2018 24th International Conference on Pattern Recognition (ICPR)
Beijing, China, August 20-24, 2018

An Efficient System for Hazy Scene Text Detection using a Deep CNN and Patch-NMS

Sabyasachi Mohanty, Tanima Dutta and Hari Prabhat Gupta
Dept. of Computer Science and Engineering, IIT (BHU) Varanasi, India

***Abstract*—Scene text detection systems detect texts in natural scene images. Hazy scene text detection is a specific case of scene text detection where detection is done in hazy weather conditions. Haze affects the contrast of the image. In this paper, we reframe the traditional two class hazy scene text detection problem into a four class problem. We develop a deep learning based model that combines features from all layers for accurate and fast text detection from hazy images. In addition, we develop a novel training approach for the four class problem. Merging and patch-NMS are used as post processing steps for fast word detection. We also create a new dataset of hazy scene images and obtain significant improvements on an existing hazy scene text dataset.**

I. Introduction

Text detection in natural scene images is an important research problem in computer vision and pattern recognition communities. Scene text detection refers to the task of determining the precise location of all text occurrences in an image of a natural scene and enclosing them with a bounding box. This task is challenging because texts have a wide range of size, color, orientation, aspect ratio, and font. Moreover, the background objects present in scene images often resemble texts closely, like bricks, leaves, and windows. Scene text detection has many applications such as image tagging, indoor navigation, and robotic vision.

In scene text detection, very often, the scenes have good lighting condition and scene texts have high resolution and contrast with the background objects. *Hazy scene text detection* (*scene text detection in hazy images*) is a specific case of scene text detection where text detection is done in scene images with hazy weather condition. Hazy scene text detection has its wide application in Driving Assistance System (DAS) [1]. An intelligent DAS needs to detect texts in traffic panels, lane markings, and milestones accurately and fast to make driving safe. Haze reduces contrast in scene images, making it difficult to process and obtain useful information from hazy images.

Most works [2]–[4] on the study of hazy images have been done as a way to reduce the impact of haze (dehazing) on an image. These works do not aim at performing a specific task on hazy image, such as text or face detection. These tasks are, mostly, performed as post-processing after dehazing. Therefore, hazy scene text detection is still in its rudimentary phase. To the best of our knowledge, the only existing method for the task of hazy scene text detection is described in [1], where authors first enhance the contrast of a hazy scene image and then perform scene text detection. It is to be noted that the text detection technique used in [1] is applicable to any image and is not specific to hazy images. Therefore, as per our knowledge, no work exists in the state-of-the-art literature that detects texts specifically for a hazy scene image. In this paper, we develop a novel system that observes the hazy scene text detection task as a unified problem. It does not separate dehazing from text detection. The traditional hazy scene text detection task is seen as a two class problem. In our system, we reframe it as a four class problem.

Deep learning based approaches [5]–[8] have significantly improved the performance for the tasks of generic object recognition and detection. A recent trend in generic object [7], [8] and scene text detection [9]–[11] is to use feature maps of different layers to make the final detection. Most works [8], [9], [11] combine the maps of different layers in various ways. However, the features are combined from specific layers and hence, miss out on important features from the remaining layers. Therefore to solve this issue, in this paper, we design an effective and fast deep learning based system which combines the maps of all layers, leading to very high detection accuracy without affecting the detection speed for hazy scene text detection. Most state-of-the-art scene text detectors [12]–[14] use traditional Greedy Non-Maximum Suppression (G-NMS) as a post-processing step to obtain the final detections, which is slow. In our system, we use two post-processing steps: merging and patch-NMS (P-NMS) to detect final word level bounding boxes very fast.

Character level detection approaches are often followed by scene text detection methods [13]–[15]. However, character level detection requires several post-processing stages which leads to error accumulation and poor text detection performance [16]. To overcome this problem, our model detects text regions directly at the word level.

The major contributions of this paper are summarized as follows:

- We visualize the hazy scene text detection task in a new way where the detection is performed as a unified task of learning features of haze and text simultaneously by viewing the task as a four class problem.
- We develop an efficient and fast CNN model (HT-Net) for hazy scene text detection that uses features from all layers to detect word boxes. To the best of our knowledge, no work exists in the literature that uses deep learning for the task of hazy scene text detection.
- We develop two new post-processing steps to obtain the final text bounding boxes at the word level. One of these steps (P-NMS) uses an optimized version of G-NMS.
- We develop our own dataset of hazy scene images with text regions. This dataset contains texts in a variety of color, font, scale, aspect ratio, orientation, and script to verify the robustness of our model. We achieve high accuracy and efficiency on this dataset as well as the dataset of [1].

