IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 21, 2020, accepted May 3, 2020, date of publication May 12, 2020, date of current version June 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2994214

Urdu-Text Detection and Recognition in Natural Scene Images Using Deep Learning

SYED YASSER ARAFAT[1,2] **AND MUHAMMAD JAVED IQBAL**[1]
[1]Department of Computer Science, University of Engineering and Technology (UET), Taxila 47080, Pakistan
[2]Department of CS and IT, Mirpur University of Science and Technology (MUST), Mirpur 10250, Pakistan

Corresponding authors: Syed Yasser Arafat (syed.yasser.arafat@gmail.com) and Muhammad Javed Iqbal (javed.iqbal@uettaxila.edu.pk)

ABSTRACT Urdu text is a cursive script and belongs to a non-Latin family of other cursive scripts like Arabic, Chinese, and Hindi. Urdu text poses a challenge for detection/localization from natural scene images, and consequently recognition of individual ligatures in scene images. In this paper, a methodology is proposed that covers detection, orientation prediction, and recognition of Urdu ligatures in outdoor images. As a first step, the custom FasterRCNN algorithm has been used in conjunction with well-known CNNs like Squeezenet, Googlenet, Resnet18, and Resnet50 for detection and localization purposes for images of size 320 $\times$ 240 pixels. For ligature Orientation prediction, a custom Regression Residual Neural Network (RRNN) is trained/tested on datasets containing randomly oriented ligatures. Recognition of ligatures was done using Two Stream Deep Neural Network (TSDNN). In our experiments, five-set of datasets, containing 4.2K and 51K Urdu-text-embedded synthetic images were generated using the CLE annotation text to evaluate different tasks of detection, orientation prediction, and recognition of ligatures. These synthetic images contain 132, and 1600 unique ligatures corresponding to 4.2K and 51K images respectively, with 32 variations of each ligature (4-backgrounds and font 8-color variations). Also, 1094 real-world images containing more than 12k Urdu characters were used for TSDNN's evaluation. Finally, all four detectors were evaluated and used to compare them for their ability to detect/localize Urdu-text using average-precision (AP). Resnet50 features based FasterRCNN was found to be the winner detector with AP of.98. While Squeeznet, Googlenet, Resnet18 based detectors had testing AP of.65, .88, and .87 respectively. RRNN achieved and accuracy of 79% and 99% for 4k and 51K images respectively. Similarly, for characters classification in ligatures, TSDNN attained a partial sequence recognition rate of 94.90% and 95.20% for 4k and 51K images respectively. Similarly, a partial sequence recognition rate of 76.60% attained for real world-images.

INDEX TERMS BLSTM, deep neural network, FasterRCNN, image classification, Nastalique, optical character recognition (OCR), regression residual neural network (RRNN), synthetic urdu text, text detection, two stream deep neural network (TSDNN).

I. INTRODUCTION

As autonomous vehicles and other intelligent devices/Cell-Phones/mobile devices [1]/Robots [2] are coming online and they also need to understand the environment in which they are operating. Urdu is, cursively written complex language than other non-cursive languages like English, Russian, etc. Urdu is the national language of Pakistan and also 6-Indian states [3], Hence covering more than 260 million people. Different kind of instructions for people's guidance is often written everywhere in public places, especially on different sign-boards, hoardings, banners along the roads. Non-availability of outdoor datasets for the training of models is a problem for Urdu-text/script. So, the availability of outdoor Urdu text datasets facilitates in evaluating different kinds of learning models for judging their effectiveness for text detection, Orientation prediction, and ligature recognition is needed. Beside that there is a need for a robust script detection system that can locate Urdu script in complex open outside environments, from a given input patch of an image. It may need orientation correction and consequently, the detected text region is further processed by Optical Character

The associate editor coordinating the review of this manuscript and approving it for publication was Chenguang Yang.

Recognition (OCR) to recognize what has been written. The output then, in turn, can be used for content analysis and indexing.

The main motivation for Urdu photo-OCR in the modern-day world is due to its various applications like scene understanding, robot / autonomous vehicle navigation, text reading for visually impaired, image retrieval, data mining of google-street-view images, real-time multi-lingual translation product Advertisement identification, Political people's posters identification, etc.

Detection of Urdu-text is challenging due to some of the following problems like types of scripts (Nastaliq, Naskh, handwritten, hybrid, etc.), scale, large variance of text patterns, oriented text, complicated background, color, contrast, perspective distortions, image quality and variation of text size [4]–[6]. Figure 1 shows key challenges in the detection of Urdu-Text/Script. These issues in detection make it difficult for any photo-OCR system to detect and recognize text with maximum confidence.

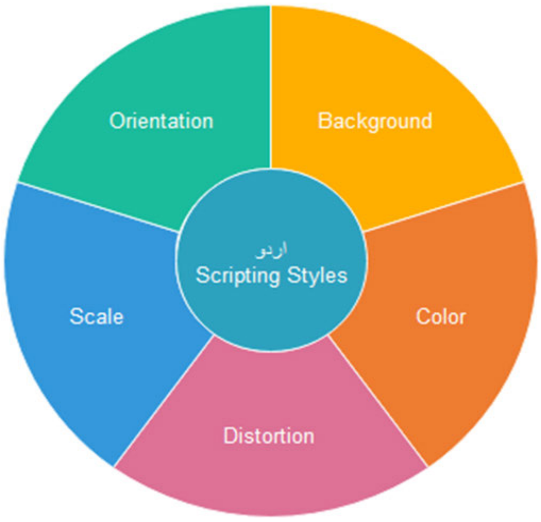


FIGURE 1. **Major challenges in Urdu detection.**

Detecting text (especially English and Chinese) in an outdoor environment has been done extensively [7]–[10]. Recently, little work has been done on outdoor Urdu-text detection in images [11] along with Arabic text detection [12]. Although, there are research studies on artificial Urdu-text detection in video frames [4]–[6], [13]–[17]. While regarding Urdu script orientation determination no well-known study exists. Generally, any text detection and recognition system (also called photoOCR) for outdoor imagery consist of two main stages/pipelines, detection, and recognition. Detection is the first step for the next recognition stage in any photoOCR system. Its importance is 2-fold as if the first module is not capable of robustly identifying the text regions, then the later stage of recognition is doomed and results in a low recognition rate. Natural outdoor text images are more challenging than simple written Urdu text on documents/books as different writing

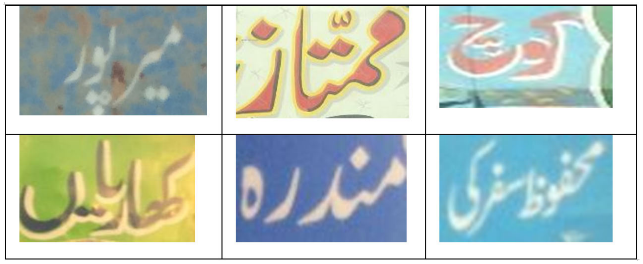
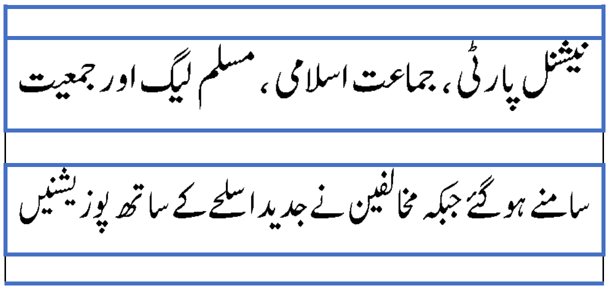


FIGURE 2. **Urdu Nastaliq scripts found in outdoor-natural environment.**

styles are found in natural outdoor images, with complex backgrounds, orientations, illumination as shown in Figure 2. In the context of a training machine learning algorithm from given Urdu text images, constructing annotation of these images are a difficult task due to multiple ligatures and multi-line ligatures. Traditionally projection profiles [18], [19], Connected Component [3], [20], or MSER [21]–[23] based segmentation has been done to partition digital images into multiple segments of text or non-text regions. Text segmentation/detection is thus inevitable for further recognition of text.

Much of the current research on Urdu recognition is performed on the cleaned and segmented artificially generated Urdu Nastaliq text such as Urdu Printed Text Images (UPTI) [24], custom extracted [15], generated text with clear background [25], video tickers [26] or handwritten Urdu text [27] as opposed to extracting from outdoor or real-world images with complex background. This work is a step in that direction that integrates synthetic Urdu-text in natural outdoor images. The word text/script is often used interchangeably in the literature.


نیشنل پارٹی، جماعت اسلامی، مسلم لیگ اور جمیعت

سامنے ہوگئے جبکہ مخالفین نے جدید اسلحے کے ساتھ پوزیشنیں

FIGURE 3. **Samples from UPTI dataset.**

Figure 3 shows the samples from the UPTI dataset, where the text has been given in a clear white background. For pure outdoor Urdu-text detection, we were only able to find one relevant study [16]. Chandio *et al.* [11] and Ali *et al.* [28] used manually cropped Urdu characters from natural outdoor scenes for recognition of Urdu-characters, while in [16] Asghar *et al.* used custom annotated Urdu images with ICDAR2017-MLT Arabic text for detection purpose, they used a combination of their custom collection of Urdu images and Arabic images from ICDAR 2017-MLT [7], They have dealt with Urdu and Arabic text at the word level.

TABLE 1. Comparison of detection results on artificial urdu text dataset [30].

Methods /Study	Year	No. of Images (Urdu)	Precision	Recall	F-Score
Siddiqi et.al. [13]	2012	1000	0.71	0.8	0.75
Raza et.al. [14]	2013	1000	0.75	0.86	0.80
Raza et.al. [4]	2013	1000	0.80	**0.89**	0.84
Jamil et.al. [15]	2016	100	0.58	0.84	0.69
Salahuddin et.al.[5]	2018	1000	**0.83**	0.88	**0.85**
Zunera et. Al. [6]	2018	1000	0.72	0.89	0.80

A similar kind of study is done by Ahmed *et al.* [29] and Oulladji *et al.* [30] for Arabic outdoor text. Similarly, numerous studies [3], [31]–[36] exist for the recognition of Urdu and give comparisons to prove the worthiness of their research. Besides custom used datasets [6], there is only one famous Artificial Urdu dataset for text detection, and that is Artificial-Urdu-Text-Dataset [37], that is openly available. It consists of 1000 video frames from news channels and is openly available. Table 1 summarizes the results of Detection on artificial Urdu text in video frames, and the highest result achieved so far, has a Precision of 0.83 [5].

