QEST: Quantized and Efficient Scene Text Detector using Deep Learning

KANAK MANJARI, School of Computer Science Engineering and Technology, Bennett University, Greater Noida
MADHUSHI VERMA, School of Computer Science Engineering and Technology, Bennett University, Greater Noida
GAURAV SINGAL, Department of Computer Science and Engineering, Netaji Subhas University of Technology, Delhi
SUYEL NAMASUDRA*, Department of Computer Science and Engineering, National Institute of Technology Patna, Bihar, India; Universidad Internacional de La Rioja, Logroño, Spain

Scene text detection is a complicated and one of the most challenging tasks due to different environmental restrictions, such as illuminations, lighting conditions, tiny and curved texts, and many more. Most of the works on scene text detection have overlooked the primary goal of increasing model accuracy and efficiency, resulting in heavy-weight models that require more processing resources. A novel lightweight model has been developed in this paper to improve the accuracy and efficiency of scene text detection. The proposed model relies on ResNet50 and MobileNetV2 as backbones with quantization used to make the resulting model lightweight. During quantization, the precision has been changed from float32 to float16 and int8 for making the model lightweight. In terms of inference time and Floating-Point Operations Per Second (FLOPS), the proposed method outperforms the state-of-the-art techniques by around 30-100 times. Here, well-known datasets, i.e. ICDAR2015 and ICDAR2019, have been utilized for training and testing to validate the performance of the proposed model. Finally, the findings and discussion indicate that the proposed model is more efficient than the existing schemes.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Machine learning**.

Additional Key Words and Phrases: Deep Neural Network, Model Quantization, Edge Computing, Inference Time, Floating Point Operations per Second, Resource Constrained

1 INTRODUCTION

Scene text detection is the technique of finding words from scenic images. It is a complex process since there are so many environmental limitations, such as illuminations, lighting conditions, tiny and curved letters, and so on, that make it challenging to identify, and extract such messages. The methods used to recognize text in this

*The corresponding author

Authors' addresses: Kanak Manjari, School of Computer Science Engineering and Technology, Bennett University, Greater Noida, India, email:KM5723@bennett.edu.in; Madhushi Verma, School of Computer Science Engineering and Technology, Bennett University, Greater Noida, India, email:madhushi.verma@bennett.edu.in; Gaurav Singal, Department of Computer Science and Engineering, Netaji Subhas University of Technology, Delhi, India, email:gauravsingal789@gmail.com,gaurav.singal@nsut.ac.in; Suyel Namasudra, Department of Computer Science and Engineering, National Institute of Technology Patna, Bihar, India; Universidad Internacional de La Rioja, Logroño, Spain, email:suyelnamasudra@gmail.com.

2375-4699/2022/3-ART $15.00
https://doi.org/10.1145/3526217

scenario may be classified into two parts: (1) segmentation-based and (2) regression-based. A Full Convolution Network (FCN) [34] was used in a segmentation-based approach where further post-processing was required for transforming the output in the word boxes. Researchers have mostly utilized Single Shot Detectors (SSD) [32], Faster Region-based Convolutional Neural Networks (Faster R-CNN) [44], or mask region-based convolutional neural networks in the regression-based method, such as Mask R-CNN. These experiments on both techniques sought greater efficacy by trading model complexity and attained an F1 score of 80-90% for text detection. However, their performance was not best achieved for scene text detection.

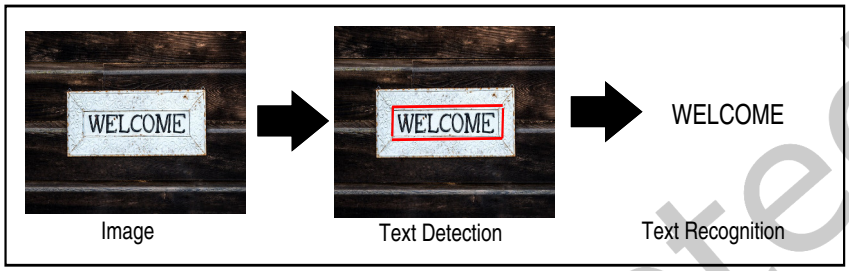

Fig. 1. Process of English text detection and extraction

One of the useful techniques for detecting and recognizing text or characters from the input images is Optical Character Recognition (OCR). It has piqued the interest of many researchers since it offers a wide range of uses, including assisting the differently-abled in reading books, number plate identification, and so on [37]. It consists of two stages: detection and recognition. The detection process involves segmentation of the region in the input picture containing the border of the texts, while recognition refers to the extraction of texts from that segmented area. As it can be easily seen in Fig. 1, an image has the text "WELCOME," the detection module will segment the border of the section of the picture having this text, and the recognition step will extract these texts from the segmented area of the image containing the text. This technique does not perform well when the document is of poor quality, and it is not worth doing for small amounts of text detection.

A morphological way of extraction of texts from the image has been adopted in [15]. This approach remains unaffected by noise, skew, or text orientation. It is also free from the best global and local thresholds based on fixed-size blocks. The OCR algorithm has been incorporated with this method to create a logical system for text analysis in the image, but it is computationally expensive. A novel scene text detection model called TextMountain has been developed in [62] where the key idea was to make use of border-centre information. The proposed technique can handle curved text as well as multi-oriented text. The experiments have been done on MLT, ICDAR2015, RCTW-17, and SCUT-CTW1500 datasets. The results have proved that the proposed method performs better than the existing ones with an F1 score of 76.85% on the MLT dataset that outperforms the current techniques, and it can be further enhanced.

Deep learning has been used for text detection and extraction in the past decades [45]. The performance of such models has been measured in terms of accuracy. Less importance has been given to other parameters related to the efficiency of a model, such as memory usage and inference time. In today's era, edge computing has been a trending technology as the focus is on developing mobile solutions [40, 49]. Edge computing provides information for scene text detection, collected by edge devices, such as Jetson Nano, and smartphones [11, 53]. As a result, models must be developed that are efficient enough to run on edge devices without compromising computations. Researchers have built efficient text identification models that are both quick and cost-effective, with low floating-point operations per second (FLOPS). PvaNet [18] has been used as a backbone for Efficient and Accurate Scene

Text Detection (EAST) [61] which was created to perform scene text detection. To extract visual characteristics, the EAST method uses a convolutional neural network. It is geared for text detection, and gives a detailed per-pixel prediction of words or text lines. Candidate selection, text area construction, and word segmentation are all eliminated for optimization. The post-processing only comprises thresholding and NMS on predicted geometric shapes. Initially, PvaNet was used to extract features from three layers in order: feature extractor stem, feature merging branch, and output layer. However, because of the limited sensing field, it could not extract additional image characteristics. One of the remedies discovered to tackle this problem was altering the backbone as stated in [46, 48]. A similar approach was adopted and employed for text detection in [46], but with a different backbone, i.e., the Residual Network (ResNet) [16] which is heavier than PvaNet but achieves greater accuracy. The Visual Geometry Group Network (VGGnet) [48] for the same purpose has been used in, though it was found to be a heavy network. Undoubtedly, these approaches produced good results but at the cost of increased computations, which is not suitable for edge devices [39]. The goal is to reduce the model complexity to make it suitable for edge devices [38, 41]. Hence, it is worth exploring and working on deep learning model reduction methods to achieve a good performance along with lightweight as well as a compact model for the assistive devices [36, 55]. This paper introduces a method that has been developed for a lightweight model that can perform text detection and extraction using quantization [20]. Dice loss [54] has been used instead of binary cross-entropy loss to improve the performance and the hyperparameters have been adjusted before and during the training. Quantization has been performed on the resultant model with two precisions, i.e. float16 and int8. Then the inference results have been re-evaluated on the default (float32) and both quantized models. One more method, i.e., hybridization, was performed by combining ResNet50 and MobileNetV2. Although the accuracy was not improved, the size of the hybridized model was reduced to almost 50%.