978-1-5386-3787-6/18/$31.00 ©2018 IEEE

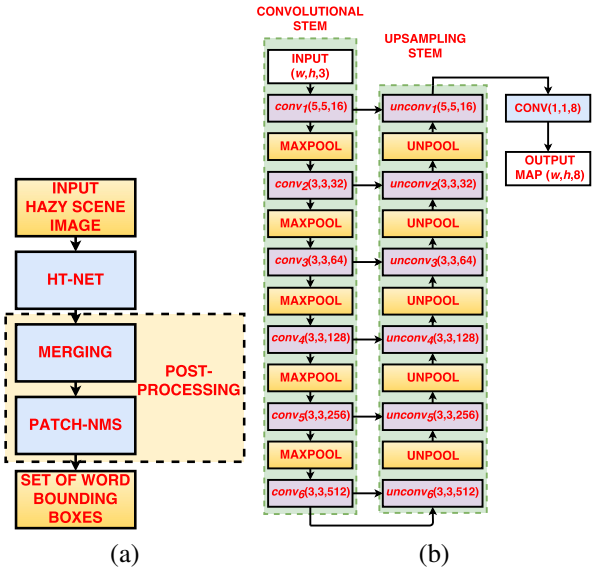

Fig. 1. Part (a) shows the block diagram of the proposed system. Part (b) depicts the architecture of HT-Net. Here, (x, y, z) in a block represents the resolution (x,y) and depth (z) of the feature map produced in that block. (w, h) is the resolution of the input hazy image.

The rest of the paper is organized as follows: The next section describes the proposed methodology for scene text detection. Section III illustrates the experimental results and we conclude the paper in Section IV.

II. Proposed Methodology

In this section, we describe the the challenging problem of scene text detection in hazy images. Fig. 1(a) depicts the block diagram of our system.

A. Class Determination for Hazy Scene Text Detection

• Scene Text Detection: Text detection is a challenging problem in natural scene images as text regions have lots of variations in terms of font, size, color, and orientation. However, most text detection datasets have many instances of texts that have either high contrast compared to background, are not occluded by other objects, or are not dominated by the background. These three properties of scene text images make text detection a two class (text and background) problem since a CNN detector learns strong convolutional features of texts that are highly discriminative. Moreover, the convolutional features of background are much different from texts. In short, the features of texts and background are very different and there is a distinct boundary separating the two classes.

• Scene Text Detection in Hazy Images: Hazy images of natural scenes are dominated by haze. The density of haze also varies across an image as well as across the whole dataset. In such a dataset, such as [1] and our dataset, texts are often occluded by haze. Moreover, the images have very low contrast due to the presence of haze, which makes it difficult to distinguish texts from haze as well as background. As a result, a CNN detector cannot perform well in learning strong convolutional features of texts. In addition, convolutional features obtained from text areas having haze with varying density encodes information of both texts and haze. This confuses the detector and reduces detection accuracy. It is to be noted that the main challenge is to address the fine margin between the learnt features of texts, haze, and regions containing both.

To address this issue, we propose a robust system for scene text detection in hazy images. It addresses hazy scene text detection as a four class problem, denoted as follows:

Haze class (H): All the areas in an image containing haze and no text fall in this class. This is the dominant class in a hazy image dataset.

Text class (T): Text class refers to the text instances that are not covered with haze.

Haze+Text class (HT): The text instances in a scene image covered with haze fall in this category.

Background class (B): All objects in the image dataset that do not belong to the above three classes belongs to the B class.

B. Architecture of HT-Net

In this section, we propose a fully convolutional network, referred by HT-Net, to detect text regions in hazy images. Fig. 1(b) shows the architecture of HT-Net. In a deep CNN, the following issues are commonly observed:

• The feature maps in the layers closer to the input image (early layers) of a deep CNN capture less semantic information, while strong semantic information is captured by the maps in the deeper layers.

• The deeper layers of a deep CNN have low resolution feature maps which fail to detect very small text instances whereas maps in early layers are better capable of detecting small texts.

• The feature maps obtained after repeated sampling (both downsampling and upsampling), have activations which localize text objects poorly.