While for recognition of Urdu characters from outdoor images there are few custom datasets [11], [15], [25] and for recognition of printed characters words there is a famous dataset UPTI [24], which recently has been updated and has been presented with name UPTI2.0 [38] because the performance on UPTI has reached near saturation [33], [35]. There also exist CLE-18000 [32], [39] which contains near 18K ligatures (compound characters). Others [11], [25], [31], have used custom cropped ligatures for recognition purposes [11].

As datasets are the benchmark for evaluating the accuracy of algorithms, Table 2 summarizes the frequently used datasets in Urdu text detection and recognition along with the state-of-the-art datasets used for English and other languages. It can be seen that the English Language has the most available datasets [40]–[43] with the text style of a horizontal, oriented, and curved text. Also, English datasets with the multilingual text [7], [8], [15], [44] cover other languages than Urdu text. The number of images in English datasets varies from 500 to more than 60K. Although researchers have mentioned Urdu text datasets [6], [15], [16], [37] for detection and recognition, only one dataset [37] is publicly available for text detection in video frame images. Also, scholars [45], [46] have done bilingual recognition on MSRA-TD [43] and other datasets.

This paper performs a comprehensive analysis of CNN features for determining their ability and effectiveness for Urdu text detection in outdoor pictures at the ligature level, the orientations of ligatures, and the recognition of individual characters in a text. The scripting style, which is focused on in this study is Nastaliq. To perform a balanced analysis using CNN features, a synthetic dataset with two variations of ligatures is created. The main contributions of this paper are as follow;

- The development of datasets with embedded Urdu text in natural scene images. These synthetic images contain 132 and 1600 unique ligatures respectively for 4.2k and 51k images, with 32 variations of each ligature. It means each ligature is projected to 4-backgrounds with 8-color variations of font.
- Comparison of CNN model's features for evaluating their ability to detect Urdu text using the larger image-input layer FasterRCNN algorithm. Convolutional features of well known CNNs, that is Squeeznet, Googlenet, Resnet18, and Resnet50 are used for comparison.
- The first study in Urdu text/script detection literature, which has used four different kinds of CNNs features for detection purposes.
- Prediction and Correction of Oriented Urdu text in images.
- TSDNN based recognition of Urdu Ligatures using Resnet50, googlenet features and BLSTM for both synthetic and real outdoor images

The rest of the paper is organized as follows. Section 2 describes the proposed methodology, datasets, various feature extractors used, the evaluation protocol, RRNN, and TSDNN. Section 3 discusses the experimental results and Section 4 gives general discussion. while finally Section 5 concludes the paper and gives future directions.

II. PROPOSED METHODOLOGY

The proposed methodology consists of four main modules. First is 'Synthetic Dataset Generation', Second is Custom 'FasterRCNN builder', Third is 'Regression Residual Neural Network' for Urdu text angle prediction. While fourth is Two Stream Deep Neural Network (TSDNN) for recognizing the sequence of characters in an Urdu text.

First Module Generates five types of Synthetic dataset (SD): SDAi, SDA and SDB, SDAi embeds CLE text ligatures as a small image at random locations in Other images of size 320 × 240. While in SDB images of small size with

TABLE 2. Notable datasets for text detection.

Year	Dataset	Scenic / Artificial	Task Type	Language	Orientation Types	No. of Images
2018	Custom [6]	Frames	Detection	Urdu		2000
2012	Artificial Urdu Text Dataset [37]	Artificial	Detection	Urdu	Horizontal	1000
2011	Custom [15]	Artificial	Detection + Script Identification	Urdu, English, Arabic, Hindi, Chinese	Horizontal	500
2015	ICDAR2015 [8]	scenic	Detection+ Recognition	English + Chinese	Horizontal, Multi-oriented	1,670
2012	MSRA-TD500 [43]	Scenic + indoor	Detection	English	Horizontal, Multi-oriented / rotated	500
2017	ICDAR2017 [7]	scenic	Detection+ Recognition	Arabic, Chinese, English, French, German, Italian, Indian, Japanese, and Korean	-	9000(detection) + 84,868 cropped word images
2016	COCO-Text [44]	Scenic + mix	Detection + Recognition	English + non-English Script	-	63,000 images
2017	Total-Text [42]	scenic	Detection	English	Horizontal, Multi-oriented, Curve	1555
2017	CTW-1500 (Curve text in a wild) [41]	scenic/wild	Detection	English	Curve + horizontal	1500
2003	ICDAR2003 [40]	Mix	Detection + Recognition	English	Horizontal	600
2018	Custom [11]	scenic	Recognition	Urdu (characters)	Horizontal	18000
2019	[16] Custom + ICDAR2017	Scenic/ mix	Detection	Urdu + Arabic (word level)	Horizontal, Oriented, Curve	1100+1000
2019	EASTR-41K [47]	Scenic	Recognition	Arabic + English	Horizontal	2469
2020	**Proposed**	Scenic	Detection + Orientation + Recognition	Urdu	Horizontal, Oriented	51200

rotated Urdu text embedded in it. Ligatures are rotated in the range of $-50°$ to $50°$. SDA further has two sets SDA4 and SDA51 corresponding to 4.2K and 51K images respectively for recognizing the sequence of Urdu characters. Similarly, SDB has two sets SDB4 and SDB51 corresponding to 4.2K and 51K rotated ligature images. The holdout ratio of 0.9 was chosen for all sets. SDAi also has 4.2K images.

The second Module consists of a Custom FasterRCNN builder, which generates four different types of FasterRCNNs. In it, two sub-modules, a CNN model-selection module and the other is the training module, which trains FasterRCNN using the features from the selected CNN model. This module has two inputs, one is an image and the other is the location of Urdu text in that image. Each CNN input layer is changed according to parameters as mentioned in column-5 of Table 3, it is a preprocessing step before any further processing by the second module. Figure 4 illustrates the working of the proposed methodology. Initially, an input image with annotation of the bounding box (rectangular coordinates) is given along with the selected CNN model. The bounding box(es) identifies the region of interest (ROI) in CNN features. With these bounding boxes, constrained features from training images are selected. Then FasterRCNN trains a classifier and a regressor/bounding box predictor for given regions. Classifier classifies the region as foreground (text) or as background. While on the other hand, regressor modules learn to predict the positions of potential bounding boxes in each image. The third module is a Regression

TABLE 3. Summary of pretrained CNN models for feature extraction.

Model Name	No. of Layers Originally*	No of Layers after conversion to FasterRCNN*	Feature Extraction Layer	Input Size
squeezenet	68	78	'fire5-concat'	[227 227] -> 320x240
resnet18	72	83	'pool5'	[224 224] -> 320x240
resnet50	177	188	'activation_40_relu'	[224 224] -> 320x240
googlenet	144	155	'inception_4d-output'	[224 224] -> 320x240

*as per implementation in Matlab

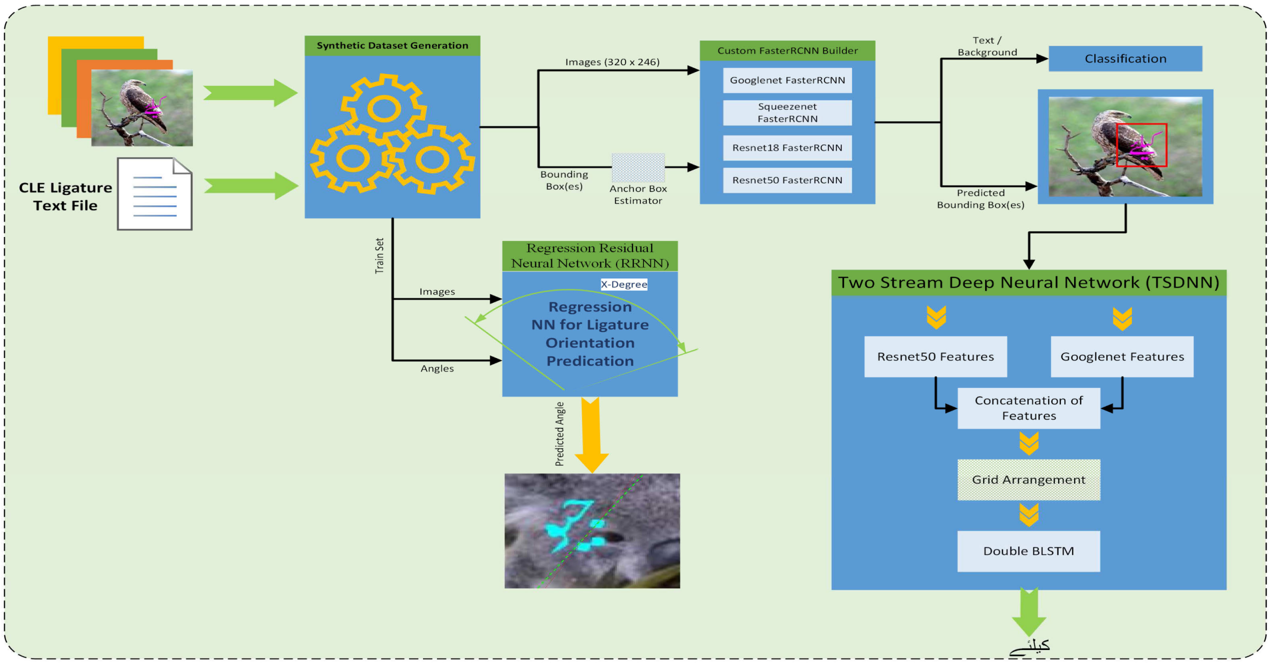


FIGURE 4. The proposed methodology for Ligature Detection, Orientation, and Recognition of Urdu text.

