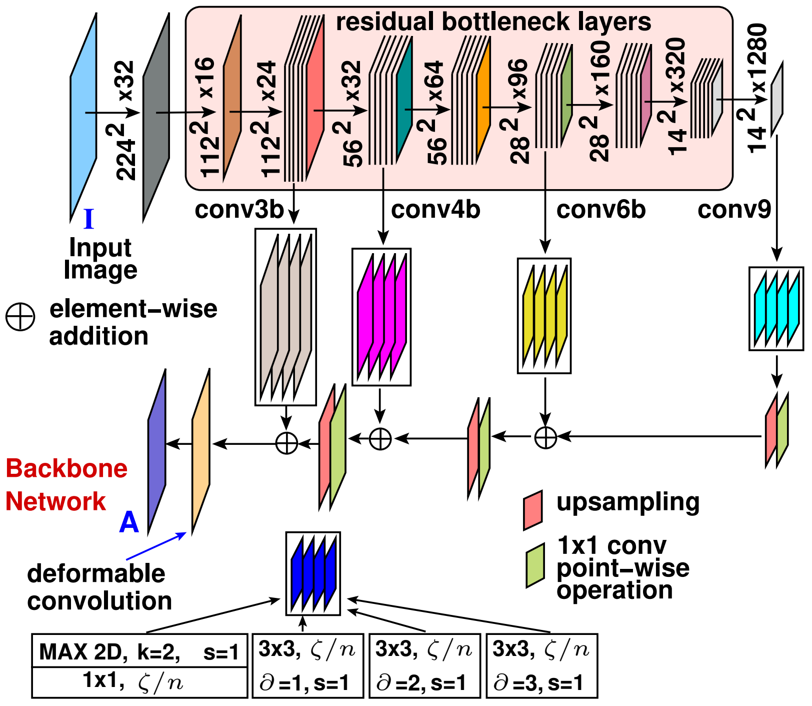


Figure 4.3: Architecture of backbone network.

dilation rate $\partial = \{1, 2, 3\}$, and each maintains the number of channels at ζ/n, where ζ represents the number of channels of output feature map of conv9. This helps to capture multi-scale context information along with spatial cues. A fourth incorporated operation is a 2D max-pooling layer with stride = 1, and the size of the kernel is set to 2×2. It is used to max-out activations over the local window. It is followed by 1×1 convolution layer to downsample the number of channels by a factor n, where $n = 4$. These four operations extract dense feature maps simultaneously. We stacked the output obtained from four (denoted by n) operations across channel dimensions to obtain the resultant with ζ channels. Next, we apply bilinearly upsampling to the output stacked feature map by a factor of 2. This upsampled feature map is concatenated with the stacked feature map of conv6b. In the case of conv3b, conv4b, and conv6b, prior to applying SFP block, the channels are upscaled to ζ using 1×1 convolution. The concatenate stacked feature map of conv9 + conv6b are bilinearly upsampled by a factor 2 and merged with the stacked feature map of conv4b. The concatenated stacked feature map of conv4b + conv6b + conv9 are concatenated with stacked feature map of conv3b to get the feature map $A' \in \mathbb{R}^{112\times112\times1280}$. The final output feature map A is obtained by applying a deformable convolution over A' to enhance the transformation modeling capability of the network. The kernel size of deformable convolution is 3×3 with stride one. $A \in \mathbb{R}^{112\times112\times1280}$ is the output feature map of the CEB network. All concatenation operations are element-wise addition.

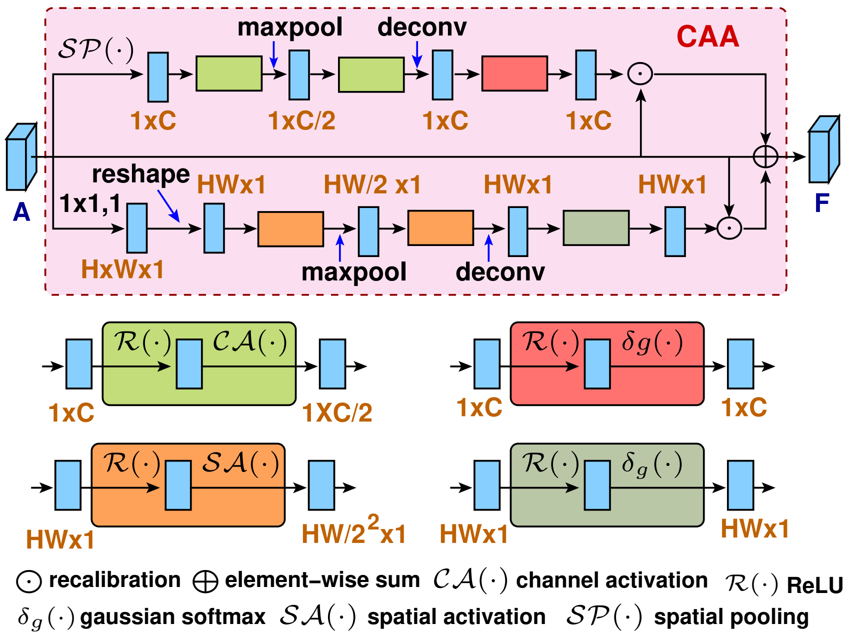
4.1.2 Context Aggregation Attention (CAA) module

Intra-class compactness and inter-class separability of features denote how far are the features with different labels and thus are correlated to the quality of the learned features. Due to the presence of blurred edges, both the intra-class compactness and the inter-class separability are simultaneously minimized, which leads the learned features to become less discriminative. Semantic segmentation describes a problem of assigning one label to each pixel of an image with $\mathcal{N}$ different classes. We propose a context aggregation attention module to improve the intra-class compactness and inter-class separability by aggregating contextual information, which will improve the average precision.

