Chapter 6

NAST dataset for Multilingual Arbitrary-shaped Scene Text Spotting (MAST)

The goal of this chapter is to improvise the Chapters 3, 4, and 5 for arbitrary-shaped scene text detection, multilingual scene text recognition, and script identification as depicted in Figure 6.1. The captured scene images in real-life are of arbitrary-shape, which varies aspect-ratio, size, and scale, even within a word, as shown in Figure 6.2. Further, across the world, within the countries, different scripts are available. Thus in practical application, scene images have multilingual text instances. Therefore, we address the computation of arbitrary-shape text mask and training specifications for multilingual text recognition in this chapter. We design a multilingual arbitrary-shaped scene text spotting mechanism, named as *MAST*, for computing text masks and recognizing multilingual text instances using deep learning architecture.

The rest of the chapter is organized as follows. The first section describes the details for computing text mask and training information for multi-language recognition. The next section elaboration on the experimental results, and finally, we conclude in the

Figure 6.1: Illustration of natural images (first row) representing the necessity of our multilingual text spotter. Second row is the recognized scene text instances by the proposed network.

last section.

6.1 Proposed Architecture

In this work, we consider MobileNetV2 as a backbone network followed by the context aggregation attention module of Chapter 4 prior to add the mask prediction module, as shown in Figure 6.3. The output feature map of the context aggregation attention module is F. We then localize a text instance in a scene image as precisely as possible using a text mask.

6.1.1 Text Mask Computation

We define a *text mask* as a $2(r+1)-$vertex polygon drawn surrounding the text instance as shown in Fig. 6.4, where r is the number of regions. With the increase in the value of r, the precision in localization will increase, but in turn the computational complexity of the network will also increase. Thus, there is a trade-off between the precision and the computational overhead, as illustrated in Figures 6.5. We maintain the threshold value in intersection-over-union (IoU) between an irregular 10-vertex polygonal text mask

Figure 6.2: Illustration of natural images (first row) representing the necessity of our network for detecting curve text instances. Second row is the recognized scene text instances by the proposed network.

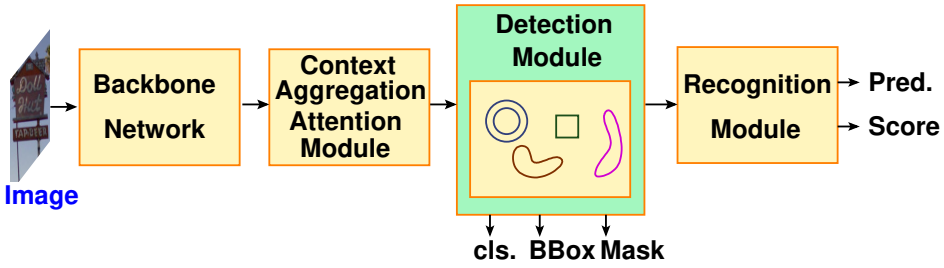
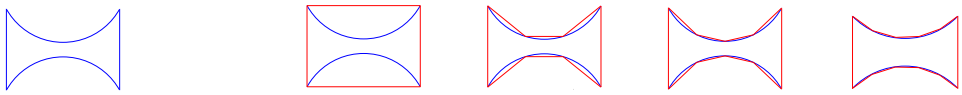


Figure 6.3: Overall architecture of arbitrary shaped text spotter.

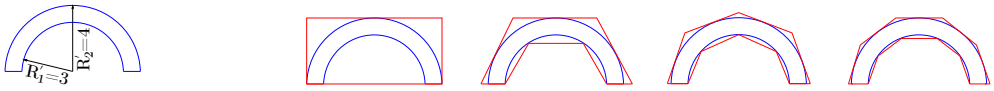
and a human perceived ground truth mask as 0.5. We thus consider $r = 4$ to limit the computational overhead of our network. The attention map $F \in \mathbb{R}^{H \times W \times \zeta}$ is normalized using 1×1 convolution, which is used for text mask prediction and classification.

