IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 15, 2021, accepted September 3, 2021, date of publication September 7, 2021,
date of current version September 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3110787

MMU-OCR-21: Towards End-to-End Urdu Text Recognition Using Deep Learning

TAYYAB NASIR[1], **MUHAMMAD KAMRAN MALIK**[2], **AND KHURRAM SHAHZAD**[3]
[1]Punjab University College of Information Technology (PUCIT), University of the Punjab, Lahore 54590, Pakistan
[2]Department of Information Technology, Faculty of Computing and Information Technology, University of the Punjab, Lahore 54590, Pakistan
[3]Department of Data Science, Faculty of Computing and Information Technology, University of the Punjab, Lahore 54590, Pakistan

Corresponding author: Tayyab Nasir (tayyabnasir22@gmail.com)

ABSTRACT Optical Character Recognition (OCR) is a technique that generates text from an image. Recognizing the importance of OCR in real-world settings, a plethora of techniques have been developed for Western, as well as Asian languages. Urdu is a prominent South Asian language and a number of different solutions for Urdu OCR have been proposed. However, fewer attempts have been made to develop end-to-end deep learning-based solutions for recognizing printed Urdu text. Furthermore, several benchmark corpora for Urdu OCR have been developed that can be used for training and evaluation of different OCR techniques. However, there are a number of limitations of the existing Urdu corpora: firstly, most of them have either character or word or text images, which are usually rendered using only a single font, Nastaleeq. Secondly, the volume of the existing datasets is so small that it is not suitable for working with the deep-learning techniques that have achieved groundbreaking results for OCRs. To that end, in this study, we have proposed a very large Multi-level and Multi-script Urdu corpus (MMU-OCR-21). It is the largest-ever Urdu corpus of printed text that is effectively suitable to work with deep learning techniques. In total, the corpus is composed of over 602,472 images, including text-line and word images in three prominent fonts, and their respective ground truth. Also, we have performed experiments using multiple state-of-the-art deep learning techniques for text-line and word level images.

INDEX TERMS Artificial neural networks, corpus generation, image processing, optical character recognition, text recognition, Urdu OCR.

I. INTRODUCTION

OCR is the process of converting handwritten or printed text images into machine readable format [1], [2]. It is widely acknowledged that there are numerous applications of OCR systems. For instance, given a book or newspaper image, an OCR system can automatically read and convert the text in an image to a sequence of ASCII or Unicode characters, which can subsequently be used for various purposes, such as searching, highlighting, annotating, and translating [3]. Furthermore, the application domains of OCR include government, as well as private organizations. For instance, it can be used by immigration department for passport recognition, City Traffic Police for number plate recognition [4], [5], and banking organizations for automatic processing of demand drafts and cheques [6]. In addition to the above, OCR can also be used to preserve and digitize ancient literature as a heritage. Another useful application of OCR is assistive technology for blinds and visually impaired users. Due to numerous application areas, OCR is widely acknowledged as an established research problem in the field, where a plethora of rules-based, as well as supervised learning techniques, have been developed [3], [7].

OCRs can be classified in a number of different ways. For instance, Online and Offline OCR [8]. Where, Online OCRs can recognize texts on-the-fly by merely analyzing the patterns of the strokes generated by a stylus or drawn on a touch screen panel [1], [3], [9]. On the contrary, Offline OCRs work on an existing text from an image [1], [3], [7], [9]. It is widely acknowledged that Offline OCRs are harder to develop [2], [10].

OCRs can also be classified as Segmentation-based or Segmentation-free [11], where the former type requires segmented text at character or ligature level, for learning and prediction. Where, ligature refers to a single connected body which could be a single character or a combination of two

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

or more characters [11], [12]. Typically, segmentation-based systems require an image of character which is cropped using some segmentation technique [13]–[15]. On the contrary, a Segmentation-free OCR has the ability to recognize complete words or text lines, without requiring any prior segmentation. Typically, an OCR of this class takes a text image as input and subsequently, recognizes text in the image. Segmentation-based OCRs differ from Segmentation-free OCRs as the former requires less amount of data, whereas, the latter requires a large amount of training data for better generalization. However, Segmentation-based OCRs require labeling of individual characters in images which is a more challenging task.

The recent advancements in deep learning have been used to provide a single end-to-end solution for many of the existing problems, such as speech recognition [16]–[18], machine translation [19], text summarization [20], [21], and image captioning [22], [23]. Furthermore, deep learning has also been used for developing end-to-end solutions for OCR [7], [11], [24]–[27]. An end-to-end solution for OCR allows a single model to enclose all the substituent tasks of an OCR into a single model. This allows optimizing the model as a single cohesive unit. This concept of end-to-end learning is made possible by deep learning.