The feature map A is fed into CAA-module as input, as shown in Fig. 4.4. The CAA module has two branches, *i.e.*, spatial-wise attention and channel-wise attention. In the spatial-wise attention branch, we squeeze the channels using 1×1 convolution operation on feature map A. The channel squeezed feature map is reshaped to have a size of $HW \times 1$, which is denoted by Q. We then apply the first spatial activation layer, named as $SA(\cdot)$, as follows:

$$\mathbf{r}_i = \mathbf{q}_i \times \sigma(\mathbf{w}_{ir} \times \mathcal{R}(\mathbf{q}_i) + \mathbf{b}_{ir}), \tag{4.1}$$

where $\mathbf{r}_i$ and $\mathbf{q}_i$ are the values at $i-th$ position in the feature map R and Q, respectively. w_j and b_j are learnable parameters. $\sigma(\cdot)$ is the sigmoid activation function. Next, we conduct **maxpool** operation to reduce the size by a factor of 2. The reduced feature



CAA
maxpool
deconv
$\mathcal{SP}(\cdot)$
1xC
1xC/2
1xC
1xC
A
F
reshape
1x1,1
HxWx1
HWx1
HW/2 x1
HWx1
HWx1
maxpool
deconv
$\mathcal{R}(\cdot)$ $\mathcal{CA}(\cdot)$
1xC
1XC/2
$\mathcal{R}(\cdot)$ $\delta g(\cdot)$
1xC
1xC
$\mathcal{R}(\cdot)$ $\mathcal{SA}(\cdot)$
HWx1
HW/2²x1
$\mathcal{R}(\cdot)$ $\delta_g(\cdot)$
HWx1
HWx1
⊙ recalibration ⊕ element-wise sum $\mathcal{CA}(\cdot)$ channel activation $\mathcal{R}(\cdot)$ ReLU
$\delta_g(\cdot)$ gaussian softmax $\mathcal{SA}(\cdot)$ spatial activation $\mathcal{SP}(\cdot)$ spatial pooling

Figure 4.4: Architecture of context aggregation attention module.

map S is then passed through the second spatial activation layer to obtain the feature map U, defined by:

$$\mathbf{u}_i = \mathbf{s}_i \times \sigma(\mathbf{w}_{iu} \times \mathcal{R}(\mathbf{s}_i) + \mathbf{b}_{iu}), \tag{4.2}$$

where $\mathbf{u}_i$ and $\mathbf{s}_i$ are the elements at $i - th$ position in the feature map U and S, respectively. We further incorporate the third spatial activation layer after applying **deconv** operation on U to obtain the feature map X as follows:

$$\mathbf{x}_i = \mathbf{v}_i \times \sigma(\mathbf{w}_{ix} \times \mathcal{R}(\mathbf{v}_i) + \mathbf{b}_{ix}), \tag{4.3}$$

where $\mathbf{x}_i$ and $\mathbf{v}_i$ are $i-th$ element of X and the upscaled feature map V, respectively. It is assumed that the distribution of text-specific features with respect to the underlying background is a Gaussian distribution and thus include Gaussian softmax activation on X to obtain output feature map K, which is stated as follows:

$$\mathbf{k}_i = \mathcal{R}(\mathbf{x}_i) \times \frac{\mathbf{w}_{ik}\ \mathfrak{e}^{(\mathcal{R}(\mathbf{x}_i)+\lambda\Psi(\mathcal{R}(\mathbf{x}_i),\mu_i,\varphi_i))} + \mathbf{b}_{ik}}{\sum_{j=1} \mathfrak{e}^{(\mathcal{R}(\mathbf{x}_j)+\lambda\Psi(\mathcal{R}(\mathbf{x}_j),\mu_j,\varphi_j))}}, \tag{4.4}$$

where φ and μ denote standard deviation and mean of Gaussian distribution. The cumulative density function (CDF) is represented by $\Psi(\cdot)$ and $\mathbf{k}_i$ is the value at the $i-th$ position in K. We reshape the feature K to obtain O of dimension $H \times W \times 1$. The softmax function is dependent on λ for its usability. If $\lambda = 0$, it will be the traditional softmax function. Gaussian softmax approximates the distributions related to text-specific features. It considers large variations in the samples for training. The traditional softmax function typically learns from the current observing sample. The use of μ and φ helps to address intra-class compactness and inter-class separability.

In the channel-wise attention branch, we first perform spatial pooling, denoted by $SP(\cdot)$, which reshapes the feature map A to dimension $HW \times C$ followed by channel-wise average pooling to obtain a feature map D of size $1 \times C$. Then, we apply the first channel activation layer, represented by $CA(\cdot)$, as follows:

$$\mathbf{z}_i = \mathbf{d}_i \times \sigma(\mathbf{w}_{iz} \times \mathcal{R}(\mathbf{d}_i) + \mathbf{b}_{iz}), \tag{4.5}$$

where $\mathbf{z}_i$ and $\mathbf{d}_i$ are the values at $i-th$ position in the feature map Z and D, respectively. Next, we perform **maxpool** operation to reduce the channel size to $\frac{C}{2}$. The reduced feature map $E \in \mathbb{R}^{1 \times C/2}$ is then again passed through the second channel activation layer to obtain feature map G as follows:

$$\mathbf{g}_i = \mathbf{e}_i \times \sigma(\mathbf{w}_{ig} \times \mathcal{R}(\mathbf{e}_i) + \mathbf{b}_{ig}), \tag{4.6}$$

where $\mathbf{g}_i$ and $\mathbf{e}_i$ are $i-th$ element of the feature map G and E, respectively. We perform **deconv** operation to upscale the channel to C followed by the third channel activation layer to compute the obtained feature map L as follows:

$$\mathbf{l}_i = \mathbf{m}_i \times \sigma(\mathbf{w}_{il} \times \mathcal{R}(\mathbf{m}_i) + \mathbf{b}_{il}), \tag{4.7}$$

where $\mathbf{l}_i$ and $\mathbf{m}_i$ are the elements at $i-th$ position in the feature map L and M, respectively. We use Gaussian softmax activation on L to obtain output feature map P, which is given by:

$$\mathbf{p}_i = \mathcal{R}(\mathbf{l}_i) \times \frac{\mathbf{w}_{ip}\ \mathfrak{e}^{(\mathcal{R}(\mathbf{l}_i)+\lambda\Psi(\mathcal{R}(\mathbf{l}_i),\mu_i,\varphi_i))} + \mathbf{b}_{ip}}{\sum_{j=1} \mathfrak{e}^{(\mathcal{R}(\mathbf{l}_j)+\lambda\Psi(\mathcal{R}(\mathbf{l}_j),\mu_j,\varphi_j))}}, \tag{4.8}$$

where $\mathbf{p}_i$ is the $i-th$ value in the feature map P. The feature map A is rescaled with the attention maps of spatial and channel-wise attention branches as follows:

$$\widehat{O} = \{\mathbf{o}_1\ \mathbf{a}_1, \cdots, \mathbf{o}_{H\times W}\ \mathbf{a}_{H\times W}\} \tag{4.9}$$

$$\text{and } \widehat{P} = \{\mathbf{p}_1\ \mathbf{a}_1, \cdots, \mathbf{p}_C\ \mathbf{a}_C\}, \tag{4.10}$$

where $\widehat{O}$ and $\widehat{P}$ are the rescaled map of spatial and channel-wise attention branches. $\mathbf{o}_i$ and $\mathbf{a}_i$ are the $i-th$ values in O and reshaped feature map A. These rescaled maps are finally concatenated using element-wise addition to obtain the output feature map F as follows:

$$\mathbf{f}_i = \mathbf{w}_1\widehat{\mathbf{o}_i} + \mathbf{w}_2\widehat{\mathbf{p}}_i + \mathbf{b}_{if}, \tag{4.11}$$

where $\widehat{\mathbf{o}_i}$, $\widehat{\mathbf{p}}_i$, and $\mathbf{f}_i$ is the $i-th$ element in the feature maps $\widehat{O}$, $\widehat{P}$, and F. The output of CAA-module is feature map F.

4.1.3 Detection Module

In this section, we aim to describe the text regions by using region-level features. We reduce the overhead latency of region proposals by using light-head feature maps. To obtain light-head text proposals, we use large separable convolution to get a thin feature map with number of channels as 128, which improves the speed of the detector. We reduce channels of the feature map F into a compact feature map. We apply the detection task on a thin feature map to reduce the overhead of oriented region proposal network and abandon global average pooling to improve performance. Oriented bounding-boxes are used for classification and regression. The output text proposals, denoted by $N \in \mathbb{R}^{H' \times W' \times C'}$, is fed to the recognition module.

Inspired by [1], we have incorporated RoIRotate for extracting salient features in a text proposal. We apply bilinear interpolation that helps to ignore misalignments between text proposals and the captured features. Further, RoIRotate gives variable-length output features, which is more appropriate for text spotting. We compute affine transformation parameters for text proposals based on the ground truth coordinates and share shared feature maps for each region to obtain canonical horizontal feature maps of text regions, which is formulated as:

$$\dagger_x = l\mathbf{cos}\theta \dagger \mathbf{sin}\theta x, \tag{4.12}$$

$$\dagger_y = \dagger\mathbf{cos}\theta + l\mathbf{sin}\theta y, \tag{4.13}$$

$$s = \frac{h_\dagger}{\dagger + b}, \tag{4.14}$$

$$w_\dagger = s(l + r), \tag{4.15}$$

$$\mathcal{M} = \begin{bmatrix} \mathbf{cos}\theta & -\mathbf{sin}\theta & 0 \\ \mathbf{sin}\theta & \mathbf{cos}\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \dagger_x \\ 0 & 1 & \dagger_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.16}$$

where $\mathcal{M}$ is the matrix for affine transformation. $\{h_\dagger, w_\dagger\}$ denote the spatial resolution of feature maps obtained after the transform. (x, y) indicates the coordinates of an element in shared feature maps. The distance to the top, bottom, left, right sides of the text proposal is represented by $(\dagger, b, l, r)$. θ is the orientation. The final text proposals are computed by:

$$\forall i \in [1...h_t], \forall j \in [1...w_t], \forall c \in [1...C] \tag{4.17}$$

and for

$$V_{ij}^{c} = \sum_{n}^{h_s} \sum_{m}^{w_s} U_{nm}^{c} \kappa(x_{ij}^{s} m; \Delta_x) \kappa(y_{ij} n; \Delta_y) \tag{4.18}$$

$$\begin{pmatrix} x_i^s \\ y_i^s \\ 1 \end{pmatrix} = \mathcal{M}^{-1} \begin{pmatrix} x_i^{\dagger} \\ y_i^{\dagger} \\ 1 \end{pmatrix} \tag{4.19}$$

where V_{ij}^{c} and U_{nm}^{c} are the (i,j) output value and (n,m) input value in channel c. $\{h_s, w_s\}$ define spatial resolution of the input. The sampling kernel $\kappa(\cdot)$ has $\{\Delta_x, \Delta_y\}$ parameters. It is a bilinear interpolation process. The output the text proposal are represnted by N. The aspect-ratio and scales of text proposals are kept same of architecture in Chapter 3, *i.e.*, $\{1:2, 1:5, 1:8\}$ and $\{8, 16, 32\}$, respectively.