For a given region window RW_i, the mask prediction module takes the coupled map F as input and reduces the concatenated map to a 15-D vector, as shown in Figure 6.5. The obtained 15-D vector represents the five equally spaced center-coordinates over a text mask with their respective text heights. This helps to address arbitrary-shaped text detection. A text mask targets an intersection between a text instance and its associated ground-truth mask based on the 15-D vector. We avoid any quantization of a mask boundary and rather we prefer to use bilinear interpolation [147] to predict



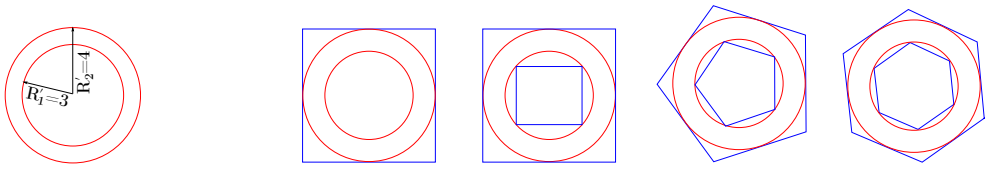
Human Perceived GTM $IoU = 0.42$ $IoU = 0.68$ $IoU = 0.82$ $IoU = 0.88$

Illustration of IoU values of a text mask having $2(r+1)-$vertex polygon with respect to a human perceived ground truth mask (GTM). The value of r is chosen as {1, 3, 4, 5}.



Human Perceived GTM $IoU = 0.34$ $IoU = 0.61$ $IoU = 0.79$ $IoU = 0.86$

Illustration of IoU values of a text mask having $2(r+1)-$vertex polygon with respect to a human perceived ground truth mask (GTM). The value of r is chosen as {1, 3, 4, 5}



Human Perceived GTM $IoU = 0.34$ $IoU = 0.46$ $IoU = 0.59$ $IoU = 0.72$

Illustration of IoU values of a text mask having $2(p+1)-$vertex polygon with respect to human perceived ground truth mask (GTM) with inner and outer radius as R'_1 and R'_2, respectively. The value of p is chosen as {1, 4, 5, 6}.

Figure 6.4: Illustration of IoU values of an arbitrary-shaped text mask.

the precise values of the features at four regularly sampled locations in each bin and aggregate the resultant mask using max-pooling operation. We apply polygonal Non-Maximum Suppression (NMS) [148] on the text mask to obtain the final text masks. The $i-th$ text mask U_i is obtained by:

$$U_i = \begin{cases} U_i & \textbf{iou}(GTM, V_i) < U_t \\ 0 & \textbf{otherwise}, \end{cases} \tag{6.1}$$

where GTM represents the ground truth mask and V_i and U_t are the list of initial text masks and IoU threshold, respectively.

It is concatenated with 2-D classification score. The obtained 17-D vector represents the output for a window in the final output map F. Each pixel in F corresponds to a specific local window. We avoid any quantization of a mask boundary, rather, we prefer to use bilinear interpolation [147] to predict the precise values of the features at four regularly sampled locations in each bin and aggregate the resultant mask using

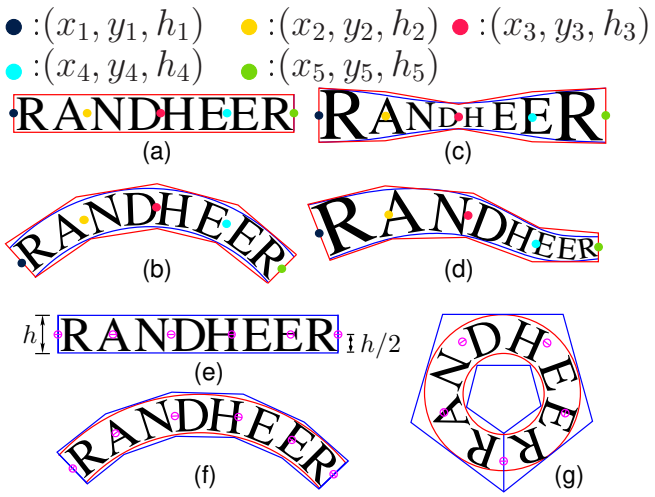


Figure 6.5: Illustration of text mask prediction in mask branch.

max-pooling operation.