The proposed HT-Net address these issues. The architecture of HT-Net has three different components to tackle each of these issues, which are described as follows:

(1) **Convolutional Stem**: The convolutional stem is the feed forward part of the network. It has six *stages* ($conv_1$ to $conv_6$). Each stage is a set of consecutive layers producing feature maps of the same resolution. Each stage is separated from the other by a maxpool layer. The feature map in each layer of a given stage has half the resolution and double the depth of a feature map in the previous stage. The stages $conv_1$ and $conv_6$ have a single layer each, while all other stages have two layers each. The feature maps in the early stages capture low level features while the deep feature maps capture features with very strong semantic information about the four classes.

(2) **Upsampling Stem**: The upsampling stem is designed to increase the resolution of the deep feature maps of the convolutional stem. This is necessary for the detection of small instances of texts in the input image. Upsampling is performed using a 2×2 unpool operation. The upsampling stem also has six stages ($unconv_1$ to $unconv_6$). A feature map in a specific layer of a specific stage of the upsampling stem has the same

resolution and depth as the map in the corresponding layer of the corresponding stage of the convolutional stem.
(3) **Lateral Connections**: The activations of the feature maps in the upsampling stem localize poorly as they have gone sampling many times whereas the activations of the maps in the convolutional stem have good localization since they have undergone sampling only a limited times. The lateral connections exist between corresponding stages of the convolutional and upsampling stems. These connections aim to enhance the strong semantic information present in feature maps of the upsampling stem with good localization ability, which is provided by maps of the convolutional stem. Each stage has a single lateral connection where the feature map of the last layer of that stage of the convolutional stem is connected to the corresponding feature map in the upsampling stem. In a lateral connection, two feature maps are concatenated and a 1×1 filter is used to reduce the depth of the resulting map. The feature map at the top of the upsampling stem has the maximum resolution, semantic information, and also good localization ability. We choose this as the final feature map for detection.

The output map is obtained by performing convolution (CONV) on the final feature map using a 1×1 filter. The output map has a depth of 8 channels. For a pixel P_i in the output map, 8 channels represent three different predictions, where first four channels predict the 2 central coordinates, height, and width of the bounding box detected by P_i, fifth channel predicts the confidence score of the bounding box of containing an object irrespective of class and the final three channels give the confidence score of the box containing an object of the classes T, HT, and H, respectively.

C. Training Details of HT-Net

• MULTI-TASK LOSS FUNCTION: HT-Net performs three tasks simultaneously, *i.e.*, detection of Bounding Boxes, assigning a class-agnostic confidence score to each box, and assigning a class-specific confidence score to each box for each of the three classes (H, HT, and T). Therefore, we design a multi-task loss function which is defined as follows:

$$L_{mul} = L_{cls1} + \lambda_1 L_{cls2} + \lambda_2 L_{reg}, \tag{1}$$

where L_{cls1} is the class agnostic loss function, L_{cls1} is the total loss associated with the predicted confidence score for the three classes, and L_{reg} is the regression loss associated with the predicted bounding box dimensions. The values of λ_1 and λ_2 are both set to 1 to give equal weight to each loss. Let a pixel P_i in the final output map predict the coordinates, height, and width of the bounding box, which are encoded as a 4-dimensional vector $\hat{z}_i$; the class agnostic confidence score as t_i; and the class specific scores as t_{iH}, t_{iHT}, and t_{iT} for the classes H, HT, and T, respectively. The pixel P_i is said to have true class label *l*, where $l \in \{H, HT, T\}$ if the box predicted by it has greater than 0.5 overlap with a Ground Truth (GT) box of class l. If the predicted box has more than 0.5 overlap with GT boxes of multiple classes, then the class of the GT box with which it has maximum overlap is its true label. For pixel P_i, true labels for the three classes are denoted by t^*_{ij} where $j \in \{H, HT, T\}$ and $t^*_{ij} = 1$ only if j is the class to which the predicted box at P_i belongs, else $t^*_{ij} = 0$. Similarly, $t^*_i = 1$ if the box at P_i belongs to any class, else $t^*_i = 0$. We define the various losses as follows:

$$L_{cls1} = \sum_{i=1}^{res} l_{cls1}(t_i, t^*_i), \tag{2}$$

$$L_{cls2} = \sum_{i=1}^{res} \sum_{j\in\{H,HT,T\}} l_{cls2j}(t_{ij}, t^*_{ij}), \tag{3}$$