4.1.4 Recognition Module

Humans have a great ability to focus on particular spatial features like texture, color, and shape in a scene to understand the overall context, referred as a cognitive visual mechanism. In computer vision, the ability of humans are replicated by integrating attention mechanism into the deep learning model that adaptively focus on the most important areas.

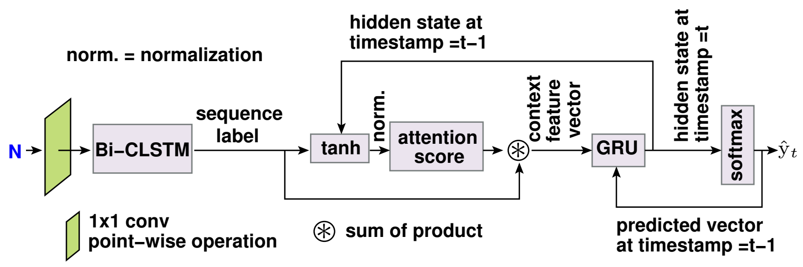


Figure 4.5: Architecture of recognition module.

In this section, we incorporate a robust recognition module to concentrate on the relevant text and overlook visual clutter. Our recognition module takes the text proposal

N as input from our network, as shown in Fig. 4.5. First, feature maps are channel-wise reduced using 1×1 convolution filter to obtain $\mathcal{C}$ dimension feature map. Next, we have utilized bidirectional circular long-short term memory (Bi-CLSTM) with $\mathcal{B}$ hidden states to generate label sequence. Finally, attention-based gated recurrent unit (GRU) is stacked on the top of Bi-CLSTM to improve the performance of recognizing the label of a text sequence. Inspired by [129], we incorporated the attention mechanism with GRU to predict the accurate label of a given sequence. Given the previous hidden state $\mathfrak{h}_{t-1}$ of GRU, we estimate the attention vector with a sequence feature vector $\mathcal{X}_t$ at $t - th$ timestep, which is calculated as:

$$\mathcal{Z}_{i,t} = \mathfrak{W}_z^\top \mathbf{tanh}(\mathbf{U}_h^\top \mathfrak{h}_{t-1} + \mathbf{U}_x^\top \mathcal{X}_i + \mathbf{b}_z), \tag{4.20}$$

where $\mathcal{Z}_{i,t}$ is un-normalized attention value of $\mathcal{X}_i$ sequence and $\{\mathfrak{W}_z, \mathbf{U}_h, \mathbf{U}_x, \mathbf{b}_z\}$ are the attention model trainable parameters. We normalized $\mathcal{Z}_{i,t}$ to estimate attention $\mathcal{V}_{i,t}$ score, which is computed as:

$$\mathcal{V}_{i,t} = \frac{\mathbf{exp}(\mathcal{Z}_{i,t})}{\sum_{j=1}^{K} \mathbf{exp}(\mathcal{Z}_{j,t})}, \tag{4.21}$$

where K is the length of feature maps. The glimpse of a given feature vector $\mathcal{X}$ is estimated by:

$$\mathcal{L}_t = \sum_{i=1}^{K} \mathcal{V}_{i,t} \mathcal{X}_i, \tag{4.22}$$

where $\mathcal{L}_t$ emphasizes the important context of text at every t step. After obtaining the context vector $\mathcal{L}_t$, we feed it into GRU as an extra input with a previous output value obtained by integrating $\mathfrak{h}_t$ into softmax. The hidden state $\mathfrak{h}_t$ of GRU is calculated as:

$$\mathfrak{h}_t = \Phi(\mathfrak{h}_{t-1}, \mathcal{L}_t, \hat{\mathrm{y}}_{t-1}), \tag{4.23}$$

$$\hat{\mathrm{y}}_t = \delta(\mathfrak{W}_y \mathfrak{h}_t + \mathrm{b}_y) \tag{4.24}$$

where $\hat{\mathrm{y}}_{t-1}$ and $\hat{\mathrm{y}}_t$ are the predicted vectors at time $t-1$ and t, respectively. $\{\mathfrak{W}, \mathrm{b}\}$ are learnable parameters, and $\{\Phi(\cdot), \delta(\cdot)\}$ represents GRU, and softmax function to calculate probabilities of text sequence.

4.2 Experimental Results

To evaluate the efficiency of Blurred TextSpotter (BTS) by conducting a comprehensive set of experiments for text detection, word spotting, and end-to-end recognition in scene images. ImageNet [9] and Synthtext [8] datasets is used as a pretrain model in our Blurred TextSpotter. Standard metrics are used for performance measures in terms of accuracy and number of training parameters. We consider five benchmark datasets for our experimentation.