- Scene text detection has been performed in this paper as it is challenging due to several environmental limitations, such as illuminations, lighting conditions, tiny and curved letters, etc.
- A novel lightweight model has been developed for text detection and extraction on scene text datasets where ICDAR2015 and ICDAR2019 have been used for training. ResNet50 and MobileNetV2 have been used as the backbone of the proposed model, along with dice loss for training.
- The model's size has been reduced using quantization in the proposed model to make it lightweight, which makes it apt for deployment on mobile devices. A hybridized model has been computed by combining ResNet50 and MobileNetV2. The size of the hybridized model was reduced to almost 50% with a decrease in the accuracy.
- The performance analysis of the proposed model has been done using the ICDAR2019 test dataset along with real-life images captured by using the developed prototype of a smart cane.

The remaining paper is organized in the following manner: section 2 examines the related work of scene text detection, section 3 presents the proposed technique and describes the quantization process, in section 4, the comprehensive experimental data have been reported, and section 5 presents the summary and conclusion.

2 RELATED WORKS

Many researchers in the past decades have adopted text detection. The researchers, including deep learning techniques, have used different methods. The papers relevant to this work has been discussed in this section.
Zhou et al. [61] proposed a pipeline named EAST to cope with difficult scene text situations. This approach predicts arbitrary forms and quadrilateral shapes in an image through a single neural network. The proposed method achieved an F1 score of 0.7820 on the ICDAR 2015 dataset and an F1 score of 0.7820 on the COCO-Text and MSRA-TD500 datasets. Liu et al. [33] created an end-to-end trainable network for text detection and identification called Fast Oriented Text Spotting (FOTS). The findings show that the suggested technique outperformed the state-of-the-art methods utilizing the ICDAR 2015, ICDAR 2017 MLT, and ICDAR 2013 datasets. On ICDAR 2015, the text detection task outperformed the prior approach by more than 5%. Long et al. [35] proposed a technique

called TextSnake, to solve the real-world difficulties of previous methods while handling considerably more free-form text examples, such as curved text. It has effectively represented the texts in horizontal, vertical, and curved forms. It has achieved comparable performance on Total-Text, SCUT- CTW1500, ICDAR 2015, and MSRA-TD500 datasets. In terms of F-measure, it outperformed the baseline by more than 40% on Total-Text. TextBoxes is an end-to-end trainable brief scene text detector that is reliable. It efficiently identifies arbitrary-oriented scene text in a single network forward pass, as described by Liao et al. [27]. On the ICDAR 2015 and COCO Text images datasets, F1 scores of 0.817 and 0.5591 has been achieved, respectively.

Juang et al. [23] included affective computing into a 3D guidance system designed for a real-world campus using eye-tracking technologies. The interest region embedded in the environment has been used to analyze the user's gaze position and recognize the attention and emotion [1, 14, 47]. The feedback content reflects the area related to the emotion. Jiang et al. [22] created an object tracking framework based on event count images from an event camera to alleviate the lost-track issue caused by fast light changes in High Dynamic Range (HDR) settings, such as tunnels. The detector uses pre-labelled data during training, but erroneous or missing detections have been observed. The tracker produces consistent results for each initialized object, although drifting issues and failures have been detected. Jeon and Jeong [21] have proposed a compact and accurate scene text (CAST) detector to overcome the problems of computationally expensive models while maintaining high accuracy [2]. A balanced decoder has been carefully constructed instead of utilizing typical convolutional layers as decoders. Its performance has been found to be 1.1 times poorer in terms of F1 score but 30-115 times better in terms of FLOPS. Subedi et al. [53] has proposed a low-cost, high-performance OCR system that uses the most miniature embedded developer kit.

Zhang and Feng [60] have concentrated on Chinese text detection and recognition since there has been less study on this topic than there has been on English text detection and recognition. The feature extraction network has been switched from PvaNet to ResNet50 and MobileNet V2 to improve the sensing field of the EAST method. It improves the network's depth, allowing more features to be retrieved. Dasgupta et al. [8] have concentrated on text identification in the wild since there have been numerous issues with wild scene texts, such as occlusions, changing sizes, and orientations. A fully convolutional neural network has been developed in [29] with a unique feature called the Feature Representation Block (FRB) that performs superior information abstraction. Other benchmark datasets, such as ICDAR 2015, ICDAR 2017 MLT, COCO-Text, and MSRA-TD500 have been used to evaluate this method and substantially improve the results. He et al. [17] have proposed a two-stage architecture for real-time multi-scale scene text recognition. The first step is a new Scale-based Region Proposal Network (SRPN) that efficiently localizes texts from various sources. The second stage is a fully convolutional network-based scene text detector containing text terms from the first stage's suggestions. On the ICDAR2015 dataset, this technique received an F1 score of 85.40% at 16.5 frames per second. Liu et al. [31] have combined visual and text attention and developed a dual attention network [43] for image caption generation. They have used a fully convolutional network for image tag and fuse tag generation to train the model.

Ghosh et al. [13] has created an automated approach for identifying human behaviour based on a few handwritten lower case English characters from a to z. Loops, cursive, straight lines, slants, stroke thickness, and contour forms, among other structural elements, have been extracted using the suggested technique. Wu et al. [57] have developed a novel region reinforcement network to establish a relation between image and text. The experiments were conducted on MSCOCO and Flickr30k datasets to verify the effectiveness of this method. Xu et al. [59] have used structural information, such as poses to identify the location of texts.

Daldali and Souhar [7] have used the grayscale image as input data, resulting in superior outcomes when extracting handwritten text lines without the requirement for a binary representation of the document image. On testing with a public database of 123 handwritten Arabic manuscripts, the line recognition score was 97.5% with a matching score of 90%. Revanasiddappa and Harish [45] has introduced a new feature selection technique for text classification [52] based on Intuitionistic Fuzzy Entropy (IFE). The IFCM clustering technique has been used

to calculate the intuitionistic membership values. Compared to other well-known feature selection methods, the proposed method yields good results. To identify an appropriate approach of feature extraction, Boulid et al. [5] employed Arabic characters [51]. Both the strategies are based on two different visions: one portrays the image in terms of pixel distribution, while the other explains it using local patterns.