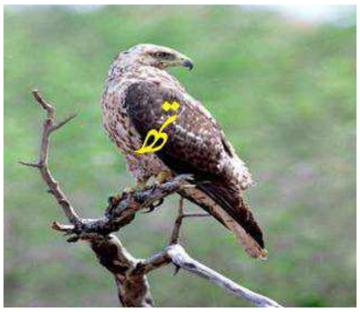
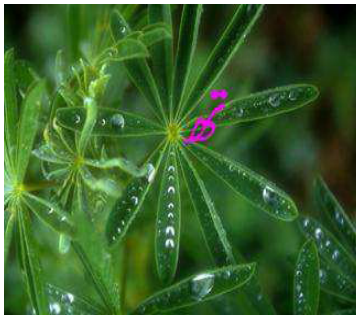
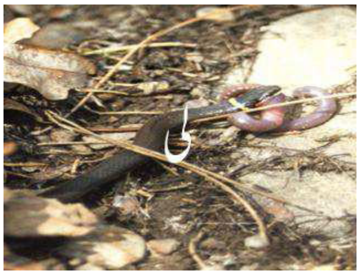
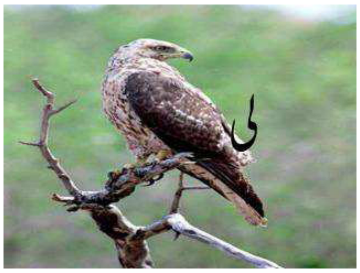
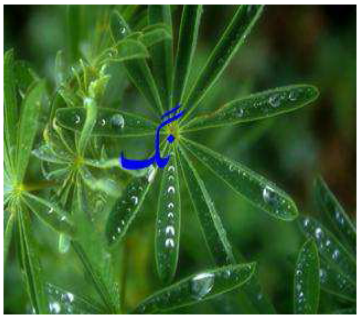
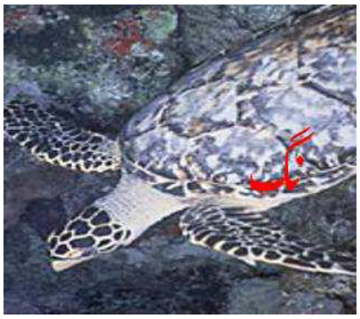
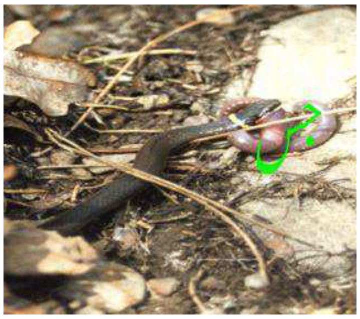
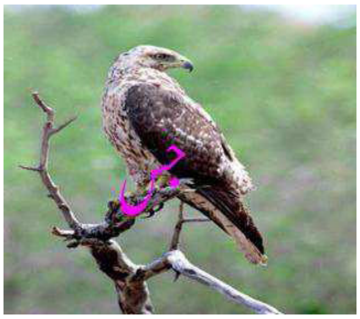
Residual Neural Network (RRNN) for predicting the angle of rotation of a given ligature. Which is trained to predict the orientation of written Urdu ligature. The fourth module is Two Stream Deep Neural Network (TSDNN), that can take the detected or ground truth rectangles in the image and recognizes the text/ligature written in that image, essentially, it's a kind of sequence to sequence classifier.

In the end, accuracies are calculated for each module as per measures being used in the computer vision field.

A. DATASET PREPARATION FOR DETECTION

To train any text detector presence of quality and uniform training data is the key thing, and it results in a good quality detector. For our comparison proposes, we have generated more than 4.2K images (SDAi) of more than 132 unique ligature images/classes from the CLE [39] dataset's textual annotation. It needs to be noted that the originally CLE dataset has ligature images with a black background and varying aspect ratio. Few images from the CLE dataset are shown in Figure 6. We just choose textual annotation of each ligature from CLE, then for each ligature, a transparent ligature-image is generated which is embedded in an outdoor natural scene image with 28 variations at random locations. These variations are created by using 4-different backgrounds and using Nastaliq fonts in 8-different colors namely 'cyan', 'white', 'red', 'green', 'blue', 'black', 'magenta', and 'yellow'.

TABLE 4. Synthetically generated ligature variations with different backgrounds and color fonts.

Synthetic-Image with Ligature	Transcription	Synthetic-Image with Ligature	Transcription
	تھ		تھ
	لی		لی
	نگ		نگ
	جس		جس

The dataset (SDAi) is divided into 2-parts. The first part consists of 3801 training images, while the second part consists of 423-test images. The hold out ratio is 90% and 10% respectively for training and testing images. Synthetically generated Ligature variations with different backgrounds and color fonts from our dataset are shown in Table 4.

B. CUSTOM FEATURE FASTERRCNN BUILDER

A Convolutional Neural Networks (CNNs) or Deep Neural Networks (DNNs) are a special class of neural networks, that are primarily developed to identify, locate and recognize visual features directly from 1D, 2D or ND matrices. In this study, image pixels serve as an input to CNNs. Four

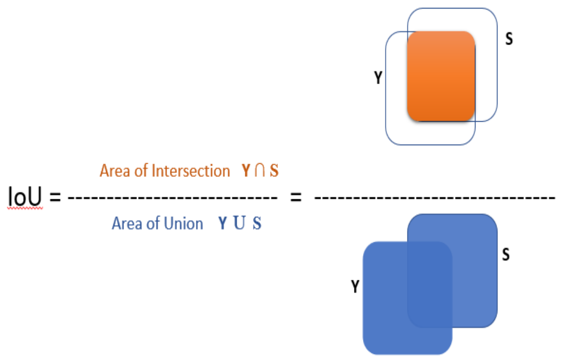


FIGURE 5. IoU Venn diagram.

existing deep neural networks, namely Squeeznet, Googlenet, Resnet18, and Resnet50 were used as a feature extractor for training a FasterRCNN algorithm. The detail of each CNN model is given in Table 3 that shows the layer's name, from where the features are taken for further processing by FasterRCNN. It also shows the size of the query image to be resized before feature extraction by the corresponding layer. The Four FasterRCNNs are trained for 20-epochs each using stochastic gradient descent with momentum (SGDM) algorithm for weights update. The learning rate was set to 0.001 for the entire 20 epochs. An overlap ratio of 0.5 to 1 is used as a positive sample for learning. The anchor boxes which are chosen for training for all FasterRCNNs were [67], [57]; [136,116]; [272,232]. The hold-out ratio was kept 0.9, resulting in 3801 images as a train-set and 423 images as test-set. Train-set images were used for training, while test-set images were used for testing the detection accuracy.

Table 3 also mentions the number of layers in each of the selected CNN, along with the name of layers used for extracting the CNN features, necessary for training the FasterRCNN. The detailed inner working of this module can be understood by Algorithm 2.

1) SQUEEZENET

Squeezenet is a small deep neural network with only 5MB size. It has an 80.3% accuracy in top-5 for imagenet [48] and was released in 2016. Squeezenet was developed goal was to create a smaller neural network with fewer parameters that can more easily fit into mobile computers and easily transmittable over a network [49] and Squeezenet Architecture is beautifully explained in [50]. The main idea in Squeezenet is squeeze and expand blocks.

In Squeeze block, 1×1(point-wise) filters are used to replace 3×3 filters, and in expand block we use 1×1 filters as a bottleneck layer to reduce the depth of the computation by following 3×3 filters. Combinedly these two blocks are called fire module as shown in Figure 7. A Squeezenet layer consists of layers of fire modules and several max-pooling layers plus the global average pooling layer. Feature map size remains the same due to the squeeze-layer and expand-layer of the fire module. While, on the other hand, pooling reduces the depth to a smaller number. The network expects an input of 227×227 pixels.

2) RESNET18

Resnet stands for Residual Network and is based on the concept of skip module [51] as shown in Figure 8. Residual networks are constructed by utilizing skip connections or jumps over some layers and connect layers feature values. Typically, there are single layer jumps or skips. An additional weight matrix may be used to learn the skip weights; these models are known as HighwayNets. ResNet-18 [52] is a convolutional neural network (CNN) that is trained on more than a million images from the ImageNet [48] dataset. The network originally had a depth of 18 layers and its architecture is graphically explained in [53]. It can classify images into 1000 object categories. Objects that are classified by this net are common household items, living things such as a keyboard, remote, TV, mouse, pencil, and many animals. After training through ImageNet for so many examples, this network has learned rich feature representations for a wide range of images. The network expects an input of 224×224 pixels.

3) RESNET50

ResNet-50 is also a CNN, that is trained on more than a million images from the ImageNet [48] dataset. It is an

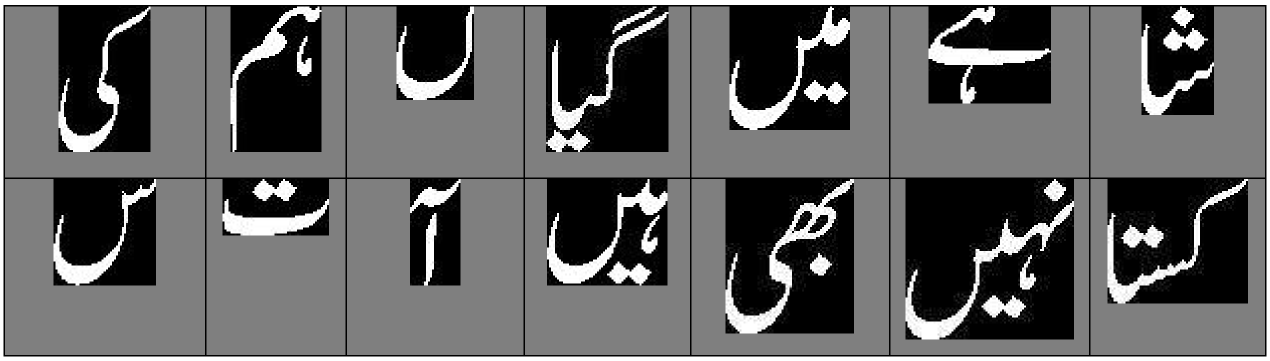


FIGURE 6. Sample images from CLE dataset.