6.1.2 Learnable Polygon Non-Maximum Suppression (LP-NMS)

Traditionally, greedy NMS is used as a post-processing step in the literature [15, 17, 18] to eliminate the redundant BBoxes. If the IoU threshold has a higher value then sufficient surrounding detections will not be suppressed followed by the introduction of high scoring false positives and the precision will decrease. If IoU threshold has a low value then multiple true positives will be merged together and as a result the recall value will diminish. Thus, it has a trade-off between precision and recall. Greedy NMS focuses on removing the overlapping detections ignoring whether a detection is a local maxima. To improve the efficiency of detection, we can prune the false positives keeping true positives. Inspired by [149], we design a model to make a soft decision by re-scoring the input detections.

In this work, we propose a *Learnable Polygon NMS* (*LP-NMS*), which is a DNN model for highly precise, accurate, and effective text detection, even in the presence of faint edges. To the best of our knowledge, no work exists in the literature of text detection that uses a DNN for NMS over text masks. Fig. 6.6 shows the architecture

of the *LP-NMS* model. The output map F of our detection network consists of several word masks. LP-NMS takes two maps obtained from F as inputs, namely a score map S and an overlap map M. We modify the intersection and union operations to address text masks (in-place of bounding boxes).

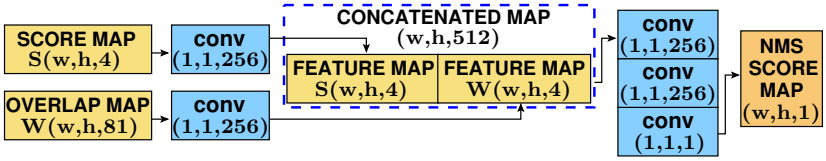


Figure 6.6: The architecture of learnable polygon maximal suppression model. Here, (i, j, z) in a mask represents the resolution (i,j) and depth (z) of the feature map produced in that mask.

- **Score Map:** We consider a neighboring window NW_i of size $l' \times l'$ (we consider $l' = 9$) for each pixel in D. *LP-NMS* on all word masks in this window is performed. Alike greedy NMS, the threshold α is considered as a trade-off parameter between precision and recall in *LP-NMS*. To make our model robust, a range of values for α is taken. The score map $\delta(\alpha)$ is obtained after applying non-maximal suppression at a threshold α in each window of D. The final score map S is obtained by the concatenation of score maps at $\alpha = \{0.25, 0.5, 0.75, 1.0\}$ is $S = \{\delta(0.25)|\delta(0.5)|\delta(0.75)|\delta(1.0)\}$, where $|$ is the concatenation operation.
- **Overlap Map:** The overlapping of two pixels means the overlap of masks at the two pixels. We consider a window of size $l' \times l'$ (we choose $l' = 9$) for each pixel P_i in D. The overlap with P_i for each pixel is computed. The values obtained are encoded as a feature map of depth d', known as overlap map M. It represents the overlap of each pixel with its d' neighbors (including itself). We choose $d' = 81$. The maps S and M are convolved with 1×1 filter and the resulting maps are concatenated. The concatenated map is then convolved with three more layers to obtain the final output map of depth one, as shown in Fig. 6.6. The output map represents the confidence score of each pixel of having a word mask after applying non-maximal suppression. The information provided by S and M will help the

network to decide whether a particular detection score corresponds to a correct detection or should be suppressed by a neighboring detection score.

6.1.3 multilingual Text Recognition

We have improvised the recognition module of Chapter 3 and Chapter 5 to incorporate multilingual text recognition and script identification. We incorporate a position embedding followed by a self-attention mechanism for recognition of text instances, as shown in Figure 6.7. We first apply the position embedding mechanism of [118] on the original input feature tensor (text mask) to obtain the position embedded feature tensor E of shape $(G, U, G+U)$, where $\{G, U\}$ is set to $\{16, 32\}$. The position embedding feature tensor E is calculated by:

$$E'(i, j, :) = \textbf{onehot}(i, U), \tag{6.2}$$

$$E''(i, j, :) = \textbf{onehot}(j, G), \tag{6.3}$$

$$E = \textbf{concat}(E', E''), \tag{6.4}$$

where $\textbf{onehot}(\imath, k)$ indicates a $k-$length vector containing $\imath-$index element with unit value and rest as zero. We aggregate the position embedding feature tensor with the original tensor to obtain aggregate tensor F is of size $(G, U, \zeta + G + U)$, where ζ is the number of channels of the input feature tensor which is set to 128.