Jacob et al. [20] have developed a quantization technique that allows inference to be performed using integer-only arithmetic, which can be implemented more effectively on resource-constrained devices. The resulting quantization method improved the accuracy-on-device latency trade-off. Wu et al. [56] have reviewed the mathematical elements of quantization parameters. The focus areas have been those quantization methods that the processors can accelerate with high-throughput integer math pipelines. A solution to the problem of accomplishing fully end-to-end quantization utilizing limited bandwidth, such as 4-bit and using quantized networks to object recognition has been presented by Li et al. [26]. They discovered that these issues emerge due to instability during the quantization process' fine-tuning step, and they offered new ways to address these issues. The impact of pruning a neural network on its size and accuracy has been demonstrated by Liebenwein et al. [28]. This entails removing duplicate parameters and retraining the goal of maintaining accuracy. According to the findings, the pruned model approximates the unpruned model with a considerable loss in accuracy. Cygert and Czyzewski [6] has discussed model compression [10] strategies and how to minimize the computing cost of a model while maintaining accuracy. They demonstrated the compressed model's susceptibility to additive noise and corruption. The proposed data augmentation, which has been shown to improve model resilience, even in extremely compressed models. Proportionate Data Analytics (PDA) for heterogeneous healthcare data stream processing has been employed by Gao et al. [12] to construct Internet of Things (IoT) aided healthcare systems [3]. The categorization has been simplified using linear regression to distinguish errors from variations over time periods.

A variety of methods have been adopted in all these papers mentioned above, but they have only focused on improving the accuracy. Only a couple of papers focus on reducing the computations to make the model lightweight. However, the proposed model is accurate as well as efficient. Thus, it bridges the gap pointed above.

3 METHODOLOGY

The proposed method has five phases, as mentioned in Fig. 2. Firstly, the dataset has been prepared for training, followed by backbone selection for model training. Text detection is performed, followed by text recognition using the proposed architecture.

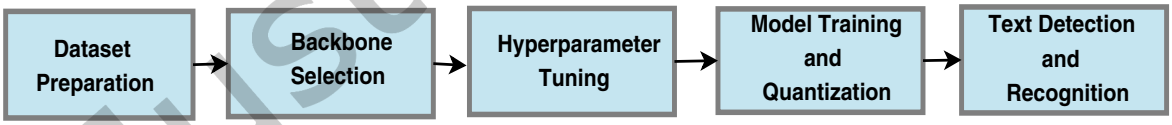

Fig. 2. Steps for implementation of experiment

3.1 Dataset Preparation

In this experiment, public test datasets have been used, which are ICDAR2015 [24] and ICDAR2019 [42] dataset. The ICDAR2015 dataset contains 1000 and 500 samples for training and testing, respectively. It's also annotated on a word-by-word basis, and all of the images are the same size, at 1280 ×720 pixels. ICDAR2019 consists of ten languages: Arabic, Bangla, Chinese, Hindi, English, German, Italian, Japanese, and Korean. For training and testing the model, the ICDAR2019 dataset has been used. The ICDAR2019 dataset has 10,000 scene sample images in its training set, each with its own ground truth file giving a set of bounding box coordinates for each text word

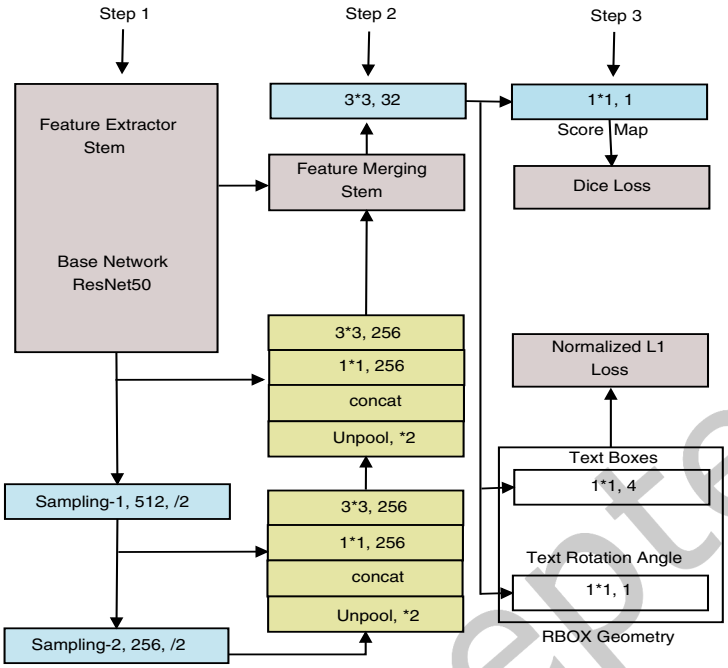

Fig. 3. Architecture of EAST text detection model

in the image. Because the resolutions of the photos in both datasets fluctuate, the size of each image should be equal. For the proposed model, a long-size-based picture scaling method has been used. The ICDAR2019 dataset includes 720×1280 long-side images, respectively, after resizing.

3.2 Backbone Selection

As discussed above, the ResNet50 model has been used as the backbone network of the text detection algorithm in this experiment. Before the development of ResNet50, the depth of neural networks was quite limited. The gradient descent problem occurs when a network is deepened in a deep neural network. As a result, ResNet50 was offered as a solution to the problem of network deterioration [25]. Let x be the input for the convolution layer, where *H(x)* is recorded as the learned feature. It should be able to learn the residual *F(x) = H(x) - x*, resulting in the initial learning feature being F(x) + x. The parameter Ws influences the value x in the eq. (1) such that the output forms are the same. The formula for the residual unit is as follows:

$$y = F(x, W_i) + W_s x \tag{1}$$

Here, y is the output, x is the input to the convolution layer, Ws and Wi are hyper parameters

3.3 Hyperparameter Tuning

The initial learning rate has been set to the value of 10^{-4}. Total $50k$ iterations, the model has been trained. $L2$ regularization with a 10^{-5} weight decay has been used. For the model training, a batch size of 16 was used. Batch Normalization (BN) has been implemented with $\gamma = 1$ and $\beta = 0$ for better results [19].