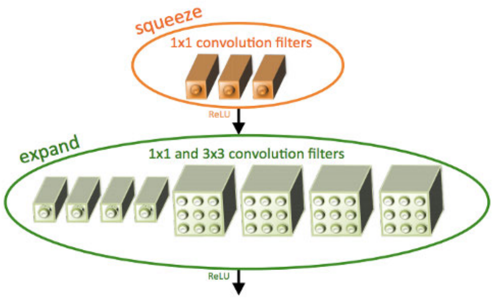


FIGURE 7. **Configuration of convolutional modules in Fire-Module of Squeezenet architecture [50].**

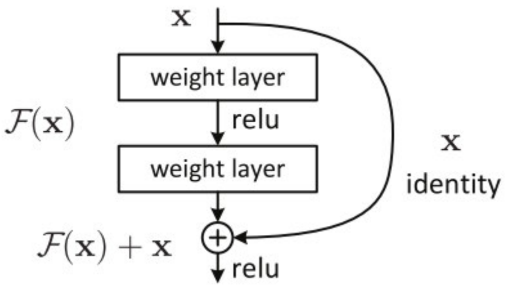

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

FIGURE 8. **Resnet skip module architecture [51].**

extension of Resnet18. The network originally consisted of 50 layers [54] and can classify images into 1000 object categories. These categories contain common life usable things such as a keyboard, remote, TV, mouse, pencil, and animals. Due to training on millions of images, the network has learned rich feature representations like other mentioned CNNs. The network expects an input image of the size 224-by-224 [53]. In its architecture, researchers have connected the previous or earlier outputs after bypassing 2-layers and are applied to continue till before FC-layer (Fully Connected). Bypassing 2 layers is a key intuition, as per authors bypassing a single layer did not give many improvements. Two-layer jump between layers can be thought of as a Network-In-Network model.

4) GOOGLENET

Googlenet [55] is 144-layers pre-trained convolutional neural network. It is the winner of the ILSVRC 2014 competition. It was developed by Google team, and sometimes known as Inception V1. Googlenet had a top-5 error rate of 6.67% in the competition. This error-rate was very close to the human-level performance. And was a challenge for competition organizers to evaluate the results. As its performance was rated against human's performance. These days, we can have pretrained Googlenet trained on 2-kind of datasets, it is either the ImageNet [48] or Places365 [56] dataset. The first network was trained on ImageNet like other mentioned CNNs and can classify images into 1000 classes. On the other hand, a variant model is also available that was similarly trained on Places365 as on ImageNet, but it classifies images into 365 different place categories. The categories in this dataset are field, park, runway, lobby, etc.

Algorithm 1 TSDNN testing Algorithm

Algorithm for Training

```
I)  For each training image $\Delta$ in $\rightarrow$ SDA (TrainSet)
    a) Get Googlenet features $\rightarrow \lambdabar\Delta$
    b) Get Resnet50 features $\rightarrow \omega\Delta$
    c) Get image label ligatures $\rightarrow \xi\Delta$
    d) Integrate step-1 & step-2 features
       $\delta \leftarrow [\lambdabar\Delta\ \omega\Delta]$
    e) Standardize $\delta$ with zero paddings $\rightarrow \delta$'
    f) Convert $\delta$' features to a Model compatible
       format $\sigma \leftarrow \Psi_1$ ($\delta$')
    g) Convert the label of the image to a Model
       compatible format $\rho \leftarrow \Psi 2$ ($\xi\Delta$)
II) Train the TSDNN using $\sigma$, $\rho$ as an input-features
    and target vectors respectively
```

C. FASTERRCNN

FasterRCNN [57] detector is an upgrade to an already existing R-CNN detector [58] and Fast-RCNN [59], [60], both these algorithms use Edge Boxes [61] algorithm to generate region proposals for potential text regions. The working of Faster-RCNN is different from R-CNN, and Fast-RCNN as it uses Region Proposal Network (RPN) to generate region proposals directly as part of the network. It means in it RPN is used as a substitute for the Edge Boxes algorithm.

The time for generating region proposals in FasterRCNN is much smaller than the edge box algorithm. Succinctly, the ranking of anchor boxes is done by RPN which indicates the most likely anchor boxes, which are most likely to contain objects of interest. So, generating region proposals is faster in FasterRCNN and is better adjusted to input data. Faster-RCNN produces two types of output, one is the classification category and the other is coordinates of predicted rectangles. Basic components and working mechanisms of FasterRCNN are nicely depicted in [60].

D. EVALUATION PROTOCOL (AVERAGE PRECISION)

Average precision (AP) or mean average precision (mAP) is the most commonly used metric in measuring the accuracy of object detectors like R-CNN, Faster YOLO [62] also known as YoloV2, SSD [63], etc. Also, in projects involving information retrieval, researchers come across familiar terms. mAP is the average of AP. In it, the AP for each class is calculated and then their values are averaged. Under the Common Objects in Context (COCO) [44] text context, there is no difference between AP and mAP. Average precision computes the average precision value for recall value over 0 to 1.

In the PASCAL Visual Object Classes (VOC) [65] and MS COCO challenge [44], AP is appropriate and commonly used metrics for object detection and information retrieval tasks.

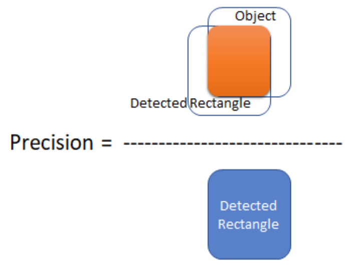


FIGURE 9. Geometrical description of Precision.

In our context of Urdu text detection, AP depends upon the concept of IoU (intersection over a union of rectangles or bounding boxes) as shown in Figure 5.

Y in Figure 5 is representing ground-truth rectangle and S is representing a detected/predicted rectangle containing text. The final decision regarding text is detected or not decided according to the following condition.

```
IF
        IoU > $\varphi$ [True or Text Detected]
ELSE
        [False or Text Not Detected]
END
```

Here φ is representing a certain threshold of say 0.5. If the rectangle is correctly identified against an annotation, then it's a True-Positive (TP). If the rectangle is predicted but there is no text, then its false-positive (FP).

$$precision = \frac{Intersection\ of\ GT(Object)\ and\ Predicted\ Rtectangle}{Predicted\ Rtectangle} \quad (1)$$

Also, if there is a text in an image according to annotation but the system doesn't recognize that text, in that case, it is false-negative (FN). Mathematically Eq. 1 shows precision, here GT represents ground truth object and the other is predicted/detected rectangle. Also, it can be understood easily in graphical form, as shown in Figure 9.

For calculating AP, we iterate through all image queries as per Eq. 1. While for calculating mAP, Eq. 2 can be used. Here Q is the number of image queries. Average P(qi) is the average precision (AP) for a given query image category, qi (as a single image can contain multiple detections). It simply means that, for a given query image category, qi, we compute its corresponding AP for each category, and then the mean of each category across all query images. Then all AP scores would give us a single number, and that is called mAP. It quantitatively describes how good the trained model is for detecting rectangles with respect to ground truth rectangle coordinates.

$$\text{mAP} := \sum\nolimits_{i=1}^{Q} Average\ P(q_i)/Q \quad (2)$$

So, AP measures the accuracy of are your predictions. i.e. the percentage of your positive predictions that are correct from retrieved corpus [66] .

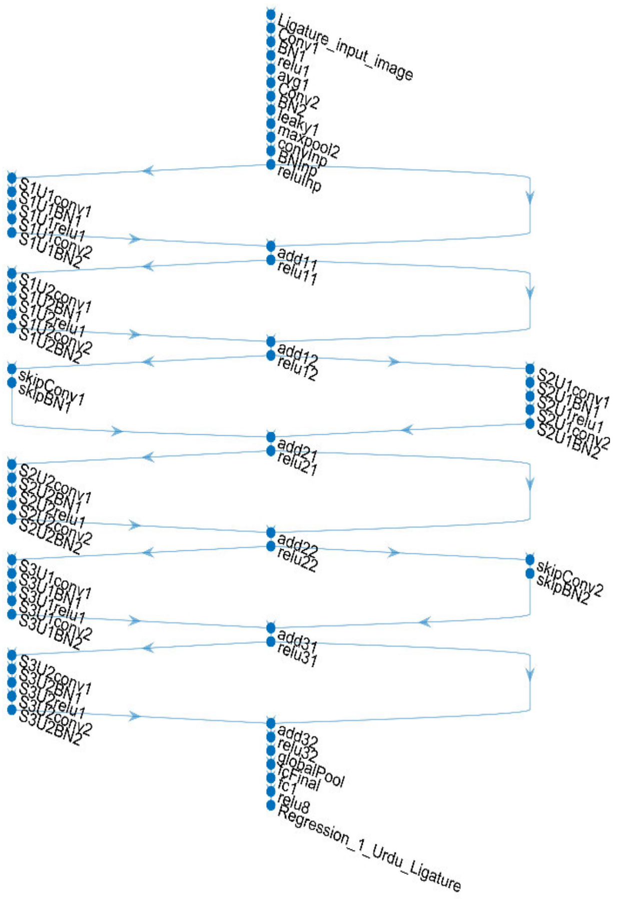


FIGURE 10. 63-Layer Regression Residual Neural Network (RRNN).

E. LIGATURE ORIENTATION PREDICTION

A residual classification network is adapted to work as a regression residual neural network (RRNN) to learn the ligature's angular rotations [66]. 63-Layer Residual neural network with the Regression layer is shown in Figure 10. RRNN has residual or direct connections from preceding layers similar to Resnet architecture, that bypass the normal flow of parameter values through network layers to forward layers. These residual/shortcut connections enable the gradients to propagate more easily from the regression layer to the Upper layers of the network during the training phase. That makes it possible for those layers to robustly learn desired features for a given problem. The RRNN with an image input layer of size 140 $\times$ 100 was trained on 3801 images and Tested on 423 images (SDB4), and it is also trained on 46036 images and Tested on 5115 images (SDB51). The learning rate of 1e-3 and minibatch size of 50 was chosen for training. The predictions of the model are shown in Figure 11, where the green-line shows the ground truth angle, while the red-line shows the predicted angle for a given ligature image. Various predictions for 4-ligature images are given.