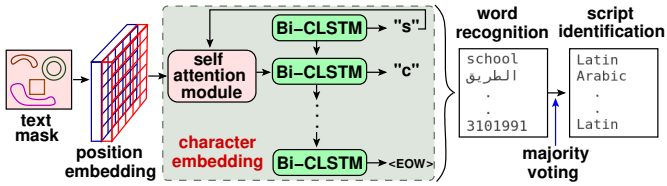


Figure 6.7: Architecture of the proposed recognition module.

Next, we predict a sequence of character classes $\mathbf{y} = (\mathbf{y}_1, \cdots, \mathbf{y}_t, \cdots, \mathbf{y}_T)$ iteratively for T steps. At a step t, the feature tensor is F, the last hidden state is $\mathbf{G}_{t-1}$, and the last predicted character class is $\mathbf{y}_{t-1}$. Unlike [118], we have used C-LSTM [119] to keep

the network light-weight. We expand $\mathbf{G}_{t-1}$ from a vector to a feature tensor $\mathbf{h}_{t-1}$ by copying, where $\mathbf{h}_{t-1}$ is of shape (G, U, V) and V is the hidden size of the recurrent neural network that is set to 128. LSTM accepts an input sequence $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_t, \cdots, \mathbf{x}_T)$, where each of $\mathbf{x}_t$ is a vector corresponding to step t. The output sequence $\mathbf{y}$ is computed by using the following LSTM equations iteratively from $t = \{1, \cdots, T\}$:

$$\mathbf{i}_t = \sigma(\mathfrak{W}_{ix}\mathbf{x}_t + \mathfrak{W}_{iy}\mathbf{y}_{t-1} + \mathfrak{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i), \tag{6.5}$$

$$\mathbf{f}_t = \sigma(\mathfrak{W}_{fx}\mathbf{x}_t + \mathfrak{W}_{fy}\mathbf{y}_{t-1} + \mathfrak{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f), \tag{6.6}$$

$$\mathbf{m}_t = \sigma(\mathfrak{W}_{mx}\mathbf{x}_t + \mathfrak{W}_{my}\mathbf{y}_{t-1} + \mathbf{b}_m), \tag{6.7}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{m}_t \odot \mathbf{i}_t, \tag{6.8}$$

$$\mathbf{o}_t = \sigma(\mathfrak{W}_{ox}\mathbf{x}_t + \mathfrak{W}_{oy}\mathbf{y}_{t-1} + \mathfrak{W}_{oc}\mathbf{c}_t + \mathbf{b}_o), \tag{6.9}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{tanh}(\mathbf{c}_t), \tag{6.10}$$

$$\mathbf{y}_t = \mathfrak{W}_{yh}\mathbf{h}_t, \tag{6.11}$$

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

$$\alpha_t(i, j) = \frac{\mathbf{exp}(\xi_t(i, j))}{\sum_{i'=1}^{G}\sum_{j'=1}^{U}\mathbf{exp}(\xi_t(i', j'))}, \tag{6.12}$$

$$\xi_t = \mathfrak{W}_t \times \mathbf{tanh}(\mathfrak{W}_s \times \mathbf{H}_{t-1} + \mathfrak{W}_f \times F + \mathbf{b}_\varepsilon), \tag{6.13}$$

where ξ_t and α_t are of shape (G, U). $\mathfrak{W}_t$, $\mathfrak{W}_s$, $\mathfrak{W}_f$, and $\mathbf{b}_\varepsilon$ are trainable weights, and

bias, respectively. We then update the glimpse $\mathbf{g}_t$ of step t by imposing the attention weight to the original feature tensor F as follows:

$$\mathbf{g}_t = \sum_{i=1}^{G}\sum_{j=1}^{U} \alpha_t(i,j) \times F(i,j). \tag{6.14}$$