A plethora of studies had relied on deep learning techniques for building up OCRs for European languages, such as English, German, and French, following the Latin script, with remarkably high accuracies [11], [24], [28]. However, Asian languages significantly differ from the European languages. For instance, English, German, and French are written from left to right, whereas, Arabic, Persian, and Urdu, which follow the Arabic script, are written from right to left. Therefore, the OCR techniques developed for European languages cannot be used for Asian languages. Several attempts have been made to develop OCRs for Asian languages, such as Arabic, Persian, Bengali, Urdu, Punjabi, and Hindi [10], [25], [29].

Urdu is a prominent South Asian language, having well over 100 million speakers worldwide, making it the 20th most spoken language of the world according to the Ethnologue. Urdu has a total of 39 characters and several of these characters occur in multiple forms including isolated, prefix, postfix, and middle form [30]. Urdu has several writing styles, including Devani, Kofi, Naskh, Nastaleeq, Riqa, and Taluth. Among these, the two prominent styles are Naskh and Nastaleeq [31], [32]. Predominantly the printed material available today is in Nastaleeq font, whereas Naskh is the more dominant font for the digital material [30], [31]. For instance, Akhbare-Jehan is a widely distributed printed magazine that uses Nastaleeq. In contrast, BBC Urdu website uses Naskh font. Figure 1, illustrates the differences between the two scripts with the help of an example word. It can be observed from the figure that the initial characters in the Naskh script are aligned along a straight-baseline, whereas in the Nastaleeq script the characters have a diagonal baseline which changes from right to left. Also, there are significant differences when it comes to ligature formation and how

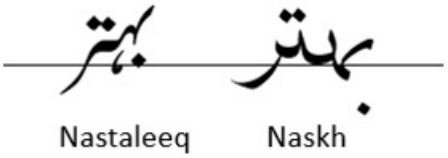

FIGURE 1. Example of an Urdu word printed using two different fonts.

the shape of characters varies when using the two different fonts. Similarly, the presence of the 'hook' diacritic under the letter ہ (hey) in case of Nastaleeq can be observed in the same figure, whereas it is absent in case of Naskh.

In the presence of such a variation, the performance of a deep learning technique trained on Nastaleeq script is not effective for detecting Naskh font text. It is therefore desirable to develop an Urdu OCR that is capable of identifying text written in any font for a wider applicability. However, to the best of our knowledge, an adequately large and comprehensive dataset containing the required variation is not available for Urdu OCR. Currently, the available datasets are either very small or do not contain adequate variations that are necessary for building end-to-end OCR systems using deep learning. Furthermore, almost all the existing Urdu OCRs are developed for Nastaleeq font, which are not applicable when the source images contain text in other fonts.

To that end, in this study, the following key contributions are made:

- Firstly, we have employed a systematic and rigorous procedure to develop a holistic Urdu OCR corpus that is large enough for training an end-to-end deep neural network model. The corpus has 602,472 image files which include 301,623 text-line images, 300,000 word images, and 849 character images. Altogether, the corpus contains 619,555 words and 2,106,267 characters. The key features of our corpus are that it is a multi-level and multi-script, and it encapsulates three levels, text-line, word and character level, as well as multiple fonts to render text images at all the three levels. The fonts used for rendering the images are: Naskh, Nastaleeq and Tehreer.
- Secondly, five different deep neural networks based end-to-end segmentation free models are used to evaluate the proposed corpus.

The rest of the paper is organized as follows: Section II presents the related work. Section III presents the details of the corpus generation process. The specifications of our developed corpus and its comparison with other corpora are presented in Section IV. The deep learning techniques that we have used for OCR are discussed in Section V. The details of the experimental settings and the analysis of the results are presented in Section VI. Finally, the conclusions are provided in Section VII.

II. RELATED WORK

A plethora of studies has been conducted to develop OCR techniques for Western, as well as Asian languages. Furthermore, several benchmark corpora have been made available for these languages. On the contrary, fewer attempts have been made for Urdu language OCRs, and there is scarcity of benchmark datasets for the Urdu language. In particular, the fundamental requirements of deep learning techniques are to have a generalized and substantially large corpus that can be used for effective learning and prediction. However, despite the presence of several Urdu OCR corpora for printed [33]–[36], handwritten [37]–[42] and natural scene text recognition [43], it is found that the existing datasets are small in size and they have inadequate diversity for training deep neural network models. The details of these datasets along with a comparison with our proposed corpus is presented in Section IV. For a better comprehension, traditional machine learning and deep learning techniques are discussed, separately.