As Detected Ligature's angle of rotation can be determined by the RRNN, their orientation can also be corrected as shown

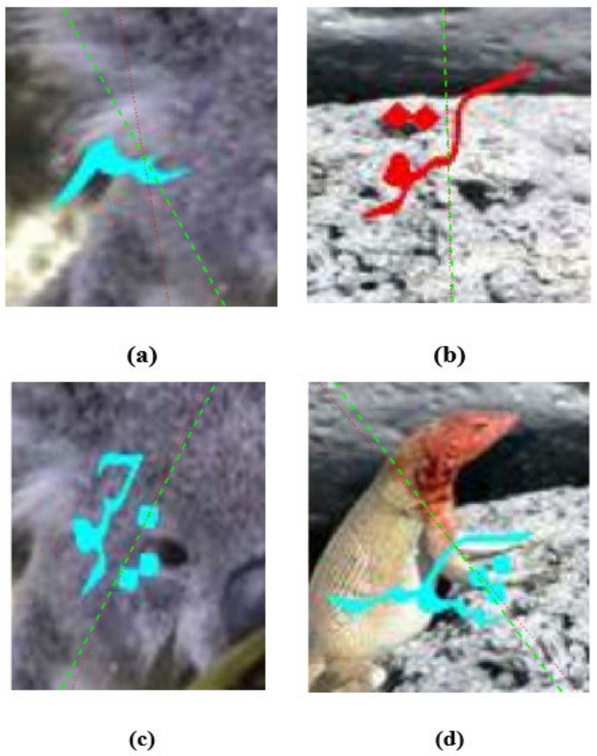


FIGURE 11. Green line indicates Ground-truth (GT), Red line indicates prediction(P): (a)GT-30° P-10° (b)GT-02° P-01° (c)GT31° P-27° (d)GT-39° P-43°.

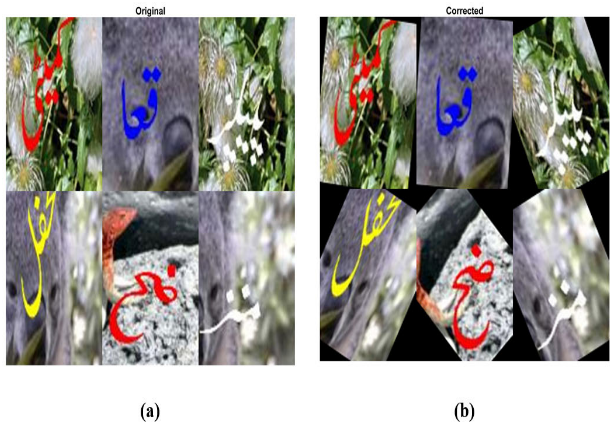


FIGURE 12. Rotated ligatures and corresponding angle rotation corrected ligatures.

in Figure 12. Oriented ligatures can be seen on the right side of the figure, where tile of images either left or right can be seen against a black background. These corrected ligatures then can be fed to OCR for improved recognition of text. The model can correct ligatures rotation between the range of $-50°$ to 50°.

F. LIGATURE RECOGNITION

The Two Stream Deep Neural Network (TSDNN) module in the proposed methodology is inspired by [67], with the difference of handling color images, number of features, and using CNN features found to robust for Urdu text detection, i.e., googlenet and resnet50. These two CNNs are selected based on their detection performance in the detection step in our experimentation. The length of the feature vector obtained from googlenet is 1×104388, and resnet50 gives a feature of length 1×200704. When these two features are integrated, they give a feature vector of 1×304192. Later features extracted from both these CNNs are arranged into a grid pattern of 4992×64, which have 319488 cells or values, including zero paddings for values beyond a true number of feature values. Here 64 corresponding to 32 Unicode characters needs to be learned. It means 2 values per Unicode serve as training in the network.

They are arranged so that features have positional dependency which needs to be learned in the later stages by double-layer BLSTM. Generally features extracted in earlier steps, are considered as a sequence of inputs σ, and corresponding ligature labels as a sequence of required target vectors as ρ. In our case, ρ has a maximum length of 32 primary Urdu characters inclusive of spaces. If a ligature length is less than 32 characters, then the remaining space is treated as a space character. The learning rate of 0.0001 and minibatch sizes of 40 and 200 is selected corresponding to SDA4 and SDA51 images models. TSDNN module shown in Figure 4 succinctly describes the main components in this module. Also, Algorithm 1 describes the steps required for training TSDNN model, while Algorithm 3 describes steps to recognize the Urdu ligature text in test-set images.

III. EXPERIMENTAL RESULTS

To vindicate the proposed methodology, the potential of the methodology is demonstrated through experiments on each submodule by tuning the parameters according to various modules. The brief description of conducted experiments are described below. The proposed methodology was implemented using Matlab and ran on NVidia GTX1070 GPU based hardware. Two sets of synthetic datasets were used, First A total of 4.2K ligature images were processed and secondly, 51k ligature images were processed to evaluated different modules.

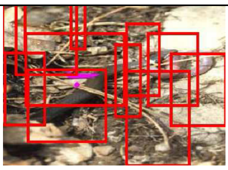
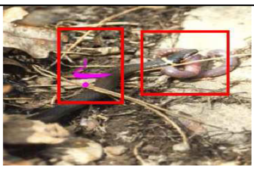
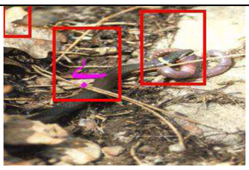
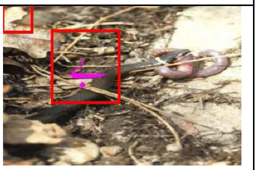
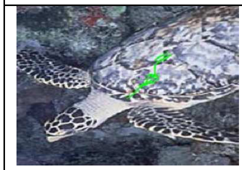
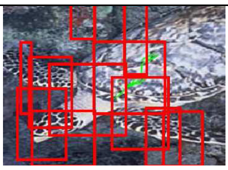
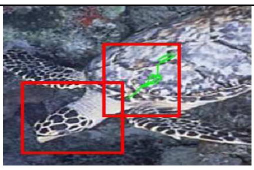
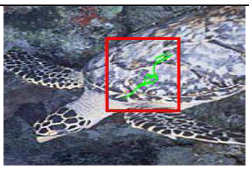
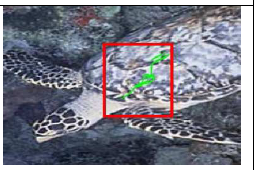
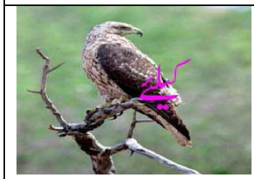
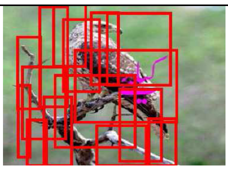
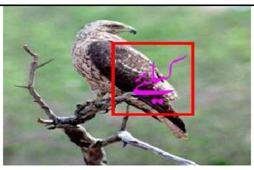
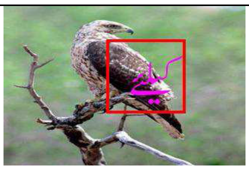
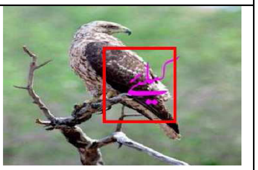
A. DETECTION OF LIGATURES

Synthetic outdoor text dataset images (SDAi) were presented to four trained FasterRCNNs. Each FasterRCNN model was trained on different well known CNN's extracted features. Each trained model produced its output in the form of rectangles or bounding boxes on the input images. Table 5 shows the prediction of bounding boxes for similar ligature input query image, using dissimilar CNN model features. Column-1 shows the query image containing the ligature. While columns 2nd-to-5th shows the predicted rectangles by FasterRCNN corresponding to different feature extractors. It can be seen that column-2 has multiple detections for Squeeznet feature-based FasterRCNN.

Algorithm 2 Algorithm for Urdu-Text Detection(Custom FasterRCNN module)

```
Algorithm for Urdu-Text Detection
START
INPUT: NN, annotation (location)
OUTPUT: Rectangular coordinates of Urdu-text, MFstGoolgenet, MFstSqueezenet,
        MFstResnet18, MFstResnet50, t_google, t_squeez, t_res18, t_res20
        Ap_google, Ap_squeeze, Ap_res18, Ap_res50
        NN: Number of images containing embedded Urdu-text
        annotation (location): rectangular coordinates of text in images
        Rectangular coordinates of Urdu-text: Urdu text location
        MFstGoolgenet: Faster RCNN model based on Googlenet Features
        MFstSqueezenet: Faster RCNN model based on Squeezenet Features
        MFstResnet18: Faster RCNN model based on Resnet18 Features
        MFstResnet50: Faster RCNN model based on Resnet 50 Features
        PreTexLoc: function Predict the Text-Location rectangle coordinates
imageSize ← [320 240]
// Anchor box estimation
$\alpha$ ← AnchorsEstimation (NN, annotation)
// Custom FasterRCNN Module
MFstGoolgenet ← ConstructCustomGoogleFasterRCNN (imageSize, $\alpha$)
MFstSqueezenet ← ConstructCustomSqueezeFasterRCNN (imageSize, $\alpha$)
MFstResnet18 ← ConstructCustomResnet18FasterRCNN (imageSize, $\alpha$)
MFstResnet50 ← ConstructCustomResnet50FasterRCNN (imageSize, $\alpha$)
[ $\theta$-tr, $\theta$-ts] ← division of NN into train and test set
// Text Detection Training Module
For each training image i in → $\theta$-tr
        a) Extract Googlenet features → $\lambda i$
        b) Extract Squeezenet features → $\omega i$
        c) Extract Resnet18 features → $\rho i$
        d) Extract Resnet50 features → $\xi i$
End For
Train Model MFstGoolgenet for features $\lambda i$, and compute training time t_google
Train Model MFstSqueezenet for features $\omega i$, and compute training time t_squeeze
Train Model MFstResnet18 for features $\rho i$, and compute training time t_res18
Train Model MFstResnet50 for features $\xi i$, and compute training time t_res20
$\eta$_google ← PreTexLoc($\lambda i$)
$\eta$_squeeze ← PreTexLoc($\omega i$)
$\eta$_res18 ← PreTexLoc( $\rho i$)
$\eta$_res50 ← PreTexLoc( $\xi i$)
Ap_google ← Evaluate_AP( MFstGoolgenet, $\eta$_google)
Ap_squeeze ← Evaluate_AP(MFstSqueezenet, $\eta$_ squeeze)
Ap_res18 ← Evaluate_AP(MFstResnet18, $\eta$_ res18)
Ap_res50 ← Evaluate_AP(MFstResnet50, $\eta$_ res50)
For each testing image I in → $\theta$-ts
        a) Extract features using any trained model £→ $\varphi I$
        b) [rectangle_coordinates, objectness_ score, category] ← Predict ($\varphi I$)
        c) Display image along with rectangle_coordinates, category
        d) $\Theta$ ← [$\Theta$ rectangle_coordinates]
End For
Ap_£ ← Evaluate model £ using $\Theta$
FINISH.
```

TABLE 5. Detection results for different input query images.