$$L_{reg} = \sum_{i=1}^{res} l_{reg}(\hat{z}_i, \hat{z}^*_i), \tag{4}$$

where res is the total number of pixels in the output feature map. l_{cls1} is the class-agnostic classification loss associated with each detected box. l_{cls2H}, l_{cls2HT}, and l_{cls2T} are the classification losses associated with the bounding box of the classes H, HT, and T, respectively. l_{cls1}, l_{cls2H}, l_{cls2HT}, and l_{cls2T} are chosen as cross-entropy loss function [12] to facilitate easier training. We choose $l_{reg\,H}$, $l_{reg\,HT}$, and $l_{reg\,T}$ as the smooth $L1$ loss function [11] as it is less sensitive to outliers. It is to be noted that the regression loss is considered only for those pixels for which $t^*_i = 1$. Also, not all the training samples for H class contribute to the loss function.
• OTHER TRAINING DETAILS: We use the SynthText [17] dataset to pre-train HT-Net which is then fine-tuned on the real dataset on which it is to be tested. Standard stochastic gradient descent [11], [12] is used for learning with a momentum of 0.85. Input images are fed to HT-Net in mini batches where each mini batch has a total of 16 images. We can train our model with inputs of any size, since it is fully convolutional. A multi-scale training scheme is chosen in training our model following the work in [5].
• SELECTION OF H CLASS SAMPLES (BOXES) FOR TRAINING: In all the hazy text detection datasets used by us, the H class constitutes most of the training samples. This leads to slow convergence as training becomes unbalanced. To remedy this, we limit the number of H class samples used for training. Only the samples of H class that have maximum contribution to L_{mul} are chosen such that the ratio between H and T class samples does not exceed 3:1. Each training sample of H class has a mask which is set only if that particular sample is selected for training, otherwise the mask is unset and the sample has no contribution to the loss function.

D. Merging and P-NMS

★ MERGING: HT-Net gives a set of Bounding Boxes (BBs) as the output with the class for each BB. Since the amount of haze in a hazy image is often unevenly distributed, the detected BBs do not enclose whole words. It is observed that on many occasions, for a single word a part of it is detected as a BB of HT class due to large haze density in that part of the word, while the other parts of the word are detected as BBs of T class. To address this issue, we propose a merging technique where the HT class BBs are merged with surrounding T class BBs, as shown in Fig. 2.

For an input hazy image I, the set of T and HT class BBs are denoted by S_T and S_{HT}, respectively. Algorithm 1 describes the merging process where NEIGH(i) denotes the set of T class BBs in a 15×15 neighborhood of a HT class BB (denoted by i). The set of T class BBs in NEIGH(i) occurring in a direction opposite to the direction in which the T class BB (denoted by j) exists with respect to the HT class BB (denoted by i) is given by OPPOSITE(i,j). The function MAXSCORE(i,NEIGH(i)) gives the highest similarity score between a HT class BB (denoted by i) and NEIGH(i) while MAXBOXINDEX((i),NEIGH(i)) gives the T class BB with which i has the maximum similarity.

The function CHECK(i) checks whether the similarity score i is sufficiently high or not and returns a boolean value accordingly. This similarity score is obtained based on how similar a HT class BB (denoted by i) and a T class BB (denoted by j) are, in terms of their class-agnostic confidence score, scale, aspect ratio, and the spatial distance between them. MERGE merges two or three BBs depending on the number of parameters passed to it and gives a new BB. The important observations made from Algorithm 1 are as follows:

- A HT class BB only looks for a maximum of two T class BBs in its neighborhood. This is because a single word can grow atmost in two directions, even though other words may be close to it.
- A BB formed by merging HT class BB with one or two T class BBs is assigned the class T as other fragments of the same word may have been classified as HT class and they will merge with this newly created T class BB eventually.
- Finally, all the BBs belonging to T and HT classes are assigned to a common Final Text (FT) class since after the merging process is complete, all the BBs lose their identity in terms of T or HT and belong to a single class.