4.2.1 Implementation Details

Similar as Chapter 3, a pretrained model is used on Synthtext [8] dataset in the detection process for three epochs that have weights initialized with ImageNet [9] dataset. In the case of a recognition task, we randomly assign the weights, which is initialized from $\mathcal{N}(0,1)$ distribution. We first use synthtext dataset [8] for training followed by fine-tuning on the benchmark datasets on which the network will be tested. We incorporate data augmentation to improve the robustness of our text spotter. Three training/testing splits are utilized for validation. Blurred TextSpotter is implemented on Intel E5-2670v3 CPU at 2.30 GHz speed and NVIDIA Titan X graphic card.

• **Training.** We train our detection and recognition models simultaneously on a combined dataset which is composed of ICDAR 2015, ICDAR 2013, SVT, COCO-Text, and Total-Text training datasets for three epochs with a sample ratio of 2:2:1:1:1:1. A random crop of up to 30% of its height and width is performed as preprocessing. The mini-batch size is kept 8 in the initial phase of experimentation. We maintain the batch sizes 256 and 512 for RPN and Mask R-CNN.

We perform end-to-end training, which is challenging with the presence of notable complexities in learning, unbalanced distribution of image data, and convergence rates. Synthtext [8] compute synthetic images for both character-level and word-level annotations. There is still a big gap between realistic and synthesized images, which may create difficulty in training our model to generalize real-world images. To bridge the gap and to accelerate the process of convergence, the curriculum learning approach is incorporate that helps to train the network progressively from complex data. A joint model is used to train multiple tasks that can propagate the generalization capability from synthesized images to real-world data. We randomly select 600k from the 800k synthetic images. We train the recognition task with 120k iterations and a learning rate 10^{-3}, where the detection branch is frozen. Another 80k iterations are used only for detection, where the learning rate is 10^{-4}. Next, 60k scene images of COCO-Text, ICDAR 2013, SVT, Total-Text, and ICDAR 2015 datasets are considered 70k iterations, which utilize data augmentation [17, 122] for enhancing the generalization capability. We keep the batch size as 4 for character-level supervision in the synthetic dataset with the learning rate at 10^{-4}.

Stochastic gradient descent and adam optimizer having a weight decay of 0.001 and a momentum of 0.9 is applied. The size of mini-batches is 8 images. The images are randomly rotated between $[-30°, 30°]$ angles precisely. In the presence of blurred scene images, some text instances will be similar to the background pattern. It is thus difficult for a network to distinguish and make the training unbalanced, which leads to slow convergence. We therefore incorporate hard negative mining strategy [17]. We train a dataset in two steps. In the first step, we set the negative ratio between the negatives and positives as 3:1. For the second step, it is modified to 6:1. We use data augmentation methods [122, 123] and multi-scale training [121] due to the lack of real samples for fine-tuning. We resize the shorter sides of the input images in multi-scale training to (600, 800, 1000, 1200, 1400) scale randomly. We also use an input image with a larger size at the third stage of training to achieve better detection of multi-scale text.

• **Interference.** We apply a single model in the inference stage to evaluate the datasets, where scales of the input images is based on the datasets. We provide a predefined number, *i.e.*, 300 text region proposals in a forward pass, which are utilized to compute the detection and recognition outputs. Three lexicons, *i.e.*, *strong*, *weak*, and *generic* dictionaries are used for testing reference [96]. The strong lexicon includes most words that are present in the image or 100 entries per image. In the weak lexicon, all words appear in the dataset. The generic dataset consists of 90k words. We choose the words that are of length three or more in dictionaries. It however excludes signs and numbers. We select end-to-end and word-spotting models for evaluation. In the end-to-end model, all the words are recognized accurately, even if a detected string absent in the dictionary. In case of word-spotting model, we only look for the existence of the word of the dictionary. Thus, the word-spotting model is lenient by ignoring numbers, symbols, and words that are of a length less than 3. Our evaluation is performed on a single scale without referring to any lexicon.

4.2.2 Ablation Study

In this section, we perform ablation studies for detection and recognition of text instances in scene images. We perform a comprehensive set of experiments to explore different aspects of our network. We conduct all experiments on split 1 of ICDAR 2013, ICDAR 2015, NAST, COCO-Text, MSRA-TD500, and SVT datasets. The standard metrics precision (P), recall (R), and f-measure (F) are used for evaluation of detection accuracy. We perform experimentation to compare our recognition results depending on *strong*, *weak*, and *generic* lexicons.

• **Effect of backbone network.** We conduct exhaustive experiments to select a backbone network that is rich in spatial information on the ICDAR 2015 dataset with an optimal number of parameters. We study the impact of MobileNetV2, SSDLite MobileNetV2+ASPP, MobileNetV2+ASPP+Encoder-Decoder, MobileNetV2+SFP+ Encoder-Decoder + Deformable layer (Ours), IGCV2 [125], and ShuffleNetV2 [124] as backbone network on the overall performance (f-measure) of Blurred TextSpotter. Table 4.1 shows that with an optimal number of training parameters, our backbone model outperforms other models. We keep the network less deeper in order to limit the number of training parameters. A deformable layer helps in precise detection.

Table 4.1: Impact of different variations of MobileNetV2, IGCV2, and ShuffleNetV2 as backbone networks on ICDAR 2015 dataset.