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

$$\mathbf{\Phi}(\mathbf{y}_{t-1}) = \mathfrak{W}_y \times \mathbf{onehot}(\mathbf{y}_{t-1}, \mathbf{k}) + \mathbf{b}_y, \tag{6.15}$$

$$\mathbf{x}_t = \mathbf{concat}(\mathbf{g}_t, \mathbf{\Phi}(\mathbf{y}_{t-1})), \tag{6.16}$$

where $\mathbf{b}$ and $\mathfrak{W}$ are bias and trainable weights of the linear transformation, respectively. In the sequence decoder, $\mathbf{k}$ is the number of classes, which is equivalent to $\wp + \chi + 1$ that includes $\wp = 36$ classes for alphanumeric characters in English (Latin), χ standard symbols, popular traffic signs, and characters from other eight languages, like Chinese, French, Arabic, Italian, Korean, German, Japanese, and Indian, and 1 class for end-of-word (EOW) symbol. The value of χ depends on the character sets in RRC-MLT 2017 dataset. Then, the conditional probability (**CP**) at step t is obtained by a linear transformation and a softmax function as follows:

$$\mathbf{CP}(\mathbf{y}_t) = \mathbf{softmax}(\mathfrak{W}_o \times \mathbf{x}_t + \mathbf{b}_o) \tag{6.17}$$

$$\text{and } \mathbf{y}_t \sim \mathbf{CP}(\mathbf{y}_t). \tag{6.18}$$

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

6.1.4 NAST Dataset Creation

NAST is consists of multilingual texts, the multi-oriented texts, and the wide variety of scenes in terms of content and image resolution. Apart of conventional horizontal and multi-oriented texts, it features curved-oriented text. A curved line is not a straight line. It is free of angle variation restriction throughout the line. NAST is highly diversified in orientations; more than half of its images have a combination of more than two orientations. The dataset is comprised of 9 languages, *i.e.*, Arabic, Bangla, Chinese, English, French, German, Italian, Japanese and Korean. It is a large dataset of noisy scene images that has texts with varying font, scale, orientation, aspect ratio, and script. The images also have varying noise density, low contrast, poor illumination, and faint edges. It contains 951 images for training and another 321 for testing. The size of images varies from 260×183 to 3436×2700. It has horizontal, multi-oriented, and curve text instances with 7592 text annotations. The ground truth is provided at character, word, and text-line-levels. Many images are collected from Google and captured using phone camera. The rest of the images in the dataset are obtained by imposing synthetic fog, rain, and image blurs on the scene images of publicly available datasets like Total-Text 2017 and RRC-MLT 2017. We also vary the contrast and illumination of benchmark images. Every image has at least one text instance per image.

It covers three tasks related to text detection and script classification. First one is detecting multilingual curve texts, second one cropped word script identification, and lastly joint text detection and script identification. We have received a total of 8 participations from the research and industrial communities. In many examples where the network is not confident about the shape of the curved text regions, we prefer to leverage with text line supervision.

6.1.5 Autonomous Learning with GT Generation

A few datasets are available for noisy text detection and recognition. The creation of any new dataset by capturing images of noisy scenes requires the generation of Ground Truth (GT) boxes, which is very laborious. The limited amount of GT box annotated datasets is a hindrance to the learning ability of our network. To address this issue, we propose techniques to make the network learn using automatic GT generation in an image dataset. To the best of our knowledge, this is the first work that generates GT for NAST detection directly at the word level using a convolutional neural network (CNN) and makes the CNN learn stronger features from generated GT. We discuss two ways of learning using the automatic GT generation.

- **Learning using Unsupervised Approach (GT-UL)**: This technique is used to learn from any dataset of images D_U that has no GT for text. The network is pre-trained on the dataset [] and fine-tuned on the dataset is run on D_U. For each image in D_U, a set of BBoxes is generated along with the corresponding text confidence scores. All the BBoxes above a certain confidence score T_U are kept as GT text BBoxes. T_U is chosen to be 0.5 to provide a good balance between precision and recall. The dataset D_U with the set of generated GT is denoted by $D_U^{'}$. When our network is trained on $D_U^{'}$, its performance improves significantly.
- **Learning using Semi-Supervised Approach (GT-SL)**: Some datasets have GT boxes for texts at the character level instead of word level. To obtain GT text BBoxes in such a dataset D_S, in addition to T_U, one more restriction is imposed, such that,

$$\frac{Area(B)}{\sum_i Area(B \cap b_i)} \geq 0.6, \tag{6.19}$$

  where B is a detected text BBox and b_i is a character GT BBox that overlaps with B. After generation of GT BBoxes of D_S, we get the dataset $D_S^{'}$. Our network is then trained on $D_S^{'}$ which further improves its performance.