A. MACHINE LEARNING FOR OCR

Machine learning and computer vision work hand-in-hand for image processing. Several studies have used machine learning techniques for OCR. Segmentation-based OCR techniques consist of a number of steps, where each step performs a certain task to generate an output which is fed to the next step. Typically, these steps are pre-processing, segmentation, feature extraction and training a machine learning based classifier. Where, pre-processing often employs techniques, such as smoothing, binarization, etc., to the input image for enhancing features of an image and reducing noise in the image. The pre-processed image is then segmented at either line, word or character level, using techniques, such as Histogram projection or Hough transformation. Subsequently, features are extracted from the segmented characters. The commonly used features for OCR are Gradient features, chain codes, etc. Finally, these extracted features are used to train machine learning classifiers, such as Support Vector Machine and Artificial Neural Network [27], [44]–[46].

A number of techniques have been employed for building segmentation-based OCRs for the Urdu language. These techniques recognize individual Urdu words and characters [47]–[49]. However, there are some techniques which first segments Urdu words into characters and then employees classification techniques to identify characters. These studies focus on recognizing Urdu characters in only Nastaleeq font [50].

In addition to segmenting and recognizing characters, studies have been conducted on segmenting and recognizing ligatures in Urdu, where a ligature is a single connected body made up of one or more of characters. There are several notable studies about segmentation-based OCRs recognizing ligatures in Urdu [33], [34], [51]–[53]. Typically, these studies employ diverse feature extraction, such as Gabor filters, DCT features, and Zoning based features [54]. Furthermore, these studies recognize Urdu ligatures in Nastaleeq font using SVM, KNN and HMM. However, a few studies have also used SIFT and SURF features for Urdu ligatures identification [55].

B. DEEP LEARNING FOR OCR

The recent advancements in deep learning have achieved groundbreaking results in numerous domains [56]. It includes building end-to-end OCR systems for English, as well as for other languages. The key reason for these groundbreaking results stem from the multi-layer trait of deep neural networks [57]. One such layered architecture consists of CNN layers which are followed by RNN layers. Furthermore, the RNN layers are followed by Connectionist temporal classification (CTC) loss function [27]. A model with a CER of 11%, working only with data related to bank cheques is proposed [26], which used the same process, using Gated Recurrent CNN for feature extraction and Bidirectional Long short-term memory (BLSTM) for mapping these features at multiple timestamps generating a probability distribution of every character at each timestamp.

Text in natural scene images is identified using an end-to-end deep neural network-based model [58]. The study used YOLOv2 with CTC loss and evaluated its performance on ICDAR 2013 and 2015 datasets. Individual components for segmentation-based OCR systems are also developed using neural networks, where each of the subsystem used a DNN of its own and it is trained separately to accomplish a particular task [59].

Deep learning techniques have also been applied to low resource languages, such as Telugu multi-font OCR [60], and multiple Bengali fonts [61]. Arabic is a notable language whose characteristics are similar to Urdu [62]. For instance, both in Arabic and Urdu the flow of sentences is from right to left. Furthermore, many words in Urdu are also derived or borrowed from Arabic and Urdu follows the Arabic orthography. Due to these similarities, many techniques that are applied to solve certain problems in Arabic can also be adapted for Urdu. A prominent Arabic OCR was proposed in [63] that was based on five steps: pre-processing, segmentation (sub-word and letter segmentation using the Zidouri algorithm [64]), thinning stage using Hilditch thinning algorithm [65], and features extraction and character classification using Decision trees. Another such Arabic OCR model was trained on DARPA corpus using stacked BLSTM which is connected with CTC loss function for predicting Arabic text sequences. Also, a language model was added to the technique at the time of prediction to enhance the output of the actual trained OCR [66].

A number of deep learning techniques have been applied to Urdu datasets, either presenting an end-to-end Urdu OCR or simply using DNNs as character or word classifiers, or even as feature extractors. For instance, Stacked Denoising AutoEncoders that were originally used for recognizing Bangla [67] were also used for recognizing Urdu ligatures [68]. The CNN + RNN end-to-end model was applied

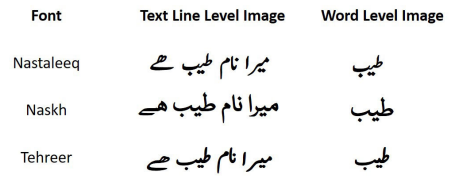

FIGURE 2. Example of text images rendered in 3 fonts.

to Urdu text images [69] to predict a sequence of characters making up the input Urdu text line. Models comprising of Long short-term memory (LSTM) layers connected to a CTC loss function were also proposed. These models work at character and ligature level, respectively [30], [70]. Another OCR system using LSTM + CTC was proposed that used the same UPTI dataset for training [71].