Algorithm 1: ALGORITHM FOR MERGING

Input: I, S_T, S_{HT}
Output: S_{FT}

$S_T = \emptyset$, $S_{FT} = \emptyset$
foreach $i \in S_{HT}$ **do** // test and merge HT class BB with nearby T class BB
  γ_{1i} =MAXSCORE(i,NEIGH(i))
  δ_{1i} =MAXBOXINDEX(i,NEIGH(i))
  if (CHECK(γ_{1i})==*true*) **then**
    γ_{2i} =MAXSCORE(i,OPPOSITE(δ_{1i}))
    δ_{2i} =MAXBOXINDEX(i,OPPOSITE(i,δ_{1i}))
    if (CHECK(γ_{2i})==*true*) **then**
      k =MERGE(i,δ_{1i},δ_{2i});
    else
      k =MERGE(i,δ_{1i});
    $S_T = S_T \cup \{k\}$
foreach $i \in (S_T \cup S_{HT})$ **do** // Assign all HT and T class BB to FT class
  $S_{FT} = S_{FT} \cup \{i\}$

★ P-NMS: After merging, all the BBs of the FT class are made to undergo suppression to eliminate the redundant BBs. Most scene text detection algorithms use G-NMS [12]–[14] for this purpose. However, G-NMS is quite slow with a time complexity of n^2, where n is the final output map resolution. For fast processing of BBs, we use P-NMS which is based on the observation that most of the redundant BBs for a particular word occur in neighboring regions (patches). A BB in a given patch rarely has significant overlap with BBs in a different patch. A patch size of 25×25 was found to give the best performance in terms of f-measure on the dataset [1]. Let S_i be the set of all the BBs in patch i of a hazy image I and S_{patch} be the set of all such S_i. Algorithm 2 describes the P-NMS process. GNMS(S_i) performs G-NMS on the set S_i. S_I is the set of the final BBs output by the proposed system.

Algorithm 2: ALGORITHM FOR P-NMS

Input: I, S_{patch}
Output: S_I

$S_{tmp2} = \emptyset$
foreach $S_i \in S_{patch}$ **do** // perform G-NMS in each patch
  S_{tmp1} =GNMS(S_i)
  $S_{tmp2} = S_{tmp2} \cup S_{temp1}$
S_I =GNMS(S_{tmp2})

(a) (b) (c)

Fig. 2. Illustration of merging where blue, green, and yellow BBs represent T, HT, and H classes, respectively. Columns (a), (b), and (c) show the input image, detected bounding boxes before and after merging, respectively.

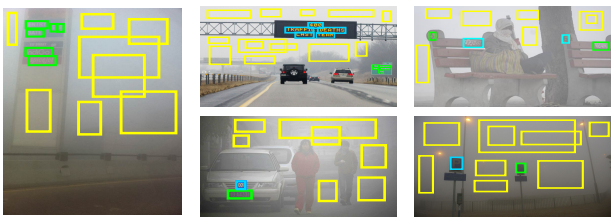

Fig. 3. Illustration of the GT text BBs in various hazy images. In each image, the blue, yellow, and green BBs represent T, HT, and H classes, respectively.

III. EXPERIMENTAL RESULTS

We compare the performance of HT-Net with [1]. However, authors in [1] do not use any deep learning based approach, instead a set of hand crafted features is used for text detection and word formation. Therefore, to truly evaluate the performance of our deep model (HT-Net), we also compare its hazy text detection accuracy and speed with several state-of-the-art deep learning based scene text detectors designed to be operated in natural images, which may not necessarily be hazy.

TABLE I
PERFORMANCE COMPARISON ON HAZY SCENE TEXT DATASETS

Dataset	Our Hazy Scene Text Data			Dataset of Animesh *et al.* [1]			Test Time Speed		
Method	**Precision**	**Recall**	**f-measure**	**Precision**	**Recall**	**f-measure**	**Resolution**	**Device**	**fps**
Proposed	**0.82**	**0.85**	**0.83**	**0.80**	**0.78**	**0.79**	MS	Titan X	13.52
Animesh et al. [1]	0.69	0.78	0.73	0.65	0.62	0.63	MS	-	0.55
He et al. [10]	0.75	0.76	0.75	0.72	0.68	0.70	MS	Titan X	1.11
DMPNet [18]	0.70	0.66	0.68	0.70	0.63	0.66	720p	Titan X	0.95
EAST [12]	0.73	0.75	0.74	0.72	0.65	0.68	720p	Titan X	**16.80**
SegLink [11]	0.71	0.77	0.74	0.71	0.70	0.70	720p	Titan X	8.90
Zhang et al. [9]	0.64	0.62	0.63	0.58	0.59	0.58	MS	Titan X	0.48
Tian et al. [19]	0.65	0.70	0.67	0.61	0.58	0.59	MS	GPU	7.14

We compare HT-Net with the state-of-the-art methods [9]–[12], [18], [19], in addition to the work in [1] and our own dataset. The datasets used for performance comparison as well as training are the SynthText dataset [17], the dataset of Animesh *et al.*, and our hazy scene text dataset. More information about SynthText dataset and the dataset of Animesh *et al.* can be obtained from the works in [17] and [1], respectively. We describe our hazy scene text dataset in details.