6.2 Experimental Results

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

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

We use standard stochastic gradient descent with adam optimizer having a weight decay of 0.001 and a momentum of 0.9. The input images are fed in mini-batches of 8 images. In the presence of fainted edges in a scene image, some background textures are very similar to text instance. It is therefore difficult for a network to distinguish. This makes training unbalanced, leading to slow convergence. We use hard a negative mining strategy to suppress them. The training on a dataset is performed into two

stages. For the first stage of training, the negative ratio between the negatives and positives is set to 3:1. It is changed to 6:1 in the second stage of the training. To make our network robust, we choose a multi-scale training scheme [121]. Furthermore, we use python script to impose synthetic fog and rain on the training images of benchmark datasets and fed additionally in the network. This is because the images with ambient noises are very less.

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

6.2.1 Impact of Recognition Module for Word Recognition and Script Identification

In this section we investigate the effect of different branches in recognition modules for different branches for word recognition and script identification, as shown in Table 6.1 and Table 6.2, respectively.

Table 6.1: Effect of different modules of recognition branch over RRC-MLT 2017 and NAST datasets for word recognition.

Position Embedding	Bi-CLSTM	Self-Attention	Word recognition	
			RRC-MLT 2017	NAST
✗	✓	✗	43.6	32.5
✗	✓	✓	53.1	40.3
✓	✓	✗	50.7	39.2
✓	✓	✓	58.8	45.7

Table 6.2: Effect of different modules of recognition branch over RRC-MLT 2017 and NAST datasets for script identification.

Position Embedding	Bi-CLSTM	Self-Attention	Script identification	
			RRC-MLT 2017	NAST
✗	✓	✗	74.8	44.5
✗	✓	✓	81.4	54.1
✓	✓	✗	78.6	50.8
✓	✓	✓	89.7	60.2

6.2.2 Comparison with the state-of-the-art

In this section, we compare the existing methods for Total-Text, CTW1500, RRC-MLT 2017, and RCTW 2017 with our network in terms of both detection and end-to-end recognition. We also verify the script identification for RRC-MLT 2017. Table 6.4 and Table 6.6 show the multilingual text detection and recognition in RRC-MLT 2017 dataset. It is evident from the results that our network can successfully detect and recognize multilingual text instances. Table 6.7 and Table 6.5 depict the detection results of curve text instances. Table 6.8 illustrate the multilingual curve text recognition. It is clear from the results that our network outperforms state-of-the-art literature in terms multilingual curve text detection and recognition.

6.2.3 NAST dataset statistics

We have sketched the statistics of the NAST dataset in terms avarage number of images for each type of instances, variation in orientation, variation in language and different alignments, like horizontal, vertical, curve and wavy.

Table 6.3: Curve text detection performance on Total-Text [6] dataset.

Methods	**Detection**		
	Precision	**Recall**	**F-measure**
EAST [15]	50	36.2	42
SegLink [17]	30.3	23.8	26.7
TextBoxes [97]	47.2	42.5	44.7
Craft [38]	87.6	79.9	83.6
CTD+TLOC [150]	74	71	73
TextField [49]	61.5	65.2	63.3
Mask TTD [54]	79.1	74.5	76.7
Mask TextSpotter [104]	81.8	75.4	78.5
TextSnake [32]	82.7	74.5	78.4
LOMO [68]	**88.6**	75.7	81.6
FOTS [1]	52.3	38	44
MSR [58]	83.8	74.8	79
CSE [109]	81.4	79.1	80.2
PSENet [59]	81.8	75.1	78.3
TextDragon [105]	85.6	75.7	80.2
Ours	88.4	**80.9**	**84.6**