3.4 Model Training

The three phases of the EAST model used in the experiment are feature extraction, feature merging, and an output layer with score map generation, as illustrated in Fig. 3.
Feature Extraction: ResNet50 model has been used as a backbone of this detection algorithm which generates feature maps at different scales. Combining these feature maps with pool-based feature maps reduces the number of calculations. Shallow features can only detect tiny texts, but in-depth features can locate target texts. So, the need is to extract the deep features from images containing text information.
Feature Merging: The four convolution layers' output is f1-f4, with f1 being the first feature fusion layer's input. The expanded dimension is fused with the output f2 of the third convolution layer after it has been upsampled. The goal of feature upsampling is to increase the dimension of the feature map to match the size of the top layer's feature map, which will aid feature fusion upsampling in the future.

$$Upsampling\ (g_i) = upsampling(h_i) \tag{2}$$

Here, g_i is the intermediate layers and h_i is feature merging part

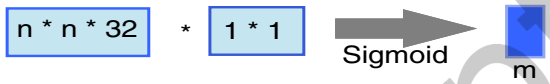

Fig. 4. Process of calculating confidence

Output Layer: The confidence, coordinates, and rotation angle of the detection box are all contained in this layer. The confidence value is determined by applying a convolutional operation to the kernel of 1×1, this yields a 1D vector, which is subsequently processed using a sigmoid function, as illustrated in Fig. 4 and eq. (3) on input (x).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

Here, x is the input and e^{-x} is the exponent of -x
The feature map's output is generated by the feature merging layer's convolution operation using four 1×1 convolution kernels. After that, the sigmoid function receives these four kernels. The final result is magnified 512 times, and the four text detection box coordinates are produced as illustrated in Fig. 5. The rotation angle of text is calculated using eq. (5).

$$output_a = sigmoid(output_b) \tag{4}$$

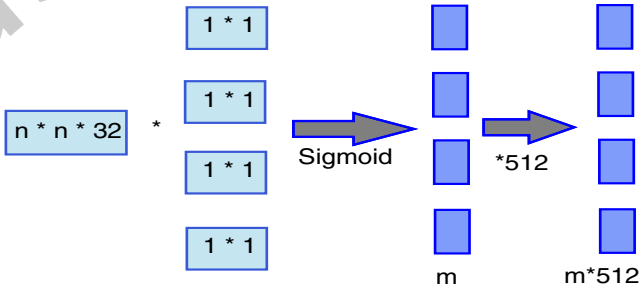

Fig. 5. Process of feature merging

$$angle(\theta) = (output_{\mathrm{a}} - 0.5) * \frac{\pi}{2} \tag{5}$$

Here, $\mathrm{output_b}$ is the intermediate output, and sigmoid is the activation function applied to it
For this experiment, two methods for loss calculation, dice loss and cross-entropy, have been used. Initially, the cross-entropy loss was used as it is the standard method for segmentation. Later, after analyzing the results, dice loss was preferred as it performs well for both balanced and imbalanced datasets. It considers both losses, locally and globally, which is important for improving accuracy. The total loss function L_{total} is the combination of score map loss (L_{score}) and geometry loss (L_{geo}). The score map is the loss in predicting the score map of detection boxes, and the loss encountered during the generation of text geometry for both small and large text regions is called geometry loss.

$$L_{total} = L_{score} + \lambda_{\mathrm{L_{geo}}} L_{geo} \tag{6}$$

Here, L_{score} is the score map loss, $\mathrm{L_{geo}}$ is the geometry loss, $\lambda_{\mathrm{L_{geo}}}$ is a regularizer that ranges from 0 and 1. The relative importance of L_{geo} is adjusted using this parameter. $\lambda_{\mathrm{L_{geo}}}$=1 has been set in this experiment as it is the standard value

Algorithm 1 Text Detection and Extraction

Input: The ICDAR2015 and ICDAR2019 dataset has been used
Output: Text is detected and extracted (bounding boxes and texts)

1: **for** image(i) **do**
2: f1-f4 = conv (feature_extractor (i)) using eq. (1)
3: intermediate_out = feature_merging (f1-f4)
4: output = (Upsampling ($\mathrm{g_i}$) intermediate_out) using eq. (2)
5: 1D vector = 1×1 conv (output)
6: Confidence (D), coordinate (C) and rotation angle (θ) calculated using respective eq. (3), (4) and (5) and loss function from eq. (6)
7: final_bounding_box = nms (startx, starty, endx, endy)
8: Extracted_Text = Extraction (final_bounding_box)
9: **end for**

3.5 Quantization

It is vital to lower the size and computational complexity of models when they migrate from servers to the edge. Quantization, which substitutes floating points with integers inside the network, is a unique and exciting approach. Quantization's core premise is that by converting weights and inputs into integer types, a specific technology may save memory and speed up calculations. However, there is a cost will occur that quantization can result in considerable accuracy loss. It has been observed that quantization is mostly accomplished through operations. There are more options than going from float32 to int8, such as going from float32 to float16. These floaring points can also be mixed, for example, Matrix multiplications can be quantized to int8, whereas activations can be quantized to float16.
A approximate estimate is quantification. The lesser the performance deterioration, in general, the closer the approximation. Converting everything to float16 will save half the memory and possibly not lose much accuracy, but there will be no speed advantage. Quantifying using int8 can result in much faster inference, but the performance will almost likely be poorer. In other circumstances, it may not even work, requiring quantization-aware training. Quantization can be done in two different ways:

- Post-training: After the model has been trained with float32 weights and inputs, the weights are quantized. Its key benefit is that it is simple to use; but, accuracy may be reduced.

- Quantization-aware training: When training, keep track of the weights. The quantized weights' gradients are also computed here. This produces the greatest results when using int8 quantization, however it takes longer than the other option.

3.6 Text Detection and Recognition

This approach extracts [30] shared convolutional features from an input picture using a shared backbone. Let us assume x as an input image, where the shared feature maps may be represented as $I \in x^{h^*w^*d}$. Here h denotes the height, w denotes the width and d denotes the channel number of I. As illustrated in eq. (7), the detection module accepts I as an input and predicts the location of text areas.

$$B, D = TextDetection(I) \tag{7}$$

Here, I is the number of channels, $B = (b_1, b_2,...,b_m)$ which is a set of m bounding boxes and $b_i = (x_{min}, y_{min}, x_{max}, y_{max})$. Along with the bounding box coordinates, it also outputs the confidence score as $D = (d_1, d_2, ..., d_m)$ for m detections. Afterwards, the text recognition module uses pytesseract [50] to extract a sequence of characters from the previously detected text regions

4 PERFORMANCE ANALYSIS

The model's performance was assessed using several criteria, including precision, recall, f1 score, and inference time.

4.1 Experimental Setup

An Intel Xeon Central Processing Unit (CPU), 128 GB random access memory (RAM), and a single GTX TITAN X has been used throughout this work for experiments. Ubuntu version 18.4 LTS has been used as the primary operating system to conduct the test. The main programming language to train the model is Python 3. The installation of Anaconda for Ubuntu 18 Python 3.7 has been done to use Cuda. The precision, recall, F1 score, inference time, and FLOPS have been used to compare all trials. The recall is the measure of the model successfully detecting true positives, whereas precision is the ratio between true positives and all positives. The harmonic mean of accuracy and recall is used to get the F1 score. The inference time is when it takes to infer a picture in milliseconds (ms). FLOPS measures a computer's performance useful in scientific calculations that require floating-point operations.

4.2 Training Details

EAST text detector is used as a deep learning model with ResNet50 as a backbone that is pre-trained on ImageNet data for training the proposed model. For 50k iterations, the model is trained. L2 regularization with a 10^{-5} weight decay has been used. For the training, a batch size of 16 was used.