Our Hazy Scene Text Dataset: We have created a large dataset of hazy images with scene texts keeping in mind the variations that texts can show in terms of font, size, orientation, aspect ratio, distortion as well as the impact of blurring and occlusion on text instances. Moreover, this dataset contains varying density of haze in its images. This dataset contains 600 images for training and another 400 for testing. We provide the GT at word level.

In our dataset, it is easy to manually annotate GT boxes for T and HT classes. However, it is difficult to manually annotate instances of haze. Therefore, we manually segment regions of images containing haze. Then, we randomly generate BBs of different scale and aspect ratio inside this segmented region. All these generated BBs act as GT for H class. The number of boxes generated is proportional to the area of the segmented region. Fig. 3 shows various hazy images with the GT BBs.

A. Implementation Details

We pre-train HT-Net on the SynthText dataset [17]. Our model is implemented using NVIDIA Titan X graphic card with an Intel E5-2670v3 CPU running at 2.30 GHz. HT-Net is fully convolution, hence it accepts inputs of all sizes

B. Text Detection Performance Comparison

We compare the performance of our system with state-of-the-art scene text detection methods [9]–[12], [18], [19] in addition to [1]. The performance of HT-Net on various datasets in terms of precision, recall, and f-measure is described as follows:

Dataset [1]: Table I shows that our system achieves the best performance in terms of precision, recall, and f-measure. This illustrates that our model is very robust as the dataset is rich in text instances of different fonts, sizes, scripts, and orientation. We have achieved significant improvement over [1] and state-of-the-art scene text detection methods [9]–[12], [18], [19].

TABLE II
EVALUATION OF HAZY SCENE TEXT DETECTION AS A FOUR CLASS PROBLEM ON THE DATASET OF ANIEMESH *et al.* [1]

Method	Precision	Recall	f-measure
HT-Net	**0.8204**	**0.8521**	**0.8360**
Model-2	0.7125	0.6741	0.6928
Model-3	0.7620	0.7486	0.7552

This is because of considering hazy scene text detection as a four class problem instead of a binary class problem.

Our Dataset: Our system achieves much better performance than the state-of-the-art deep learning based methods for scene text detection, as shown in Table I. It can be concluded from the results that the task of hazy text detection is very different from scene text detection. As a result, state-of-the-art methods for scene text detection are not very effective for hazy images.

C. Test Time Speed of HT-Net

Table I demonstrates the test time speed of HT-Net when compared with other state-of-the-art scene text detection methods. We have reported the average *frames per second* (fps) of running our system on our scene text dataset of hazy images. The other methods are also run on our dataset with the input resolution as shown in Table I, where MS stands for Multi Scale. The results suggest that our system is very fast and its speed is comparable with the fastest state-of-the-art method [12]. Note that [1] does not use deep learning, instead it uses many hand crafted features and a cascade of different stages, making it quite slow for real-time use. Please refer the supplementary file for a detailed analysis of the time complexity of HT-Net along with several other experiments.

D. Evaluation of Hazy Text Detection as a Four Class Problem

In order to evaluate the effect of the number of classes on the task of hazy scene text detection, we develop two new models trained with two and three classes and call them as Model-2 and Model-3, respectively. The training and implementation details of these two models are exactly the same as that of HT-Net, only difference being the number of classes they are trained to predict. Table. II shows the effectiveness of visualizing the problem as a four class problem when tested on the dataset of Animesh *et al.* [1]. Our HT-Net achieves much higher f-measure as compared to the other two models.

Fig. 4. The first row illustrates the results of hazy scene text detection on dataset [1], whereas the second and third rows illustrate the results of hazy scene text detection on our dataset.