Backbone	**F**	**Flops** (G)	**Params** (M)	**MAdds** (B)
MobileNetV2 [115]	92.7	0.9	3.8	2.9
SSDLite [115]	93.1	**0.8**	**3.5**	**1.5**
MobileNetV2+ASPP (one layer) [115]	93.6	1.3	5.4	6.1
MobileNetV2+ASPP+ Encoder-Decoder	93.8	1.6	6.2	6.3
MobileNetV2+SFP+ Encoder-Decoder	94.1	1.5	6.0	6.2
MobileNetV2+SFP+ Encoder-Decoder + Deformable layer (Ours)	**94.4**	1.6	6.1	6.2
MobileNetV2+SFP+ Encoder-Decoder + Deformable layer (2 layer)	**94.4**	1.9	6.4	6.4
IGCV2 [125]	91.5	4.8	23.4	10.5
ShuffleNetV2 [124]	91.9	3.1	11.0	8.8

• **Impact of SFP layer and Deformable layer.** We evaluate the importance of SFP layer in the proposed network. We also show the effect of different set of dilation rates and maxpool operation. It is observed from Table 4.2 that the presence of SFP 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 a SFP layer with dilation rates $\{1, 2, 3\}$ and a **maxpool** layer provides a better recall of the network.

Table 4.2: Performance comparison on ICDAR 2015 [2] and NAST dataset, where M stands for $\mathbf{maxpool}2D, 2 \times 2, \text{stride} = 1$.

Methods	**ICDAR 2015**			**NAST**		
	Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
BTS (without ASPP)	81.1	73.3	77.2	45.4	40.3	42.8
BTS-(M,1,3,5)	87.3	84.8	85.8	53.6	47.1	50.3
BTS-(M,1,2,4)	86.2	80.6	83.4	55.5	50.2	52.8
BTS-(M,1,2,5)	87.7	83.5	85.2	52.3	46.3	49.3
BTS-(M,1,3,4)	87.9	83.7	85.4	52.8	46.7	49.4
BTS-(1,2,3,4)	90.4	88.9	89.8	56.2	51.6	53.9
BTS-(1,2,3,5)	88.1	85.2	86.7	54.8	49.5	52.2
BTS-(1,2,4,5)	87.6	86.1	86.5	54.2	49.8	52.4
BTS-(1,3,4,5)	86.4	85.8	85.9	53.1	49.9	52.1
Ours (M,1,2,3)	**93.9**	**96.6**	**94.1**	**57.5**	**59.2**	**58.3**

• **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 4.3 and Table 4.4 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 4.3: 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	88.7	87.3	88.4	54.2	53.1	53.6
1:3, 1:6, 1:9	91.3	92.5	91.9	53.9	54.5	54.2
1:2, 1:6, 1:9	89.2	88.6	88.9	52.5	53.8	53.2
1:3, 1:5, 1:7	90.6	89.7	90.1	55.6	54.7	55.1
1:2, 1:5, 1:8	**94.7**	**95.5**	**94.8**	**57.5**	**59.2**	**58.3**

• **Impact of Attention module.** In the Blurred TextSpotter, the attention module consists of spatial and channel-wise attention. We evaluate the influence of each branch on the overall network, as shown in Table 4.5. We conduct ablation studies considering the BTS network as the baseline and create four more models. We exclude

Table 4.4: 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	90.2	90.8	50.6	48.8	49.4
8, 16, 24	93.5	91.3	92.4	54.2	55.3	54.7
8, 16, 32	**94.7**	**95.5**	**94.8**	**57.5**	**59.2**	**58.3**
16, 24, 32	94.2	93.8	94.0	53.7	52.4	53.0
16, 32, 64	92.8	92.1	92.4	52.3	55.9	54.1

Table 4.5: Impact of different branches of attention module.

Models	ICDAR 2013			SVT		
	Precision	Recall	F-measure	Precision	Recall	F-measure
BTS-X	91.9	91.4	91.5	82.8	71.1	75.3
BTS-S	93.2	92.9	93.1	85.2	76.4	78.1
BTS-C	94.1	93.1	93.9	86.3	76.8	78.4
BTS-G	94.2	94.1	94.1	85.9	76.9	79.1
Ours	**94.7**	**95.5**	**94.8**	**87.9**	**77.5**	**79.8**

the attention module and name the rest as *BTS-X* and perform experimentation using split 1 of ICDAR 2013 and SVT datasets. Similarly, spatial attention is only used in this model, which is denoted by *BTS-S*. In the next model, channel-wise attention is only incorporated. This model is known as *BTS-C*. In the last model, we only use the Gaussian softmax normalization, which is indicated by *BTS-G*.

• **Impact of branches in recognition module.** The recognition module has three branches, *i.e.*, GRU, attention module and Bi-LSTM layers. Table 4.6 demonstrates that all the branches are important for recognition with high accuracy.

Table 4.6: Effect of different branches of recognition module over COCO-Text and NAST dataset.

GRU	Bi-CLSTM	Attention Mechanism	Word recognition	
			COCO-Text	NAST
✗	✓	✗	53.6	36.2
✗	✓	✓	63.1	45.6
✓	✓	✗	55.7	41.8
✓	✓	✓	**69.3**	**53.4**

• **Impact of size of RoIs in recognition module.** The aspect-ratios, scales and number of channels of the RoIs affects the recognition process. With the increase in number of channels the recognition accuracy increases however it also increases the computational overhead. Therefore, we maintain a accuracy-cost trade-off, as shown in Table 4.7. The choice of aspect-ratios and scales taken in the proposed network helps in precise detection, which in turn enhances the recognition efficiency, as analyzed in Table 4.8 and Table 4.9.