4.3 Results Discussion

The proposed technique is compared to numerous state-of-the-art text detection algorithms to compare their performance to analyze its improvement over the existing developed models. The findings are compared using metrics like Hmean, accuracy, recall, size, inference time, and FLOPS. Gnuplot software is used to generate graphs and curves based on testing results. It is a command-line, and graphical user interface application for plotting functions, data, and data fits in two and three dimensions. The detection performance of the proposed model is tested on the ICDAR2019 dataset and reported qualitatively in Fig. 6. The true positive words indicating correctly identified are bounded by green colour rectangle boxes. The results have been analyzed for 5 Asian languages out of total 10 languages, which are English, Hindi, Bangla, Chinese, and Arabic. The model is correctly

Fig. 6. Detection performance of model on ICDAR2019 test dataset for different languages (a) English (b) Hindi (c) Bangla (d) Chinese, and (e) Arabic

able to localize texts in different colours and backgrounds of these 5 languages. The qualitative measurement of end-to-end performance is presented in Fig. 7. The model can correctly detect and recognize words from non-complex images which are easily readable. On the other hand, the model's performance is compromised on complex images due to various environmental conditions, such as illuminations, colour effects, etc.
In Table 1, the existing methods where EAST has been used for efficient text detection are compared with the proposed method. The initial EAST model has PvaNet as the backbone which is replaced with other backbones, such as VGG16, MobilenetV2, EfficientNet, and ResNet, etc. for better efficiency. In Figs. 8 and 9, the experimental results done on ICDAR2019 dataset with two variants which contains multi-lingual sample. Initially, the model is trained using all the 10 types of language sample but the Hmean is observed to be low. After which, 5 Asian language samples are chosen for model training out of 10 which are English, Hindi, Bangla, Chinese, and Arabic. The reason behind choosing the five Asian languages out of the 10 and performing the experiment on it is to focus on those languagess which are low-level or which are used in Asian countries. The ICDAR2019 has only these Asian language data which is chosen for the region-specific experiment. The Hmean is observed to be around

(a) (b)

(c) (d)

Fig. 7. End-to-End performance on ICDAR2019 and real-time dataset (a) Non-complex sample (b) Complex sample (c) Non-complex sample, and (d) Complex sample

Table 1. Result of EAST models with different networks on ICDAR dataset

Paper	Year	Precision	Recall	Hmean	Network
EAST [61]	2017	83.57	73.47	78.20	PvaNet
EAST [61]	2017	-	-	76.40	VGG16
CAST [21]	2020	83.20	74.65	78.66	MobileNetV2
Low-Cost OCR [53]	2020	6.11	9.94	7.37	EfficientNet
Proposed (ICDAR2015)	2021	84.61	77.27	80.78	ResNet
Proposed (ICDAR2019)	2021	81.26	75.45	77.89	ResNet

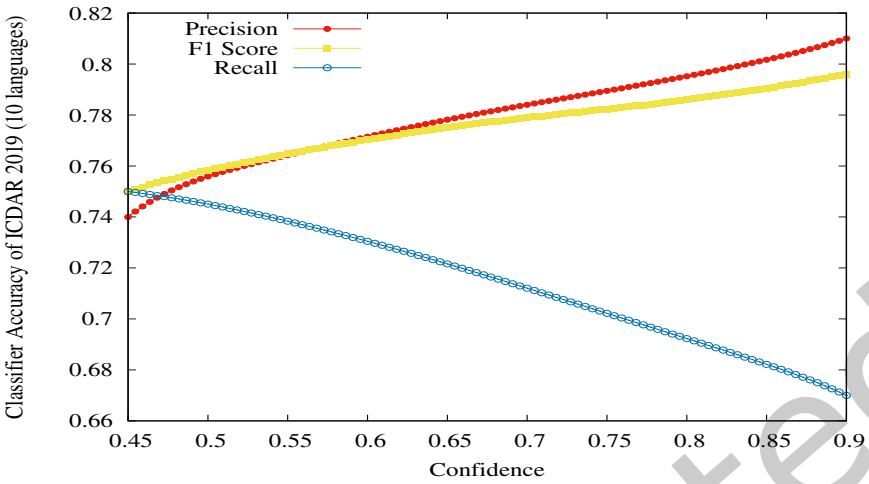

Fig. 8. Result of EAST models with original ICDAR2019 dataset

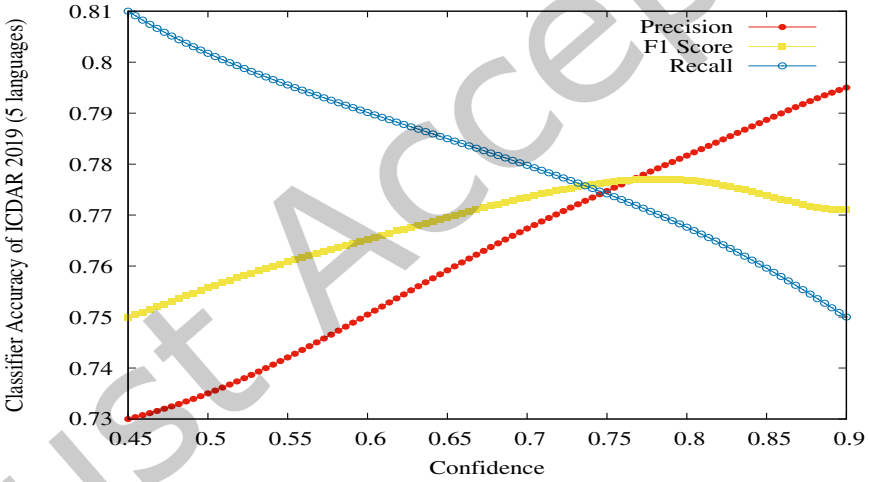

Fig. 9. Result of EAST models with custom ICDAR2019 dataset

79.60 for original ICDAR2019 as shown in Fig. 10 and slightly better for 5 chosen languages of ICDAR2019, which is good enough for experiment but not for real-time usage. As the backbone of the model has such a large impact on the model's efficiency, it's critical to pick it intelligently. With the increase in confidence or threshold value of the classifier, which is 0.5, accuracy increases and recall drops, while the F1 score tends to fluctuate around constant values. In Fig. 11(a), the loss encountered during the training process are plotted. It can be observed that the initial loss is quite high which eventually decreases in increasing the number of epochs. In Fig. 11(b), the precision-recall curve is plotted between precision (y-axis) and recall (x-axis). A good performance model should have its precision-recall curve closer to its top-right corner.