	Detection Results Based on different CNN Features			
Query Image	*Squeezenet*	*Googlenet*	*Resnet18*	*Resnet50*
				
				
				

```
Algorithm 3 TSDNN Testing Algorithm
For each query image I → SDA (TestSet)
  a) Get Googlenet features → $\Phi$I
  b) Get Resnet50 features → $\omega$I
  c) Integrate step-1 & step-2 features $\eta \leftarrow [\Phi I\ \omega I]$
  d) Standardize length of $\eta$ features → $\varepsilon$
  e) Convert $\varepsilon$ features to a Model compatible format
     $\varphi \leftarrow \Psi_1(\varepsilon)$ )
  f) Predict the sequence (decode) using Trained Model
     $PredSeq \leftarrow \rho$ ( Net, $\varphi$)
  g) Convert a predicted sequence of step-6 to Unicode
     UniCo ← $\chi$ (PredSeq)
  h) Editable Ligature Generation → $\xi$ (UniCo)
```

Googlenet and Resnet18 CNN features have little to no multiple detections. Their predictions rectangles have wide size as compare to Resnet50 based FasterRCNN detector. Resnet50 based ligature detections are center-aligned in detected rectangles and have little extra detection area. In the experiments, the dataset was divided into two parts, train-set, and test-set.

The number of images in the train-set was 3801 and 423 in the test-set. FasterRCNN [60] took different times to train each model. Resnet18 and Squeezenet took 16.85 and 16.19 hours respectively to train. Googlnet took 25 hours, while Resnet50 took the 118 hours to train which is the largest among the Four tested algorithms. Similarly, at the other end of the spectrum, Squeezenet takes less time to train among all. Resnet50 based FasterRCNN took the most time to train as it contains the largest numbers of learning layers, while Squeeznet has the least number of layers. For all four CNN models, average precision (AP) was calculated for both the train and test set. The best detection results were obtained by using Resnet50.

This study achieved the AP of .9925 for the train-set while AP of .9812 on test-set for Resnet50. Resnet18 (test AP.8782) and Googlenet (test AP .8836) can be ranked 2nd and 3rd in terms of their ability for text detection respectively. Albeit Resnet18 (train AP .9366) is higher than Googlenet (train AP .8816). The worst test-set AP of.6585 was obtained by using Squeeznet. Train-set Average precision reflects similar pattern results. Although, Squeezenet was created with fewer parameters and has a small size for use in mobile computers. It appears that fewer parameters may be right for classification purposes, but it's not a suitable feature extractor for Urdu script/text detection purposes. Figure 13 graphically summarizes the results of our experiments and the overall performance of all 4-CNN models on both the train and test set, where resnet50 performance lines for train-set and test-set can be seen as the top performer with the most time required for training. Similarly, squeeznet can be seen as the worst performer for the Urdu text detection problem.

B. ORIENTATION OF LIGATURES

For judging the robustness of RRNN, two types of experiments were done. The number of training and testing

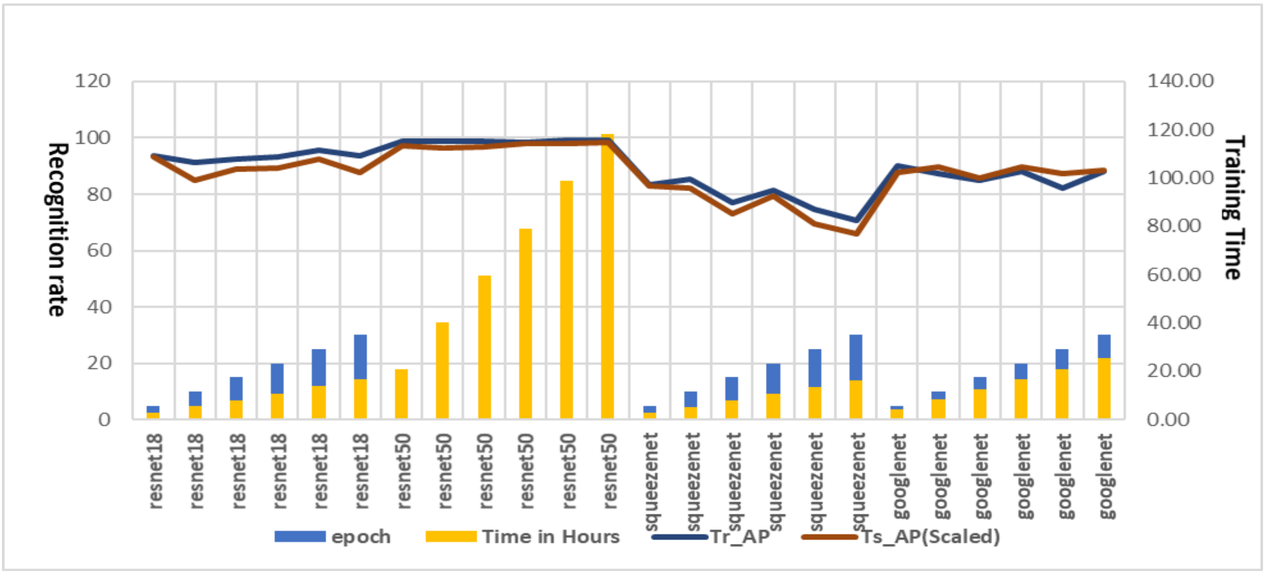


FIGURE 13. Comparison of Urdu ligature detection accuracy using a different kind of CNN features.

TABLE 6. Angle prediction training and testing accuracy using SDB4.

Ligatures Type	No of Total Ligatures	Epochs	Accuracy with Th	Test Accuracy
Trained-Set	3801 (3.8K)	20	3	79.70%
Test-Set	423 (0.423K)	20	3	55.34%
Trained-Set	3801 (3.8K)	20	**5**	**95.16%**
Test-Set	423(0.423K)	20	**5**	**79.33%**

TABLE 7. Angle prediction training and testing accuracy SDB51.

Ligatures Type	No of Total Ligatures	Epochs	Threshold Th	Test Accuracy
Trained-Set	46036 (46K)	30	3	92.37%
Test-Set	5115 (5K)	30	3	97.77%
Trained-Set	46036 (46K)	30	5	**99.99%**
Test-Set	5115 (5K)	30	5	**99.06%**

images vary in each experiment. In one experiment RRNN was trained using 3.8k images (SDB4) and evaluated using Threshold Th of 3 for evaluating the .421k images. In it, an accuracy of 79.70% for training set and 55.34% testing accuracy was achieved. While with a threshold Th of 5, training and testing accuracies were 95.16% and 79.33% respectively.

In another experiment, 51k (SDB51) images were used. The number of images in the train-set was 46036 and 5115 were in the test-set. RRNN was again evaluated using Threshold Th of 3 for evaluating the 51k images. In it, an accuracy of 92.37% for training set and 97.77% testing accuracy was achieved. While with a threshold of 5, training and testing accuracies were 99.99% and 99.06% respectively. These results show more training and data improves accuracy. Table 6 and Table 7 depicts these results.

TABLE 8. Accuracy as per two different measures.

Urdu Text	Ligature	One-to-one Accuracy	Ligature	Partial Match Accuracy
Original Text →	جنم	-	جنم	-
Predicted text →	جیم	0 %	جیم	66%
Original Text →	سفید	-	سفید	-
Predicted text →	سمید	0%	سمید	75%

C. RECOGNITION OF LIGATURES (SYNTHETIC DATASET)

Ligatures/Urdu text are considered as a sequence of primary characters in Urdu script with a maximum of 32 basic isolated characters. Recognition rate based on two measures i.e., exact and partial sequence matching was calculated for given ligature images. Sometimes sequence may match partially, meaning they may differ by only one or more characters, then they are not considered as a positive match for exact or one-to-one but do for partial match. So, the partial sequences of recognized characters or Urdu ligatures were matched by Levenshtein-distance [68], also sometimes known as edit-distance [69]. Table 8 shows the difference between one-to-one and partial matching mechanism and how it changes accuracy values.

TABLE 9. Ligature recognition results on SDA.

Ligatures Type	No of Total Ligatures	Exact-Sequence Recognition rate	Partial-Sequence Recognition Rate
Trained-Set	3801 (3.8K)	99.84% (22Ep)	99.99%
Test-Set	423 (0.423K)	0	**94.90%**
Trained-Set	46036 (46K)	96.35% (55Ep)	97.82%
Test-Set	5115 (5K)	46.92%	**95.20%**

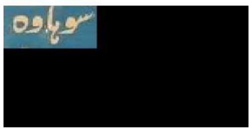
TABLE 10. Urdu text recognition results on real outdoor images.

Ligatures Type	No of Total Images	Exact-Sequence Recognition rate	Partial-Sequence Recognition Rate
Trained-Set	984	97.35%	99.35%
Test-Set	110	4.54%	**76.60%**
Tested using 3.8K synthetic model	110	0%	68.35%
Tested using 3.8K synthetic model	984	0%	71.95%
Tested using 46K Synthetic model	110	0%	69.55%
Tested using 46K Synthetic model	984	0%	73.25%

The exact/one-to-one match happens if each character(s) appears in the correct order and is the same [67], even if a single character mismatch occurs, it is considered a zero match.