The key deficiencies in the existing studies are that the datasets used for training Urdu OCR system are very small and that the datasets used for training are merely available for a single font, mostly Nastaleeq. Whereas, a Multi-level and Multi-script Urdu OCR dataset is obligatory for Urdu OCR. To that end, the focus of this work, is to overcome the size and diversity limitations of the existing corpora and develop a benchmark corpus for printed Urdu.

III. CORPUS GENERATION

This section presents the architectural details of our novel Multi-level and Multi-script Urdu OCR (MMU-OCR-21) corpus. The novelty of MMU-OCR-21 lies in the following: (a) holistic nature of the corpus that spans across multiple levels, including character, word, text line level, (b) support for three fonts, and (c) providing adequately large number of example points that will be useful to better generalize deep neural network based models. Thus, trying to overcome the limitations of the smaller single font based printed Urdu corpora that already exist. Figure 2 depicts an example of text line and word level image containing text in all three fonts that are used in our MMU-OCR-21 corpus.

Our synthesis of literature revealed that three types of approaches have been used for generating an Urdu OCR corpora. These are: real-world, controlled, and synthetic. In the real-world approach, the existing documents are collected, their images are generated, and subsequently, the ground truth (or human benchmark) is produced [36], [43]. The existing documents may include handwritten text, such as student assignments, or printed material, such as books and newspapers. A key strength of this approach is that the generated corpus mirrors real-world cases, whereas the key issues with the use of this approach are as follows: firstly, generating the human benchmark having images and their corresponding Urdu text in the digital form, is a resource-intensive task. Secondly, there is scarcity of documents belonging to a diverse range of content. Finally, due to the large vocabulary size of Urdu language and its inflectional nature, it is particularly challenging to generate a comprehensive benchmark for Urdu.

In the controlled approach, subjects are asked to reproduce a given text in their handwriting which is available in digital form. Subsequently, the written documents are scanned to generate images corpus, whereas for these images, the source text in digital form is used as a benchmark. The key benefit of this approach is that the human benchmark can be generated with little effort. Furthermore, with this approach, it is possible to control the diversity of written text by choosing appropriate subjects. For instance, subjects having different writing styles, genders, age groups, and levels of writing proficiency, can be chosen. The key limitation of this approach is that generating a substantially large dataset, that can be used for state-of-the-art deep learning techniques, is very challenging as arranging enough subjects, as well as generating sufficient samples from each resource, is a time intensive task.

In the synthetic approach, the raw digital text is given as input to an automated technique for generating printed text images. A key strength of this approach is that a large-sized corpus and its corresponding ground truth can be rapidly generated. However, the limitation of this approach is that the generated images may not have the necessary variation that is typically available in real-world settings, yet such variations can be generated computationally using data augmentation techniques including rotating, contrast and brightness changes, additive noise, and other degradations. In this study, a synthetic approach is used for generating our MMU-OCR-21 corpus due to its ability to rapidly generate a large images corpus and its corresponding ground truth. In order to handle the limitation of missing necessary variations, raw Urdu text is generated in three different fonts, Naskh, Nastaleeq, and Tehreer, and images are generated for each font. Hence, synthetically inducing the three variations of each word in the corpus. Therefore, one can say that the large-sized corpus, which is manifolds larger than the existing datasets, together with the three font variations can be a closer substitute of the real-world settings. Also, it is of key importance to mention that the provided images have no degraded versions of themselves. Thus, applying augmentation to the said dataset can be used to further increase the size and diversity of the corpus as per requirement.

Figure 3 provides an overview of the process that was used for developing the MMU-OCR-21 corpus. Recall, from the preceding section, the benchmark is defined at three levels, text line level, word level and character level, and also in three different fonts. The overall process is composed of three phases: data cleansing, three-level tokenization, and generating fixed-size text images, for three different Urdu fonts. The details are as follows:

A. DATA CLEANSING

As shown in Figure 3, the input to the corpus generation process is raw Urdu text. To generate Urdu text, in-line with a prominent study [40], raw Urdu text was collected. For collecting the raw Urdu text, news articles were scrapped from BBC Urdu bbc.com/urdu, which is an Urdu language station of a prominent news service, BBC News. The reasons