TABLE III
PERFORMANCE OF DIFFERENT POST-PROCESSING STEPS ON THE DATASET OF ANIEMESH *et al.* [1]

Method	Precision	Recall	f-measure
(M+P-NMS)	**0.82**	**0.85**	**0.83**
Only M	0.77	0.78	0.77
Only P-NMS	0.79	0.81	0.80
Only G-NMS	0.82	0.84	0.83

E. Evaluation of Merging and Patch-NMS

In this section, we compare the performance of merging and P-NMS techniques as compared to post-processing done without merging and with G-NMS. For all the comparisons, we use HT-Net as the baseline for extracting the T class and HT class bounding boxes. The dataset of Animesh *et al.* [1] is used for all the experiments in this section. For post processing, various combinations of Merging (M), G-NMS, and P-NMS are used. As observed from Table. III, the combination of (M + P-NMS) gives performance comparable with the performance obtained by using only G-NMS as the post-processing step (as done in most scene text detection approaches). However, the use of (M + P-NMS) as a post-processing is much faster than the use of G-NMS.

IV. CONCLUSION

In this paper, we reframe the challenging problem of hazy scene text detection into a four class problem, diverging from the binary class problem of traditional hazy scene text detection. We design HT-Net that detects hazy scene texts with high accuracy and speed by using features from all the layers of HT-Net. We also develop merging and P-NMS algorithms as post-processing steps to obtain final word level boxes fast.

REFERENCES

[1] C. Animesh and et al., "Fast text detection from single hazy image using smart device," in *Proc. of ICME Workshops*, 2017, pp. 423–428.
[2] Y. Schechner and et al., "Instant dehazing of images using polarization," in *Proc. of CVPR*, 2001, pp. 325–332.
[3] S. Nayar and S. Narasimhan, "Vision in bad weather," in *Proc. of ICCV*, 1999, pp. 820–827.
[4] K. He and et al., "Single Image Haze Removal using Dark Channel Prior," in *Proc. of CVPR*, 2009, pp. 1956–1963.
[5] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proc. of CVPR*, 2017, pp. 6517–6525.
[6] S. Ren and *et al.*, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.
[7] W. Liu and *et al.*, "SSD: Single Shot MultiBox Detector," *CoRR*, vol. abs/1512.02325, 2015.
[8] T. Kong and *et al.*, "HyperNet: Towards Accurate Region Proposal Generation and Joint Object Detection," in *Proc. of CVPR*, 2016, pp. 845–853.
[9] Z. Zhang and *et al.*, "Multi-oriented Text Detection with Fully Convolutional Networks," in *Proc. of CVPR*, 2016, pp. 4159–4167.
[10] W. He and *et al.*, "Deep Direct Regression for Multi-oriented Scene Text Detection," in *Proc. of ICCV*, 2017, pp. 745–753.
[11] B. Shi, X. Bai, and S. Belongie, "Detecting Oriented Text in Natural Images by Linking Segments," in *Proc. of CVPR*, 2017, pp. 3482–3490.
[12] X. Zhou and *et al.*, "EAST: An Efficient and Accurate Scene Text Detector," in *Proc. of CVPR*, 2017, pp. 2642–2651.
[13] H. Hu and *et al.*, "WordSup: Exploiting Word Annotations for Character Based Text Detection," in *Proc. of ICCV*, 2017, pp. 4950–4959.
[14] S. Tian, S. Lu, and C. Li, "WeText: Scene Text Detection under Weak Supervision," in *Proc. of ICCV*, 2017, pp. 1501–1509.
[15] S. Mohanty, T. Dutta, and H. P. Gupta, "Text preserving animation generation using smart device," in *Proc. of ICME*, 2017, pp. 1039–1044.
[16] S. Tian and *et al.*, "Text Flow: A Unified Text Detection System in Natural Scene Images," in *Proc. of ICCV*, 2015, pp. 4651–4659.
[17] A. Gupta and *et al.*, "Synthetic Data for Text Localisation in Natural Images," in *Proc. of CVPR*, 2016, pp. 2315–2324.
[18] Y. Liu and L. Jin, "Deep Matching Prior Network: Toward Tighter Multi-oriented Text Detection," in *Proc. of CVPR*, 2017, pp. 3454–3461.
[19] Z. Tian and et al., "Detecting Text in Natural Image with Connectionist Text Proposal Network," in *Proc. of ECCV*. Springer, 2016, pp. 56–72.