Table 4.7: 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	76.5	77.2	74.6	85.6	83.1	62.8
32	4.54	81.3	83.6	80.2	89.3	86.5	65.6
128	**6.21**	**86.7**	**89.8**	**85.4**	**94.6**	**92.8**	**73.9**
256	9.67	87.1	86.3	85.3	94.8	90.2	72.1
512	12.89	87.6	86.9	86.4	95.1	91.5	74.3

Table 4.8: Effect of variation in size of RoI in text spotting on ICDAR 2013 [3] dataset.

Aspect-ratio	**End-to-End**			**Word-Spotting**		
	strong	**weak**	**generic**	**strong**	**weak**	**generic**
1:2, 1:3, 1:5	86.8	87.2	82.7	90.8	87.5	82.4
1:3, 1:6, 1:9	92.5	89.5	84.6	94.4	91.3	85.1
1:2, 1:6, 1:9	87.1	85.3	80.3	92.1	89.6	86.7
1:3, 1:5, 1:7	89.4	87.1	81.8	95.3	92.8	85.3
1:2, 1:5, 1:8	**94.1**	**91.5**	**88.7**	**96.4**	**94.8**	**88.9**

• **Impact of different devices.** We implement the proposed text spotter on several smartphones, as shown in Fig. 4.6. The technical specifications, such as processing speed and memory, are shown in Table 4.10. 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 BTS network is competent with smart devices.

Table 4.9: Effect of variation in scale of RoI in text spotting on ICDAR 2013 [3] dataset.

Scale	End-to-End			Word-Spotting		
	strong	weak	generic	strong	weak	generic
4, 8, 16	87.3	85.1	81.5	89.5	86.3	82.5
8, 16, 24	91.2	86.5	83.8	92.7	90.1	86.7
8, 16, 32	**94.1**	**91.5**	**88.7**	**96.4**	**94.8**	**88.9**
16, 24, 32	93.6	89.6	85.2	94.4	93.8	88.8
16, 32, 64	90.5	86.8	84.3	93.1	91.4	87.6

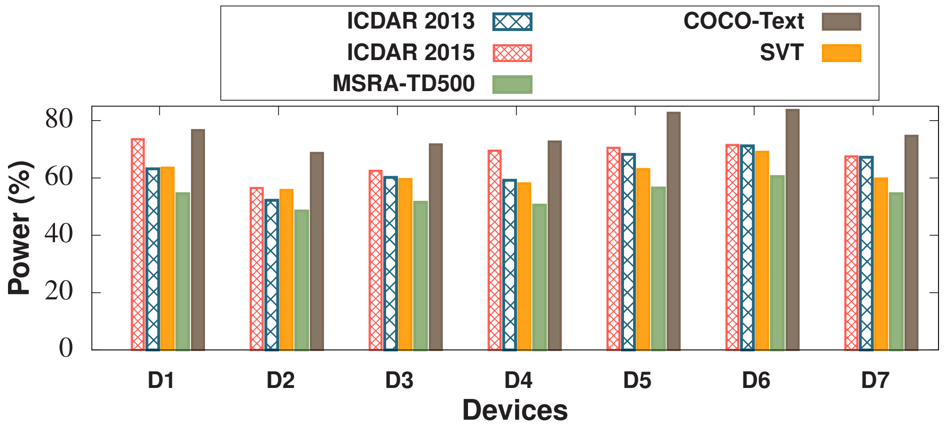


Figure 4.6: Effect of datasets on power consumption for different devices.

4.2.3 Comparison with State-of-the-Art Results

In this section, we compare our Blurred TextSpotter with the existing methods [1, 17, 38, 94, 96, 99, 102–105, 107, 112] on five different benchmark datasets. We conduct a comparative study for the parameter count of our network with the state-of-the-art methods.

• **Detection results on different datasets.** We compare Blurred TextSpotter with the recent literature for both detection and recognition on precision, recall, and f-measure evaluation metrics. We can conclude from the results in Table 4.11 and Table 4.12 that accuracy of detection in the case of Blurred TextSpotter is higher than 2% in terms of f-measure on ICDAR 2013 and ICDAR 2015 datasets in comparison with the baseline methods. BTS performs better in terms of recall on COCO-Text and SVT

Table 4.10: 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

datasets, as depicted in Table 4.14 and Table 4.15. The proposed spotter outperforms existing methods in the MSRA-TD500 dataset, as shown in Table 4.13.

• **Recognition results on different datasets.** The proposed Blurred TextSpotter accomplish state-of-the-art word spotting accuracy for all three lexicons, as shown in Table 4.16 and Table 4.17, for ICDAR 2013 and ICDAR 2015 datasets. BTS performs better than the existing methods for end-to-end text recognition in terms of both strong and generic lexicons. Table 4.18 depicts that our network provides good results even for blurred and distorted text instances in COCO-Text and SVT datasets.

• **Speed and Model Size.** The use of a cost-effective and light-weightrobust backbone network, region proposal network, and recognition module drastically decreases the parameter count and the computational overhead. Table 4.19 illustrates the test time speed in terms of flops, number of training parameters (params), and frames-per-second (fps). This evaluates the running time complexity.

4.3 Summary

In this paper, a robust solution is proposed for efficient sensing and spotting text instances in blurry scene images. The proposed Blurred TextSpotter uses MobileNetV2 and stacked feature pooling in the encoder-decoder architecture for the backbone network, which is rich in multi-scale contextual information. This mitogate the degra-

Table 4.11: 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	94.7	**95.5**	**94.8**

dation problem. We provide spatial and channel-wise attention and also enhances the intra-class compactness and inter-class separability of features. This minimizes the misclassification problem. We incorporate oriented region proposal network for obtaining text proposals. We proposed an efficient recognition module to text spotting. It is hardware-efficient and light-weight in nature. We have demonstrated abundant experimental results on several benchmark datasets to show the efficacy of our network. Our network outperform recent literature in terms of f-measure.