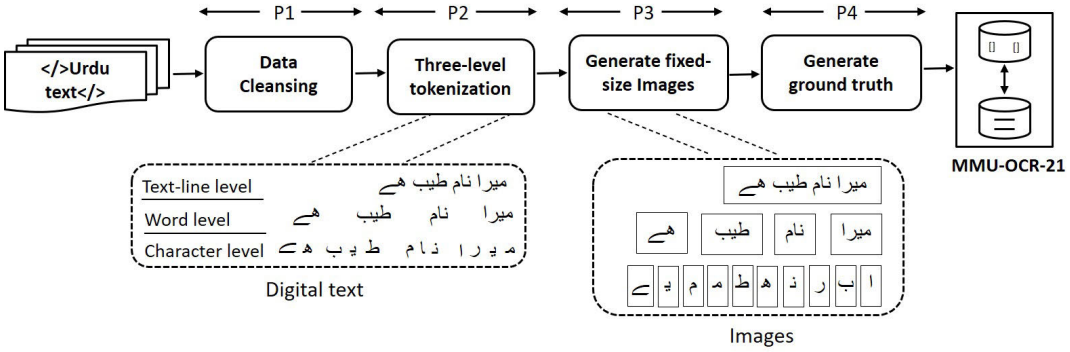

FIGURE 3. Corpus generation process.

for using news text are as follows: firstly, the news text is freely and publicly available in digital form with limited proprietary issues. Secondly, news articles cover a wide range of topics, therefore, a large collection of articles from diverse domains includes domain-specific vocabulary from several domains. Lastly, news text has been widely used in generating benchmark corpora for several languages, including Urdu, Arabic, and Hindi OCRs. Nevertheless, news articles have been used for generating information retrieval benchmarks, and benchmark corpora for plagiarism and text reuse detection. In particular, we scrapped 285,131 news articles from various genres, including history, science, defense, politics, and sports. The collected raw text is composed of 70,350,473 tokens. Where, the smallest news article is composed of seven words, the longest news is composed of 32,227 words, with an average length of 246.8 tokens.

The scrapped news articles contained non-Urdu content which had to be cleaned before any further processing. To that end, a two-step data cleansing process was employed. In the first step, regular expression-based string matching was used to remove the html tags using a built-in Python library RegEx. Furthermore, characters were parsed to filter-out the Unicodes that form emoticons. In the second step, a Unicode index was built which in fact contained every Unicode in the raw data along with the number of times it appeared. This index was then used to identify several garbage values, such as '¶', '¿', '±', '3/4', and 'Ž', by merely considering the Unicodes that had a lower occurrence frequency. These garbage values were omitted to generate the Urdu text corpus of 70,053,292 tokens, that was subsequently used for generating the image corpus.

B. THREE LEVEL TOKENIZATION

The purpose of the second phase of corpus generation procedure is to generate excerpts of Urdu text that could be used for generating the images corpus. As discussed earlier, we intend to develop a three-level images corpus, text line level, word level and character level, therefore, we have performed three levels of tokenization. The details of each step are as follows:

The first level tokenization requires identification of text-lines excerpts of Urdu text. Text line is a reasonable sequence of words that represents a useful expression or piece of information which may or may not be a complete sentence. In this step, we used Urdu language punctuation as delimiters for the 285,131 documents to generate 5,510,951 excerpts. Subsequently, a series of post-processing steps were performed which included, removing the empty text-lines, the text-lines that are too small (having less than three words), the ones that contained only numeric values, and the ones having less than eight non-white-space characters. Accordingly, 3,580,373 candidate excerpts were identified for generating text-line images in the subsequent phase.

For the word level, unique Urdu tokens were generated, also referred to as vocabulary, to avoid generating duplicate images in the subsequent phase. For that, tokenizing the corpus at word level was done using white spaces and other punctuation characters as delimiters for separating words. The cleansed documents contained a total of 70,053,292 word, forming a vocabulary of 312,813 words. Furthermore, the minimum, maximum, and average size of the vocabulary was computed and frequency of each vocabulary token was also computed to develop an understanding of the cleaned Urdu text corpus. It was observed that the smallest word in the vocabulary had two characters, the longest word had twelve characters, whereas the average length was 5.4 characters. Based on the frequency distribution, it was observed that the most common 100,000 vocabulary words constituted 60,922,354 tokens (87%) of the cleaned Urdu text corpus. The same 100,000 commonly used vocabulary words were chosen as a candidate for generating word level images corpus in the subsequent phase.

For the character level, apart from the characters that existed in our corpus other characters that belong to Urdu were identified and manually added to the corpus by native Urdu speakers. It was ensured that all the forms (isolated, initial, middle, end) of every Urdu character were made part of the corpus. Urdu digits and punctuation characters were also made part of the character level corpus. Accordingly, a set of 283 characters was identified as a candidate to generate character level images corpus in the subsequent phase.

C. GENERATE FIXED-SIZE IMAGES

The aim of this phase is to generate images for the MMU-OCR-21 that will in turn be used for the evaluation of printed

TABLE 1. Meta-information about OCR images.