Dataset link: https://drive.google.com/drive/folders/1PdCh2od5lCB- KvP g9PGi3C 2Gehs4reo

6.2.4 Evaluation of Learning using GT Generation

We have developed two new techniques to make our network learn using GT generation on unannotated or weakly annotated datasets. To evaluate the learning techniques (GT-UL and GT-SL), our network is pre-trained on [8] and fine-tuned on NAST dataset. This model is considered as the **Baseline**. Our dataset is used for both unsupervised and semi-supervised learning. For GT-UL, we run the baseline on our dataset without any supervision and then train the model on the learnt GT BBoxes. This new model is denoted by **OurDataset_UL**. In a similar manner, **OurDataset_SL** is obtained by training the baseline with GT-SL on our dataset. We also train the baseline on the existing word level annotations of our dataset as a reference and call the trained model as **OurDataset_UL**. Table 6.9 shows that the performance of the trained models improves significantly over the baseline when tested on NAST dataset. Moreover, the

Table 6.4: Performance comparison on RRC-MLT 2017 dataset for detection of text in scene images.

Methods	**RRC-MLT 2017**		
	Precision	**Recall**	**F-measure**
EAST [15]	62.2	43.8	63.8
E2E-MLT [102]	60.3	40.9	50.1
Enhanced EAST [144]	79.8	57.3	66.7
FOTS MS [1]	81.8	62.3	70.5
TextBoxes++ [98]	79.8	67.3	72.5
SegLink [17]	79.5	60.4	69.5
MaskTextSpotter [118]	72.8	82.8	76.5
Lyu *et al.* [35]	74.3	70.6	72.4
Craft [38]	80.6	68.2	73.9
Huang *et al.* [51]	80	69.8	74.3
Xue *et. al.* [19]	77.7	62.1	69
Mask TTD [54]	79.1	67.5	73.3
Liu *et al.* [145]	75.4	60.3	67.8
LOMO [68]	78.8	60.6	68.5
RRD *et al.* [36]	75.2	50.5	62.8
Fainted TextSpotter	**84.3**	**83.2**	**82.5**

performance of HT-Text, when trained using GT-UL and GT-SL on NAST dataset, is comparable with the performance it would have achieved if it had been trained on the same dataset using manually annotated GT.

6.3 Summary

This chapter elaborates a network for arbitrary-shaped scene text detection, multilingual scene text recognition, and script identification. It also discusses the statistics of the proposed NAST dataset, which covers both horizontal, vertical, oriented, and curve text detection and multi-language text recognition with script identification. The architecture can be further integrated with sophisticated models for more precise spotting.

Table 6.5: Performance comparison on RCTW 2017 datasets for detection of text in scene images.

Methods	**RCTW 2017**		
	Precision	**Re.call**	**F-measure**
EAST [15]	59.7	47.8	53.1
E2E-MLT [102]	57.5	38.9	48.2
Enhanced EAST [144]	76.4	53.2	64.8
FOTS MS [1]	79.5	62.3	67.2
TextBoxes++ [98]	75.3	63.5	68.1
SegLink [17]	77.2	58.9	68.0
MaskTextSpotter [118]	70.5	58.6	64.5
Lyu *et al.* [35]	71.8	63.3	67.5
Craft [38]	77.2	61.1	68.4
Huang *et al.* [51]	73.6	62.5	67.5
Xue *et. al.* [19]	74.9	60.5	67.7
Mask TTD [54]	77.2	64.3	70.2
Liu *et al.* [145]	71.9	55.3	62.5
LOMO [68]	**80.4**	50.8	62.3
RRD *et al.* [36]	72.4	45.3	55.7
Fainted TextSpotter	80.2	**65.4**	**73.1**

Table 6.6: Performance comparison on RRC-MLT 2017 dataset for text recognition and script identification in scene images.

Methods	**Word Spotting**	**End-to-End**	**Script Identification**
	F-measure	**F-measure**	**F-measure**
E2E-MLT [102]	**65.1**	48	-
Enhanced EAST [144]	58.3	47.3	-
FOTS [1]	57.9	48.9	-
FOTS MS [1]	59.6	52.5	-
TH-DL [12]	58.3	39.3	80.7
SCUT DLVClab [12]	61.7	58	87.6
CNN-based [12]	64.2	54.6	88.1
BLCT [12]	60.5	52.5	86.3
Synthetic-ECN [12]	51.8	46.2	79.2
Ours	**65.1**	**58.8**	**89.7**

Table 6.7: Performance comparison on SCUT-CTW1500 [7] for curve text detection in scene images.