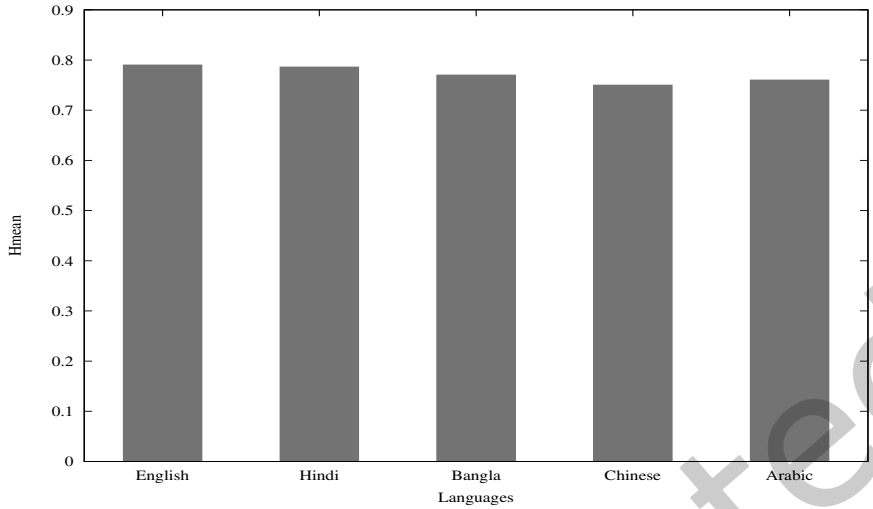

Fig. 10. Hmean of the 5 langauges of ICDAR2019 dataset

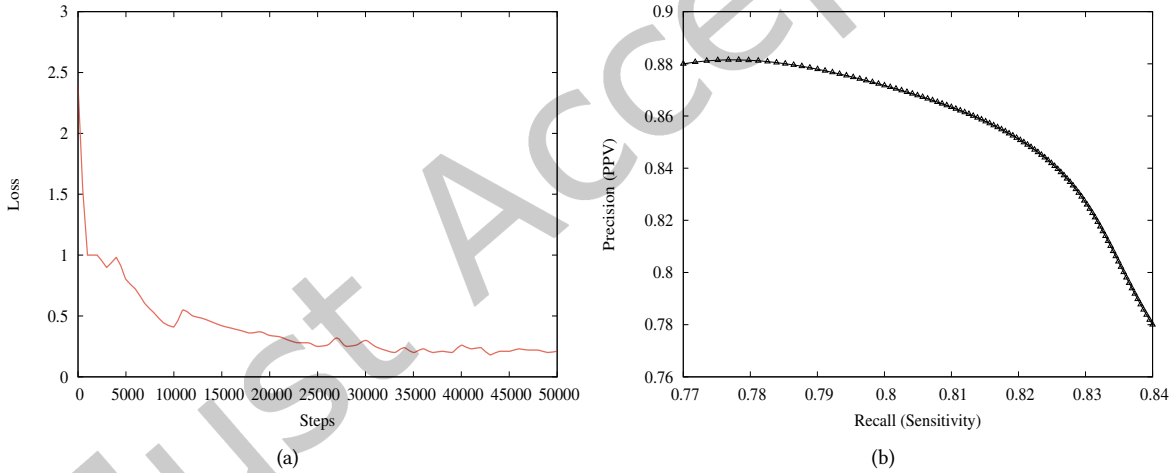

Fig. 11. Performance evaluation graphs on ICDAR2019 dataset (a) Loss per epoch during training and (b) Precision-recall curve

The model that is built after training must be frozen before it can be used. The act of recognizing and preserving all of the necessary items (graphs, weights, and so on) in a single, easily accessible file is known as freezing. Because edge devices have limited memory and compute power, a variety of optimization strategies may be used to make the model lighter and more suited for deployment on them. The default precision of the frozen model is float32, and the size of the frozen model is 96MB. The inference time is observed to be 0.64 seconds. Quantization [20] reduces the size of the model, but at the price of accuracy. It works by lowering the accuracy

of the numbers used to represent the parameters of a model, which are typically 32-bit floating integers. This reduces the size of the model, allowing for quick calculation. The precision is converted from float32 to float16 and int8 by applying quantization. The size of the original model is reduced to 50% and 25% after float16 and int8 precision, respectively, as shown in Table 2. The inference time is also reduced to 0.45 seconds after conversion into float16 accuracy. But, the inference time is increased to 20 times when the model is converted into int8 precision. During inference, the operations get converted from float to int, which may be the reason behind the increase in inference time.

Table 2. End-to-End Result of quantization of proposed model trained on ICDAR2019 dataset

Model (Type Precision)	Size	Inference Time
Float32	96 MB	0.64sec
Float16	43 MB	0.45 sec
Int8	25 MB	23.14 sec

The model's performance is not degraded much when quantized from float32 precision to float16 precision. However, when the model is quantized from float32 precision to int8 precision, the performance is significantly degraded. Although the size is also reduced, it is unsuitable for real-time usage due to lower accuracy. Therefore, the model with float16 precision is used to achieve an efficient and accurate performance. The same can be observed from the results obtained for detection, where it was found difficult to detect and extract texts from complex images.

Table 3. Result of inference of different methods using ICDAR dataset

Paper	Year	Hmean	Inference Time	FLOPS
PixelLink [9]	2018	83.70	1890msec (CPU)	650.2G
FOTS [33]	2018	87.99	133msec (GPU)	-
TextBoxes++ [27]	2018	82.90	435msec (GPU)	-
TextSnake [35]	2018	82.60	909msec (GPU)	-
CRAFT [4]	2019	86.90	42660msec (CPU)	1023.9G
CharNet [58]	2019	90.97	1230000msec (CPU)	2402.9G
CAST [21]	2020	81.06	352.90msec (CPU)	20.8G
Proposed (MobileNetV2) (ICDAR2015)	2021	74.50	350msec (CPU)	21.2G
Proposed (ResNet50) (ICDAR2015)	2021	80.78	420msec (CPU)	6G
Proposed (ResNet50) (ICDAR2019)	2021	77.89	425msec (CPU)	11G

To analyze the performance of the proposed method on resource-constrained devices, the inference is performed on the CPU. The results are summarized in Table 3. Two backbones are used during the experiment which is MobileNetV2 and ResNet50, but the performance of the model trained using the MobileNetv2 backbone is not good enough to be used in real-time. Hence, the model trained using the ResNet50 backbone is preferred for further experiment. PixelLink [9] and CRAFT [4] are 4-110 times slower than the proposed technique in terms of inference time and 50-75 times slower in terms of FLOPS. The accuracy achieved using CharNet [58] was one of the highest, but it is also the most inefficient method among all other methods discussed. The inference time is computed on GPU in FOTS, TextBoxes++, and TextSnake. The accuracy of the proposed method is almost the same as of CAST [21], the inference time is slightly higher than CAST, but the number of FLOPS is three times lower than CAST. The proposed technique significantly reduces inference time compared to all other methods.

Table 4. Analysis of Recognition Result

Word Accuracy		**Character Accuracy**	
Complex (C)	Non-Complex (NC)	Complex (C)	Non-Complex (NC)
62%	80%	68%	88%

To meet the current industry requirement, it is necessary to focus on the real-time usage of the model along with its performance. It meets the requirement of a good performance model with limited memory usage for deployment on edge devices like Jetson NANO. Thus, it can be deployed and used in the mobile environment and edge computing.