Type	Name	Description
Meta	ID	Unique Id for every image
	Add Date	Audit field for the example point
	Update Date	Audit field for the example point
Rending	Font Size	Size of font in pts, used to render the image
	Font Web Name	Name of the font used for rendering
	Width	Image width in pixels
	Height	Image height in pixels
	Image Path	Relative path to the image of the example point
	Foreground Color	The color of the printed text
	Background Color	The color of background used in the text image
	Text Alignment	The alignment of text in the rendered image
	Calculated Width	The actual width of text in image
	Calculated Height	The actual height of text in image
Content	Text Label	The ground truth
	Word Count	Count of Words in the Text Line
	Character Count	Count of characters in the example point

Urdu OCR techniques. In particular, we intend to generate a three-level images corpus, text-line, word, and character level, for the three widely used Urdu fonts, Naskh, and Nastaleeq, Tehreer.

For generating the text-line images corpus, a further analysis of the 3,580,373 candidate text-lines excerpts was performed, which included computing minimum, maximum, and the average length of each text-line. Furthermore, the standard deviation and distribution of lengths were computed. Based on the results, it was observed that the smallest text-line had merely three words, whereas the longest one had 29 words. In the presence of such a significant variation, it is of paramount importance to carefully choose text-lines, as well as specific image dimensions, that can accommodate excerpts in three different fonts. The reason behind generating fixed-sized images and not variable size images was the intend to create a ready to use OCR corpus that does not require any pre-processing and can be fed directly to deep neural network models that usually require the same input dimensions. 128×32 image dimensions were chosen for text line and word images while 32×32 image dimensions were chosen for character level. It was estimated that a substantial number of text-lines, 100,541 excerpts of each of the three different fonts can be accommodated in these image dimensions, considering the fact that the same text rendered using same font size but different fonts will generate different size images for each font as can be observed in Figure 2.

Finally, 100,541 text-line, 100,000 word, and 283 character excerpts were used to automatically render images in three different fonts, Naskh (Nafees Naskh), Nasaleeq (Jameel Noori Nastaleeq) and Tehreer (Urdu Tehreer) using our text to image rendering software. In addition to the text excerpts, font size, image dimensions, background and foreground etc., was given as input. That is, for text-line the font was set to 11 pt, whereas for word and character level the font size was set to 12 pt. For each image, the background color was set to white, foreground color was set to black, and the text excerpts were horizontally and vertically centered. Accordingly, a large corpus of 602,472 images was generated which contains 301,623 text-lines images, 300,000 word images, and 849 character images.

D. GENERATING GROUND TRUTH

In the final phase, the ground truth for each of the 602,472 images was generated, which mere is intended to provide a format that is both human readable and machine readable, and can help facilitate other researchers using the MMU-OCR-21. The ground truth is mainly composed of meta-information about the images and the associated labels. The detailed constituents of the meta-information: name, type of meta-information, and description of each element of the meta-information, is presented in Table 1.

For an in-depth understanding of the corpus structure, Figure 4 illustrates the meta-information for an example image. It shows that the total number of images of the three levels, text-line, word, and character level, as well as example images of each type. Furthermore, the figure contains the three types of meta-information about the example text-line image. The three types of meta-information are separated by horizontal lines.

IV. CORPUS SPECIFICATIONS AND COMPARISON

This section presents the specifications of our developed MMU-OCR-21 corpus followed by a comparison of our proposed corpus with the existing Urdu OCR corpora. The specifications of MMU-OCR-21 corpus are presented in Table 2. The table is composed of two parts, the upper part of the table contains specifications of the images included in the corpus, whereas the lower part shows the statistics of the ground truth text. The first column contains the specification items, whereas the subsequent three columns contain the corresponding values of the three levels. Note, some values are not applicable to a certain level, for instance, text-lines count cannot be found at word and character level, therefore these values are marked with '-' sign in the table. In total, the corpus is composed of 602,481 files, with 602,472 images files in jpg format and 9 ground truth files in csv format having a space requirement of 1.22 GBs, and it will be freely and publicly available.[1]

From the specifications of the images presented in the table, it can be observed that the corpus contains images