Table 4.12: 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	93.9	**96.6**	**94.4**

Table 4.13: Performance comparison on MSRA-TD500 dataset.

Methods	**Precision**	**Recall**	**F-measure**
DSRN [65]	87.6	71.2	78.5
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
MSR [58]	87.4	76.7	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
PAN [60]	84.4	83.8	82.1
Ours	**88.4**	**87.5**	**87.9**

Table 4.14: Performance comparison on COCO-Text [4] dataset for text detection in scene images.

Methods	**COCO-Text**		
	Precision	**Recall**	**F-measure**
Baseline A [4]	83.7	23.3	36.4
Baseline B [4]	**89.7**	10.7	19.1
Baseline C [4]	18.5	4.7	7.4
EAST [15]	50.4	32.4	39.5
TextBoxes++ [98]	55.8	56	55.9
MaskTextSpotter [104]	66.8	58.3	62.3
LATD [34]	74	51	61
Cheng *et. al.* [45]	60	33	42
Cheng & Wang [130]	48	38	42
Boundary [110]	59	67.7	63
RRD [36]	38	34	36
Yao *et al.* [57]	43.2	27.1	33.3
Lyu *et al.* [35]	69.9	26.2	38.1
WordSup [37]	45.2	30.9	36.8
Ours	**89.7**	**68.1**	**78.2**

Table 4.15: Performance comparison on SVT [5] dataset for text detection 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
Raghunandan *et al.* [42]	60.4	68.7	64.2
Tang & Wu [47]	54.1	75.8	63.1
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]	58.8	76.2	66.4
TextBoxes [97]	67.2	60.8	63.8
Khare *et al.* [44]	41.6	44.3	42.9
Ours	**87.9**	**77.5**	**79.8**

Table 4.16: Performance comparison on ICDAR 2013 datasets for the recognition.

Methods	**ICDAR 2013**					
	End-to-End			**Word-Spotting**		
	strong	**weak**	**generic**	**strong**	**weak**	**generic**
Li *et. al.* [99]	90.2	88.9	84.5	95	93.7	88.7
FOTS [1]	88.8	87.1	80.8	92.7	90.7	83.5
MLTS (spatial attention) [101]	88	85	72	-	-	-
MLTS (channel-wise attention) [101]	87	84	70	-	-	-
He *et. al.* [96]	91	89	86	93	92	87
Deep TextSpotter [94]	89	86	77	92	89	81
TextBoxes++ [98]	93	92	85	96	95	87
TextBoxes [97]	91	89	84	94	92	87
Mask TextSpotter [104]	93.3	91.3	88.2	92.7	91.7	87.7
ASTS (baseline) [107]	90.5	89.2	83.6	94.4	93.1	86.5
ASTS (weakly) [107]	92.8	**91.5**	85.9	95.9	94.7	88.5
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	**94.1**	**91.5**	**88.7**	**96.4**	**94.8**	**88.9**

Table 4.17: Performance comparison on ICDAR 2015 datasets for the recognition.

Methods	ICDAR 2015					
	End-to-End			Word-Spotting		
	strong	weak	generic	strong	weak	generic
Li *et. al.* [99]	90.2	88.9	84.5	95	93.7	88.7
FOTS [1]	88.8	87.1	80.8	92.7	90.7	83.5
MLTS (spatial attention) [101]	88	85	72	-	-	-
MLTS (channel-wise attention) [101]	87	84	70	-	-	-
He *et. al.* [96]	91	89	86	93	92	87
Deep TextSpotter [94]	89	86	77	92	89	81
TextBoxes++ [98]	93	92	85	96	95	87
TextBoxes [97]	91	89	84	94	92	87
Mask TextSpotter [104]	93.3	91.3	88.2	92.7	91.7	87.7
ASTS (baseline) [107]	90.5	89.2	83.6	94.4	93.1	86.5
ASTS (weakly) [107]	92.8	**91.5**	85.9	95.9	94.7	88.5
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	**94.1**	**91.5**	**88.7**	**96.4**	**94.8**	**88.9**

Table 4.18: Performance comparison on COCO-Text [4] and SVT [5] dataset for text recognition in scene images.

Methods	COCO-Text			SVT	
	End-to-End			Spotting	
	strong	weak	generic	None	50
Baseline A [4]	68.4	28.3	40	-	-
Baseline B [4]	54.4	9.9	16.8	-	-
Baseline C [4]	4.1	1.6	2.3	-	-
Mask TextSpotter [104]	65.8	37.3	47.6	-	-
FCRNall + multi-filt [8]	-	-	-	55.7	67.7
TextBoxes [97]	-	-	-	64	84
Jaderberg *et al.* [93]	-	-	-	53	76
Li *et. al.* [95]	-	-	-	66.2	84.9
TextBoxes++ [98]	-	-	-	64	84
Neumann & Matas [131]	-	-	-	-	68.1
Alsharif *et al.* [132]	-	-	-	-	48
Jaderberg *et al.* [92]	-	-	-	-	56
Wang *et al.* [133]	-	-	-	46	
Ours	**70.1**	**65.4**	**69.3**	**67.6**	**85.8**

Table 4.19: Test time speed in terms of on 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	1.15	6.21	**23.4**	S