The recognition results of the proposed model is analyzed on the ICDAR2019 test dataset and a set of real-life images. The samples are categorized into two categories: complex images that are denoted by C and Non-complex images that NC denotes in Table 4. The images with no illumination problem and which do not contain small texts are put into the non-complex category, and the vice-versa are put in the complex category. It has been observed that non-complex images have better recognition results than complex images. Also, the word-level accuracy is a bit compromised than character-level accuracy. Although the model's size is reduced and it has been made computationally inexpensive, the accuracy still needs to be further improved. The quantization process lowers the accuracy to a certain extent. It performs well for non-complex images where texts are clear and where there are no lighting issues. But, it would be nice if the model performs the same way in a challenging environment.

5 CONCLUSIONS AND FUTURE WORK

In the era of edge computing, it is a necessity to have lightweight software that requires limited resources, such as memory, processing power, power backup, etc., for real-time usage. Scene text detection is the technique of detecting words from scenic images. This paper is focused on solving two major challenges, namely scene text detection and quantization, in the field of deep learning and computer vision. Here, a lightweight deep learning model has been developed for scene text detection using the ResNet50 backbone. The proposed model has been quantized to make it lightweight by converting the default precision from float32 to float16 and int8, and it is more efficient and effective in size, computation time, and FLOPS. The proposed model outperforms some state-of-the-art approaches by 30 to 100 times in terms of the size of the model. The performance of the model has a balance of accuracy and efficiency, developed using the ICDAR2019 dataset, which contains samples of 10 different languages. In the context of mobile devices, there is plenty of space for this notion to grow. In the future, the proposed scheme can be extended using hybridization of two or better performing models to improve its performance. Moreover, various other languages can be tested for making them available to a wide range of users.

ACKNOWLEDGEMENT

This work is supported and sponsored by the Department of Biotechnology MINISTRY OF SCIENCE and TECHNOLOGY, Government of India. The project title is ASE-Assistive Sight Enabler (an assistive technology for the visually impaired) with file number BT/PR34546/AI/133/12/2019. Also, it is supported by the stipend provided by Bennett University to the first author.

REFERENCES

[1] Divyansh Agrawal, Sachin Minocha, Suyel Namasudra, and Sathish Kumar. 2021. Ensemble Algorithm using Transfer Learning for Sheep Breed Classification. In *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, 199–204.

[2] Usman Ahmed, Gautam Srivastava, Unil Yun, and Jerry Chun-Wei Lin. 2021. EANDC: An explainable attention network based deep adaptive clustering model for mental health treatments. *Future Generation Computer Systems* (2021).
[3] Hafiz Munsub Ali, Jun Liu, Syed Ahmad Chan Bukhari, and Hafiz Tayyab Rauf. 2021. Planning a secure and reliable IoT-enabled FOG-assisted computing infrastructure for healthcare. *Cluster Computing* (2021), 1–19.
[4] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. 2019. Character region awareness for text detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9365–9374.
[5] Youssef Boulid, Abdelghani Souhar, and Mohamed Youssfi Elkettani. 2017. Handwritten Character Recognition Based on the Specificity and the Singularity of the Arabic Language. *International Journal of Interactive Multimedia & Artificial Intelligence* 4, 4 (2017).
[6] Sebastian Cygert and Andrzej Czyzewski. 2021. Robustness in Compressed Neural Networks for Object Detection. *arXiv preprint arXiv:2102.05509* (2021).
[7] Mehdi Daldali and Abdelghani Souhar. 2019. Handwritten Arabic documents segmentation into text lines using seam carving. *IJIMAI* 5, 5 (2019), 89–96.
[8] Kinjal Dasgupta, Sudip Das, and Ujjwal Bhattacharya. 2020. Scale-Invariant Multi-Oriented Text Detection in Wild Scene Image. In *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2041–2045.
[9] Dan Deng, Haifeng Liu, Xuelong Li, and Deng Cai. 2018. Pixellink: Detecting scene text via instance segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[10] Sakshi Dhall, Ashutosh Dhar Dwivedi, Saibal K Pal, and Gautam Srivastava. 2021. Blockchain-based Framework for Reducing Fake or Vicious News Spread on Social Media/Messaging Platforms. *Transactions on Asian and Low-Resource Language Information Processing* 21, 1 (2021), 1–33.
[11] Oi-Mean Foong, Suziah Sulaiman, and Kiing Kiu Ling. 2013. Text signage recognition in Android mobile devices. *Citeseer Journal of computer Science* 9, 12 (2013), 1793.
[12] Jiechao Gao, Wenpeng Wang, Zetian Liu, Md Fazlay Rabbi Masum Billah, and Bradford Campbell. 2021. Decentralized Federated Learning Framework for the Neighborhood: A Case Study on Residential Building Load Forecasting. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. 453–459.
[13] Subhankar Ghosh, Palaiahnakote Shivakumara, Prasun Roy, Umapada Pal, and Tong Lu. 2020. Graphology based handwritten character analysis for human behaviour identification. *CAAI Trans. Intell. Technol.* 5, 1 (2020), 55–65.
[14] A. Gupta and S. Namasudra. 2022. A novel technique for accelerating live migration in cloud computing. *Automated Software Engineering* (2022).
[15] Y.M.Y. Hasan and L.J. Karam. 2000. Morphological text extraction from images. *IEEE Transactions on Image Processing* 9, 11 (2000), 1978–1983.
[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[17] Wenhao He, Xu-Yao Zhang, Fei Yin, Zhenbo Luo, Jean-Marc Ogier, and Cheng-Lin Liu. 2020. Realtime multi-scale scene text detection with scale-based region proposal network. *Elsevier Pattern Recognition* 98 (2020), 107026.
[18] Sanghoon Hong, Byungseok Roh, Kye-Hyeon Kim, Yeongjae Cheon, and Minje Park. 2016. PVANet: Lightweight deep neural networks for real-time object detection. *1st International Workshop on Efficient Methods for Deep Neural Networks (EMDNN)* (2016).
[19] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.
[20] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2704–2713.
[21] Minjun Jeon and Young-Seob Jeong. 2020. Compact and accurate scene text detector. *Multidisciplinary Digital Publishing Institute Applied Sciences* 10, 6 (2020), 2096.
[22] Rui Jiang, Xiaozheng Mou, Shunshun Shi, Yueyin Zhou, Qinyi Wang, Meng Dong, and Shoushun Chen. 2020. Object tracking on event cameras with offline–online learning. *CAAI Transactions on Intelligence Technology* 5, 3 (2020), 165–171.
[23] Li-Hong Juang, Ming-Ni Wu, and Cian-Huei Lin. 2020. Affective computing study of attention recognition for the 3D guide system. *CAAI Transactions on Intelligence Technology* 5, 4 (2020), 260–267.
[24] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. 2015. ICDAR 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 1156–1160.
[25] Bin Li and Dimas Lima. 2021. Facial expression recognition via ResNet-50. *Elsevier International Journal of Cognitive Computing in Engineering* 2 (2021), 57–64.
[26] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. 2019. Fully quantized network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2810–2819.