[1] https://www.kaggle.com/tayyabnasir22/mmuocr21

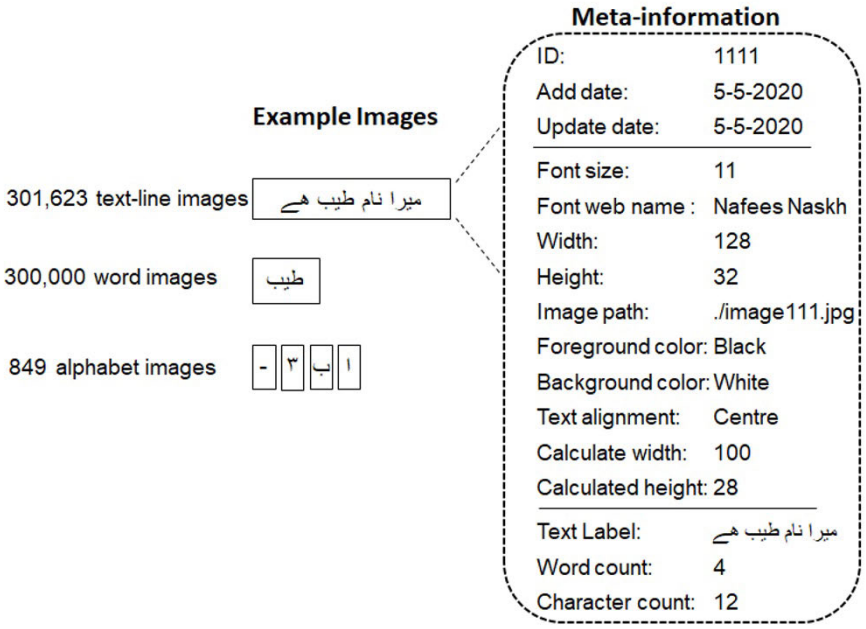

FIGURE 4. Illustration of meta information.

TABLE 2. Specifications of the MMU-OCR-21 corpus.

Items	Text-line Level	Word Level	Character Level
Naskh images	100,541	100,000	283
Nastaleeq images	100,541	100,000	283
Tehreer images	100,541	100,000	283
Font size	11	12	12
Total image count	301,623	300,000	849
Number of fonts	3	3	3
Dimensions	128*32	128*32	32*32
Word count	519,555	100,000	-
Character count	1,573,492	532,492	283
Count of text-lines	100,541	-	-
Unique characters	142	49	283
Unique words	31,347	100,000	-

in three different fonts, i.e. there are 200,824 (100,541 + 100,000 + 283) images for each font. From the specifications of the digital text presented in the table, it can be observed that the ground truth text excerpts contain 619,555 (519,555 + 100,000) words, and 2,106,267 (1,573,492 + 532,492 + 283) characters. Furthermore, text-line excerpts contain 31,347 unique words, whereas the word level contains 100,000 unique words.

The comparison of our proposed corpus with eight prominent Urdu OCR corpora is presented in Table 3. It includes, CENPARMI-2009 [38], UPTI-2013 [33], Prakash-2014 [40], CALAM-2016 [41], and CLE-2016 [36], UCOM-2017 [37], Ali-2018 [43], UNHD-2019 [39]. These corpora were identified through an extensive literature survey which was performed by searching through multiple digital libraries and google scholar, using several Urdu OCR related keywords. Subsequently, eight prominent corpora were chosen. Note, there are some other attempts to develop a corpus, such as [34], [35], [42], [72], however, either their corpora are too small or the available information is inadequate for the comparison.

It can be observed from the table that CLE [36] is the only multi-level corpus besides our developed corpus. However, our corpus is manifolds larger that the CLE corpus. Not only that, our MMU-OCR-21 is manifolds larger than all the eight Urdu OCR corpora used for the comparison. Given that the deep learning-based techniques are particularly sensitive to the size of the training corpus, requiring 10^5 to 10^6 training example points, thus, our developed corpus is a useful addition for deep learning techniques-based OCR systems. It can also be observed from the table that the existing corpora that are focused on printed Urdu, use only Nastaleeq font. In contrast, our corpus is the first of its kind that is available in three fonts, Naskh, Nastaleeq, and Tehreer.

V. DEEP LEARNING TECHNIQUES FOR OCR

For the evaluation of our proposed MMU-OCR-21 dataset a number of end-to-end deep neural network-based segmentation free OCR techniques are used. The goal is to evaluate the usefulness of our proposed corpus using several techniques. These techniques are built to OCR word and text line images. A description of each of these techniques is as follows:

The first two techniques that we use are based on the idea of combining CNN, BLSTM and CTC for recognizing text from images, as advocated by [27]. That is, using CNN for image feature extraction, passing the extracted features to stacked BLSTM layers to generate a sequence of probabilities at each timestamp, and finally, using CTC loss for training the proposed technique.

Deep Convolutional Neural Network (DCNN) has been widely used for many image classification problems. It consists of Convolutional and Max pooling layers. Where, DCNN part of the said model is used as a means for extracting

TABLE 3. A comparison of Urdu corpora.