For judging the capability of TSDNN, 2-types of experiments are conducted to determine the accuracy of the model. The number of training and testing images vary in each experiment. In the first experiment (SDA4) 3.8k images for training and 0.42k images for testing selected for training and testing respectively. While in 2nd experiment (SDA51) 46k images were used for TSDNN training and 5.1k images for testing. Few images from SDA51 were also eliminated from training or testing due to minor reasons like too much near or crossing the boundary of an image by annotated rectangular coordinates. For the SDA4 test-set, we obtained a partial sequence recognition rate of 94.90% while a similar measure for SDA51 achieved 95.20%. Finally, TSDNN generates editable Urdu text. Table 9 elucidate the results on both train and test-set.

TABLE 11. Prediction results on real outdoor test images.

Input Image	Predicted Text	Original Text
	خیر	خیر
	عا ?ک� نا	دعا کرنا
	� دانہ	دینہ
	لاوڈر	لاہور
	سوسہاوہ	سوہاوہ

D. RECOGNITION OF LIGATURES (NATURAL DATASET)

To demonstrate the robustness of TSDNN, it was also trained and tested on real outdoor images of more than 1000. This dataset contained real outdoor images, which were collected and annotated by us. Few of the images from this dataset have been shown earlier in Figure 2. It can be seen that real-world images contained Urdu text of varying length. Overall this dataset contained more than 12000 (12K) Urdu characters. 6-Types of experiments were carried out to determine the sequence classification capability of the proposed TSDNN model. In the first experiment, TSDNN was trained on real images and tested against real-world images. While in other experiments, the real world test-images are tested against 4k and 51k models, that were trained only on synthetic images. And it shows very fair results. Table 10 shows the outcome of all experiments.

Synthetic models show very good results even on real-world trainset images on which they were never trained on. 46K synthetic image model has a performance of 69.55% which is just nearly 7% shorth of real-world images trained model of 76.6%. It further highlights that as real-world images have a variety of font styles and noise, further expanding the synthetic data with noise and font style augmentation can narrow the performance gap between synthetic image model and real-world images based model. Few of TSDNN's recognition results in graphical form for real outdoor test images are shown in Table 11, where columns show original and predicted text. While a small green band shows the missing predicted character, in predicted text.

IV. DISCUSSION

we presented a novel methodology for Urdu text, covering the entire spectrum of both text detection and recognition as well as orientation prediction. Also, synthetic Urdu datasets with text embedded in outdoor images are generated. For determining the best CNN for Urdu text detection, the first set of 4.2K synthetic dataset images were taken as an input to CNNs. 4-different CNNs are used as feature extractors to train four FasterRCNN models. These models are then evaluated to compare their abilities for the task of Urdu text/script detection. FasterRCNN performed best for detection purposes, using Resnet50 based features with AP of.9812. Googlenet and Resnet18 have comparable performance. While Squeeznet features don't seem suitable well for Urdu text detection.

To Determine the rotation angle of the detected ligatures a custom residual regression neural network was trained on both sets of the synthetic datasets. Their accuracy in determining the angle of rotation for Urdu text was 79.33% for the first set and 99.06 % for the second set. These trained models can then further be utilized to correct the angle of text or used to further improve the recognition accuracy of text in the next stages of OCR.

TSDNN used googlnet, resnet50 features in combination with 2-layer BLSTM to recognize Urdu text. The model achieved Training accuracy of 99% on both sets, while the accuracy of 94.9% and 95.2% achieved for 4.2K and 51K datasets respectively. Also, the exact-match rate of 46.92 % on test-set shows that increasing the number of images improves exact-match accuracy. TSDNN results on real-world images are very encouraging because the synthetic models were not trained on a word or even lines containing multiple characters, but were able to fairly handle a variety of length of Urdu text in real outdoor images. Experimental results have demonstrated that the proposed Urdu text methodology is effective for both Urdu scene text detection and recognition scenarios, due to its high accuracy and completeness in Urdu text structure feature extraction using resnet50 and googlenet. It is also observed that more artificial training samples are essential in improving and validating the Urdu text recognition rate and Orientation prediction rate.

V. CONCLUSION AND FUTURE DIRECTIONS

In this paper, the proposed methodology, covering essential phases of photoOCR is presented and evaluated using various sub-architectures. These phases include the detection of text, Orientation determination of text, and finally recognizing the written text in outdoor images. Also 5-subsets of datasets generated with a maximum of 51K images, for evaluating the 3-main phases of PhotoOCR. The performance of datasets namely SDAi, SDA4, SDA51, SDB4, and SDB51 is assessed by presented work. By implementing the proposed methodology, the achieved performance is very encouraging. For detection problem, Resnet50 based FasterRCNN was found to be best for Urdu text detection, while for text rotation angle prediction our RRNN found to very successful.

Regarding Urdu text recognition proposed TSDNN neural network was a resounding success for both synthetic text and real-world out-door images. TSDNN finally generated editable Urdu ligatures in Unicode format.

In the future, large datasets with more transformations covering noise, rotation, color, background, font variations are vehemently required for judging the robustness of this and other techniques. They are necessary parameters for any successful Urdu script/text detector as well as a robust sequence classifier for Urdu text. Other CNN models like inception [70], Nasnet [71], Xception [72] features can be evaluated. Powerful detectors ultimately reduce the false outcomes of the photoOCR systems, which will ultimately improve the results of OCR in terms of recognition rate.

REFERENCES

[1] Q. Ye and D. Doermann, "Text detection and recognition in imagery: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 7, pp. 1480–1500, Jul. 2015.

[2] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan, "FOTS: Fast oriented text spotting with a unified network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5676–5685.

[3] A. Rana and G. S. Lehal, "Offline urdu OCR using ligature based segmentation for Nastaliq script," *Indian J. Sci. Technol.*, vol. 8, no. 35, pp. 1–9, 2015.

[4] A. Raza, I. Siddiqi, C. Djeddi, and A. Ennaji, "Multilingual artificial text detection using a cascade of transforms," in *Proc. 12th Int. Conf. Document Anal. Recognit.*, Aug. 2013, pp. 309–313.

[5] S. Unar, A. H. Jalbani, M. M. Jawaid, M. Shaikh, and A. A. Chandio, "Artificial urdu text detection and localization from individual video frames," *Mehran Univ. Res. J. Eng. Technol.*, vol. 37, no. 2, pp. 429–438, 2018.

[6] A. Mirza, M. Fayyaz, Z. Seher, and I. Siddiqi, "Urdu caption text detection using textural features," in *Proc. 2nd Medit. Conf. Pattern Recognit. Artif. Intell.*, 2018, pp. 70–75.

[7] *ICDAR2017 Competition on Multi-Lingual Scene Text Detection and Script Identification*. Accessed: Aug. 2, 2017. [Online]. Available: http://rrc.cvc.uab.es/?ch=8

[8] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny, "ICDAR 2015 competition on robust reading," in *Proc. 13th Int. Conf. Document Anal. Recognit. (ICDAR)*, Tunis, Tunisia, Aug. 2015, pp. 1156–1160.

[9] C. Yao. *MSRA Text Detection 500 Database (MSRA-TD500)*. Accessed: Aug. 2018 [Online]. Available: http://www.iapr-tc11.org/mediawiki/index.php/MSRA_Text_Detection_500_Database_(MSRA-TD500)

[10] X. Ren, Y. Zhou, Z. Huang, J. Sun, X. Yang, and K. Chen, "A novel text structure feature extractor for chinese scene text detection and recognition," *IEEE Access*, vol. 5, pp. 3193–3204, 2017.

[11] A. A. Chandio, M. Pickering, and K. Shafi, "Character classification and recognition for urdu texts in natural scene images," in *Proc. Int. Conf. Comput., Math. Eng. Technol. (iCoMET)*, Mar. 2018, pp. 1–6.

[12] S. Ahmed, S. Naz, M. Razzak, and R. Yusof, "Arabic cursive text recognition from natural scene images," *Appl. Sci.*, vol. 9, no. 2, p. 236, 2019.

[13] A. Raza and I. Siddiqi, "A database of artificial Urdu text in video images with semi-automatic text line labeling scheme," in *Proc. 4th Int. Conf. Adv. Multimedia (MMEDIA)*, Chamonix, France, 2012, pp. 75–81.

[14] A. Raza, A. Abidi, and I. Siddiqi, "Multilingual artificial text detection and extraction from still images," *Proc. SPIE*, vol. 8658, Feb. 2013, Art. no. 86580V.

[15] A. J. Jamil, A. Batool, Z. Malik, A. Mirza, and I. Siddiqi, "Multilingual artificial text extraction and script identification from video images," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 4, pp. 529–539, 2016.

[16] A. A. Chandio and M. Pickering, "Convolutional feature fusion for multi-language text detection in natural scene images," in *Proc. 2nd Int. Conf. Comput., Math. Eng. Technol. (iCoMET)*, Jan. 2019, pp. 1–6.

[17] U. Hayat, M. Aatif, O. Zeeshan, and I. Siddiqi, "Ligature recognition in urdu caption text using deep convolutional neural networks," in *Proc. 14th Int. Conf. Emerg. Technol. (ICET)*, Nov. 2018, pp. 1–6.

[18] I. U. Din, Z. Malik, I. Siddiqi, and S. Khalid, "Line and ligature segmentation in printed Urdu document images," *J. Appl. Environ. Biol. Sci.*, vol. 6, no. 3S, pp. 114–120, 2016.

[19] I. Ahmad, X. Wang, R. Li, M. Ahmed, and R. Ullah, "Line and ligature segmentation of urdu nastaleeq text," *IEEE Access*, vol. 5, pp. 10924–10940, 2017.

[20] A. Mahmood and A. Srivastava, "A novel segmentation technique for urdu type-written text," in *Proc. Recent Adv. Eng., Technol. Comput. Sci. (RAETCS)*, Feb. 2018, pp. 1–5.