[27] Minghui Liao, Baoguang Shi, and Xiang Bai. 2018. Textboxes++: A single-shot oriented scene text detector. *IEEE transactions on image processing* 27, 8 (2018), 3676–3690.
[28] Lucas Liebenwein, Cenk Baykal, Brandon Carter, David Gifford, and Daniela Rus. 2021. Lost in pruning: The effects of pruning neural networks beyond test accuracy. *Proceedings of Machine Learning and Systems* 3 (2021).
[29] Jerry Chun-Wei Lin, Yinan Shao, Youcef Djenouri, and Unil Yun. 2021. ASRNN: a recurrent neural network with an attention model for sequence labeling. *Knowledge-Based Systems* 212 (2021), 106548.
[30] Jerry Chun-Wei Lin, Yinan Shao, Yujie Zhou, Matin Pirouz, and Hsing-Chung Chen. 2019. A Bi-LSTM mention hypergraph model with encoding schema for mention extraction. *Engineering Applications of Artificial Intelligence* 85 (2019), 175–181.
[31] Maofu Liu, Lingjun Li, Huijun Hu, Weili Guan, and Jing Tian. 2020. Image caption generation with dual attention mechanism. *Information Processing & Management* 57, 2 (2020), 102178.
[32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.
[33] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. 2018. Fots: Fast oriented text spotting with a unified network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5676–5685.
[34] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3431–3440.
[35] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, and Cong Yao. 2018. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *Proceedings of the European conference on computer vision (ECCV)*. Springer, 20–36.
[36] Kanak Manjari, Madhushi Verma, and Gaurav Singal. 2019. A Travel Aid for Visually Impaired: R-Cane. In *International Conference on Smart City and Informatization*. Springer, 404–417.
[37] Kanak Manjari, Madhushi Verma, and Gaurav Singal. 2020. A survey on assistive technology for visually impaired. *Elsevier Internet of Things* 11 (2020), 100188.
[38] Suyel Namasudra. 2020. Fast and secure data accessing by using DNA computing for the cloud environment. *IEEE Transactions on Services Computing* (2020).
[39] Suyel Namasudra. 2021. Data access control in the cloud computing environment for bioinformatics. *International Journal of Applied Research in Bioinformatics (IJARB)* 11, 1 (2021), 40–50.
[40] Suyel Namasudra and Ganesh Chandra Deka. 2021. *Applications of blockchain in healthcare*. Springer.
[41] Suyel Namasudra, Ganesh Chandra Deka, Prashant Johri, Mohammad Hosseinpour, and Amir H Gandomi. 2021. The revolution of blockchain: State-of-the-art and research challenges. *Archives of Computational Methods in Engineering* 28, 3 (2021), 1497–1515.
[42] Nibal Nayef, Yash Patel, Michal Busta, Pinaki Nath Chowdhury, Dimosthenis Karatzas, Wafa Khlif, Jiri Matas, Umapada Pal, Jean-Christophe Burie, Cheng-lin Liu, et al. 2019. ICDAR2019 robust reading challenge on multi-lingual scene text detection and recognition—RRC-MLT-2019. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 1582–1587.
[43] Hafiz Tayyab Rauf, Jiechao Gao, Ahmad Almadhor, Muhammad Arif, and Md Tabrez Nafis. 2021. Enhanced bat algorithm for COVID-19 short-term forecasting using optimized LSTM. *Soft Computing* 25, 20 (2021), 12989–12999.
[44] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497* (2015).
[45] MB Revanasiddappa and BS Harish. 2018. A new feature selection method based on intuitionistic fuzzy entropy to categorize text documents. *IJIMAI* 5, 3 (2018), 106–117.
[46] Shaohui Ruan, Junguo Lu, Fengming Xie, and Zhongxiao Jin. 2018. A novel method for fast arbitrary-oriented scene text detection. In *2018 Chinese Control And Decision Conference (CCDC)*. IEEE, 1652–1657.
[47] Yinan Shao, Jerry Chun-Wei Lin, Gautam Srivastava, Alireza Jolfaei, Dongdong Guo, and Yi Hu. 2021. Self-attention-based conditional random fields latent variables model for sequence labeling. *Pattern Recognition Letters* 145 (2021), 157–164.
[48] Baoguang Shi, Xiang Bai, and Serge Belongie. 2017. Detecting oriented text in natural images by linking segments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2550–2558.
[49] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge computing: Vision and challenges. *IEEE internet of things journal* 3, 5 (2016), 637–646.
[50] Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, Vol. 2. IEEE, 629–633.
[51] Abdelghani Souhar, Youssef Boulid, ElB Ameur, Mly Ouagague, et al. 2017. Segmentation of Arabic Handwritten Documents into Text Lines using Watershed Transform. *International Journal of Interactive Multimedia & Artificial Intelligence* 4, 6 (2017).
[52] Gautam Srivastava, Praveen Kumar Reddy Maddikunta, and Thippa Reddy Gadekallu. 2021. A Two-stage Text Feature Selection Algorithm for Improving Text Classification. (2021).
[53] Bharat Subedi, Jahongir Yunusov, Abdulaziz Gaybulayev, and Tae-Hyong Kim. 2020. Development of a Low-cost Industrial OCR System with an End-to-end Deep Learning Technology. *IEEK Journal of Embedded Systems and Applications* 15, 2 (2020), 51–60.
[54] Lorenzo Traldi. 2007. Generalized dice: Many questions and a few answers. *Graph Theory Notes of New York* 53 (2007), 39–42.

[55] Kanak Manjari, Madhushi Verma, and Gaurav Singal. 2019. CREATION: Computational ConstRained Travel Aid for Object Detection in Outdoor eNvironment. In *2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE, 247–254.
[56] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. 2020. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation. *arXiv e-prints* (2020), arXiv–2004.
[57] Jie Wu, Chunlei Wu, Jing Lu, Leiquan Wang, and Xuerong Cui. 2021. Region Reinforcement Network with Topic Constraint for Image-Text Matching. *IEEE Transactions on Circuits and Systems for Video Technology* (2021), 1–1.
[58] Linjie Xing, Zhi Tian, Weilin Huang, and Matthew R Scott. 2019. Convolutional character networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9126–9136.
[59] Xiaogang Xu, Ying-Cong Chen, Xin Tao, and Jiaya Jia. 2021. Text-Guided Human Image Manipulation via Image-Text Shared Space. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), 1–1.
[60] Jianxin Zhang and Yunhai Feng. 2020. Advanced Chinese Character Detection for Natural Scene Based on EAST. In *Journal of Physics: Conference Series*, Vol. 1550. IOP Publishing, 032050.
[61] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. 2017. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 5551–5560.
[62] Yixing Zhu and Jun Du. 2021. Textmountain: Accurate scene text detection via instance segmentation. *Elsevier Pattern Recognition* 110 (2021), 107336.