	SCUT-CTW1500		
	Precision	**Recall**	**F-measure**
EAST [15]	78.7	49.1	60.4
SegLink [17]	42.3	40	40.8
CTPN [151]	60.4	53.8	56.9
Craft [38]	86	81.1	83.5
CTD+TLOC [150]	77.4	69.8	73.4
TextField [49]	83	79.8	81.4
Huang *et al.* [51]	86.8	83.2	85
Mask TTD [54]	79.7	79	79.4
TextSnake [32]	67.9	85.3	75.6
LOMO [68]	89.2	69.6	78.4
LOMO MS [68]	85.7	76.5	80.8
Wang *et al.* [53]	80.1	80.2	80.1
TextCohesion [152]	88	84.7	86.3
Tian *et. al.* [24]	82.7	77.8	80.1
MSR [58]	85	78.3	81.5
CSE [109]	81.1	76	78.4
PSENet [59]	80.6	75.6	78

Table 6.8: Curve text recognition performance on Total-Text [6] dataset.

Methods	**End-to-End**	
	None	**Full**
TextBoxes [97]	36.3	48.9
Mask TextSpotter [104]	65.3	77.4
Qin *et al.* [146]	70.7	-
FOTS [1]	32.2	35.9
Boundary [110]	65.0	76.1
ABCNet [90]	69.5	78.4
Text Perceptron [112]	69.7	78.3
CharNet [59]	69.2	-
TextDragon [105]	48.3	74.7
Ours	**65.9**	**77.8**

Table 6.9: Performance on Our Dataset with GT Learning.

Method	**Precision**	**Recall**	**F-measure**
Baseline	0.8204	0.8521	0.8360
OurDataset_US	0.8312	0.8634	0.8470
OurDataset_SS	0.8344	0.8647	0.8493
OurDataset_GT	0.8355	0.8660	0.8504

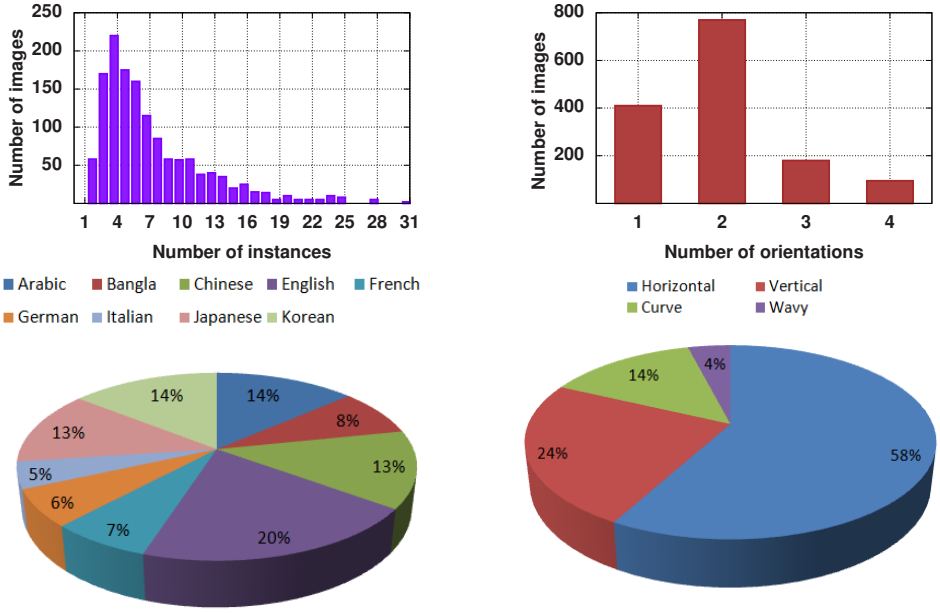


Figure 6.8: Statistics of NAST dataset.