[21] N. Sambyal and P. Abrol, "Automatic text extraction and character segmentation using maximally stable extremal regions," 2016, *arXiv:1608.03374*. [Online]. Available: http://arxiv.org/abs/1608.03374

[22] N. Gupta and A. S. Jalal, "A robust model for salient text detection in natural scene images using MSER feature detector and grabcut," *Multimedia Tools Appl.*, vol. 78, no. 8, pp. 10821–10835, Apr. 2019.

[23] C. Shi, C. Wang, B. Xiao, Y. Zhang, and S. Gao, "Scene text detection using graph model built upon maximally stable extremal regions," *Pattern Recognit. Lett.*, vol. 34, no. 2, pp. 107–116, Jan. 2013.

[24] N. Sabbour and F. Shafait, "A segmentation-free approach to Arabic and Urdu OCR," *Proc. SPIE*, vol. 8658, Feb. 2013, Art. no. 86580N.

[25] N. H. Khan, A. Adnan, and S. Basar, "Urdu ligature recognition using multi-level agglomerative hierarchical clustering," *Cluster Comput.*, vol. 21, no. 1, pp. 503–514, Mar. 2018.

[26] A. Mirza, I. Siddiqi, S. G. Mustufa, and M. Hussain, "Impact of pre-processing on recognition of cursive video text," in *Proc. Iberian Conf. Pattern Recognit. Image Anal.* Cham, Switzerland: Springer, 2019, pp. 565–576.

[27] S. B. Ahmed, I. A. Hameed, S. Naz, M. I. Razzak, and R. Yusof, "Evaluation of handwritten urdu text by integration of MNIST dataset learning experience," *IEEE Access*, vol. 7, pp. 153566–153578, 2019.

[28] A. Ali, M. Pickering, and K. Shafi, "Urdu natural scene character recognition using convolutional neural networks," in *Proc. IEEE 2nd Int. Workshop Arabic Derived Script Anal. Recognit. (ASAR)*, Mar. 2018, pp. 29–34.

[29] S. Bin Ahmed, S. Naz, M. Imran Razzak, and R. Yousaf, "Deep learning based isolated Arabic scene character recognition," 2017, *arXiv:1704.06821*. [Online]. Available: http://arxiv.org/abs/1704.06821

[30] L. Oulladji, K. Feraoun, M. Batouche, and A. Abraham, "Arabic text detection using ensemble machine learning," *Int. J. Hybrid Intell. Syst.*, vol. 14, no. 4, pp. 233–238, Aug. 2018.

[31] M. J. Rafeeq, Z. ur Rehman, A. Khan, I. A. Khan, and W. Jadoon, "Ligature categorization based Nastaliq urdu recognition using deep neural networks," *Comput. Math. Org. Theory*, vol. 25, no. 2, pp. 184–195, Jun. 2019.

[32] I. Uddin, I. Siddiqi, and S. Khalid, "A holistic approach for recognition of complete urdu ligatures using hidden Markov models," in *Proc. Int. Conf. Frontiers Inf. Technol. (FIT)*, Dec. 2017, pp. 155–160.

[33] S. Naz, A. I. Umar, R. Ahmad, I. Siddiqi, S. B. Ahmed, M. I. Razzak, and F. Shafait, "Urdu Nastaliq recognition using convolutional–recursive deep learning," *Neurocomputing*, vol. 243, pp. 80–87, Jun. 2017.

[34] N. Javed, S. Shabbir, I. Siddiqi, and K. Khurshid, "Classification of urdu ligatures using convolutional neural networks—A novel approach," in *Proc. Int. Conf. Frontiers Inf. Technol. (FIT)*, Dec. 2017, pp. 93–97.

[35] M. Jain, M. Mathew, and C. V. Jawahar, "Unconstrained OCR for urdu using deep CNN-RNN hybrid networks," in *Proc. 4th Asian Conf. Pattern Recognit. (ACPR)*, Nanjing, China, Nov. 2017.

[36] S. Shabbir and I. Siddiqi, "Optical character recognition system for urdu words in Nastaliq font," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 5, pp. 567–576, 2016.

[37] A. Jamil, I. Siddiqi, F. Arif, and A. Raza, "Edge-based features for localization of artificial urdu text in video images," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2011, pp. 1120–1124.

[38] M. F. Naeem, N. U. S. Zia, A. A. Awan, F. Shafait, and A. U. Hasan, "Impact of ligature coverage on training practical urdu OCR systems," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 1, Nov. 2017, pp. 131–136.

[39] Center for Language Engineering KICS-UET. *Valid Ligatures of Urdu (CLE)*. Accessed: Aug. 10, 2018. [Online]. Available:http://www.cle.org.pk/software/ling_resources/UrduLigatures.htm

[40] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young, "ICDAR 2003 robust reading competitions," in *Proc. 7th Int. Conf. Document Anal. Recognit.*, 2003, pp. 682–687.

[41] L. Yuliang, J. Lianwen, Z. Shuaitao, and Z. Sheng, "Detecting curve text in the wild: New dataset and new solution," 2017, *arXiv:1712.02170*. [Online]. Available: http://arxiv.org/abs/1712.02170

[42] C. K. Ch'ng and C. S. Chan, "Total-text: A comprehensive dataset for scene text detection and recognition," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Nov. 2017, pp. 935–942.

[43] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu, "Detecting texts of arbitrary orientations in natural images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1083–1090.

[44] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie, "COCO-text: Dataset and benchmark for text detection and recognition in natural images," 2016, *arXiv:1601.07140*. [Online]. Available: http://arxiv.org/abs/1601.07140

[45] K. Maheshwari, A. N. J. Raj, V. G. V. Mahesh, Z. Zhuang, E. Rufus, P. Shivakumara, and G. R. Naik, "Bilingual text detection in natural scene images using invariant moments," *J. Intell. Fuzzy Syst.*, vol. 37, no. 5, pp. 6773–6784, Nov. 2019.

[46] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "EAST: An efficient and accurate scene text detector," in *Proc. CVPR*, Jul. 2017, pp. 2642–2651.

[47] S. B. Ahmed, S. Naz, M. I. Razzak, and R. B. Yusof, "A novel dataset for English-Arabic scene text recognition (EASTR)-42K and its evaluation using invariant feature extraction on detected extremal regions," *IEEE Access*, vol. 7, pp. 19801–19820, 2019.

[48] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[49] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*. [Online]. Available: http://arxiv.org/abs/1602.07360

[50] *Deep Learning Method (7): The Latest SqueezeNet Model is Explained*. Accessed: Aug. 28, 2019. [Online]. Available: http://www.ishenping.com/ArtInfo/2891589.html

[51] S. Das. *CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and More*. Accessed: Jan. 30, 2019. [Online]. Available: https://medium.com/sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

[52] *Resnet18*. Accessed: Apr. 14, 2019. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/resnet18.html

[53] *Resnet50*. Accessed: Apr. 14, 2019. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/resnet50.html

[54] *How to Split Resnet50 Model From top as Well as From Bottom?*. Accessed: Apr. 15, 2019. [Online]. Available: https://stackoverflow.com/questions/54207410/how-to-split-resnet50-model-from-top-as-well-as-from-bottom

[55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[56] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva, "Places: An image database for deep scene understanding," 2016, *arXiv:1610.02055*. [Online]. Available: http://arxiv.org/abs/1610.02055

[57] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[58] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[59] R. Girshick, "Fast R-CNN," 2015, *arXiv:1504.08083*. [Online]. Available: http://arxiv.org/abs/1504.08083

[60] Matlab. (2019). *R-CNN, Fast R-CNN, and Faster R-CNN Basics*. Accessed: Aug. 1, 2019. [Online]. Available: https://www.mathworks.com/help/vision/ug/faster-r-cnn-basics.html

[61] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 391–405.
[62] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
[63] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, Cham, Switzerland: Springer, 2016, pp. 21–37.
[64] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.
[65] J. Hui. (2018). *mAP (Mean Average Precision) for Object Detection*. Accessed: Apr. 20, 2019. [Online]. Available: https://medium.com/jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173
[66] MatlabR2019b. *Train Residual Network for Image Classification*. Accessed: Nov. 2019. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/train-residual-network-for-image-classification.html
[67] S. Y. Arafat and M. J. Iqbal, "Two stream deep neural network for sequence-based urdu ligature recognition," *IEEE Access*, vol. 7, pp. 159090–159099, 2019.
[68] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys. Doklady*, vol. 10, no. 8, pp. 707–710, 1966.
[69] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-26, no. 1, pp. 43–49, Feb. 1978.
[70] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[71] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.
[72] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1800–1807, doi: 10.1109/CVPR.2017.195.

SYED YASSER ARAFAT received the M.S.C.S. degree from International Islamic University (IIU), Islamabad, in 2007. He is currently pursuing the Ph.D. degree with the Department of Computer Science, UET Taxila, with research on outdoor urdu-text detection and recognition. He is currently working as an Assistant Professor with the Department of Computer Science and Information Technology (CS and IT), Mirpur University of Science and Technology (MUST). He has more than 15 years of teaching experience at various National Universities. His research interests include NLP, computer-vision, deep learning, and robotics.

MUHAMMAD JAVED IQBAL received the M.Sc. degree in computer science from the University of Agriculture, Faisalabad, Pakistan, in 2001, the M.S./M.Phil. degree in computer science from International Islamic University Islamabad, Pakistan, in 2008, and the Ph.D. degree in computer science/information technology from Universiti Teknologi PETRONAS, Malaysia, in February 2015. He is currently a HEC approved Ph.D. Supervisor and working as an Assistant Professor with the Computer Science Department, University of Engineering and Technology Taxila, Pakistan. After completion of his Ph.D. studies, he has been actively involved in research. He has more than 20 international publications which include four ISI indexed impact factor journals, one book chapter Springer LNEE, and ten Scopus indexed conferences. His research interests include machine learning, data science, pattern recognition, and computational intelligence algorithms for biological data classification, bioinformatics, and big data mining. He is also a Guest Editor of the *Journal of Internet Technology* (Indexed by SCI-E) and a Reviewer of renowned national and international journals and conferences.