Dataset	Size of text excerpts			Font	Source	Type
	Text-line	Word	Character			
CENPARMI [38]	NA	19,432	89,838	Unspecified	Handwritten	Controlled
UPTI [33]	10,000	-	-	Nastaleeq	Printed	Synthetic
Prakash [40]	NA	350,000	-	Unspecified	Handwritten	Controlled
CALAM [41]	3,403 + 2,353	46,664	-	Unspecified	Handwritten and Printed	Controlled
CLE [36]	13,712	44,478	726,385	Nastaleeq	Printed	Real World
UCOM [37]	6,400	62,000	-	Nastaleeq	Handwritten	Controlled
Ali [43]	-	-	28,000	Unspecified	Natural Scene	Real World
UNHD [39]	10,000	312,000	187,000	Nastaleeq	Handwritten	Controlled
MMU-OCR-21	100,541	519,555	1,573,492	Naskh, Nastaleeq, Tehreer	Printed	Real World

image features which can then be used in predicting the actual text sequence using RNN layers. The extracted image features are fed to stacked Bi-Directional Long Short Term Memory (LSTM) layers. LSTM being a type of RNN layer predicts the probability distribution of each character at a given timestamp. Thus, generating a sequence of probability distributions which can then be used to predict complete labels consisting of characters. The main advantage of using RNN layers is that these are quite helpful in capturing the contextual information for a given sequence. The reason for using LSTM is that it is effective in capturing longer past contextual information. Let x_t be our input sequence at timestamp t, a_{t-1} be the hidden state vector of the previous timestamp and c_t be the current timestamp cell's state then the following set of equations describe a single unit of LSTM for a given timestamp t:

Update Gate:

$$u_t = sigmoid(W_{ux}x_t + W_{ua}a_{t-1} + b_u) \tag{1}$$

Forget Gate:

$$f_t = sigmoid(W_{fx}x_t + W_{fa}a_{t-1} + b_f) \tag{2}$$

Output Gate:

$$o_t = sigmoid(W_{ox}x_t + W_{oa}a_{t-1} + b_o) \tag{3}$$

Cell Input Activation Vector:

$$\widehat{c}_t = tanh(W_{\widehat{c}x}x_t + W_{\widehat{c}a}a_{t-1} + b_{\widehat{c}}) \tag{4}$$

Cell State Vector:

$$c_t = u_t \circ \widehat{c}_t + f_t \circ c_{t-1} \tag{5}$$

Cell Output Vector:

$$a_t = o_t \circ tanh(c_t) \tag{6}$$

where, $\circ$ represents the Hadamard product. Although the LSTM layer is proficient in carrying past contexts, yet it only has the ability to capture context information in one direction. That is, for any LSTM cell at timestamp t, the context from cells 1 to t-1 are carried out to this cell. Thus, Bi-directional LSTM is introduced here. Bi-directional LSTM is in fact a combination of two LSTM layers, where each has context information flowing in the opposite direction. Hence, Bi-directional LSTMs are able to capture context information from both forward and backward directions. Let a_{ft} be the output of a single LSTM cell for the LSTM layer flowing in the forward direction and a_{bt} be the same for backward direction then the combined output of the two is given as:

$$y_t = W_f a_{ft} + W_b a_{bt} + b_y \tag{7}$$

The main reason to use CTC [27] for calculating the probability of the sequence (either characters or words sequence) is that it enables the use of BLSTM for outputting the sequences for the given image, without having the need to label individual words or characters in images. The CTC loss function for the given probability at each time stamp along with the actual label sums up the score for all the paths/alignments from the input probabilities. It then calculates the negative of log of this sum, which is propagated back in the network. For our dataset U, with Urdu text images U^I along with corresponding labels U^L, one can define the CTC objective function for the i^{th} example point as:

$$p(U_i^L|U_i^I) = \sum_{a=F(y_i)} y_i \tag{8}$$

The function F maps all alignments/paths and returns the score of all the paths that make up our ground truth label U_i^L, collapsing the duplicates and removing the blank CTC tokens. The score for the individual valid alignments is then summed up giving the final score. Also y_i is the sum of probabilities of all tokens, $\mathrm{l} = l_1, l_2, \ldots, l_n$ (including the CTC blank token) that may appear at a given timestamp t. Formally, it is defined as the following:

$$y_i = \prod_{t=1}^{T} p_t(l_t|U_i^I) \tag{9}$$

The CTC loss function, which is used for training the model weights is in fact the negative log of the above calculated likelihood.

$$L = -\sum_{U_i^I U_i^L \in U} \log p(U_i^I, U_i^L) \tag{10}$$

For the prediction, CTC decoder is required which is used with the trained model. It takes the probabilities of each token returned from the RNN at each timestamp and returns the best path which may be based on the highest character probability at every timestamp t.

$$O = \max y_i \tag{11}$$

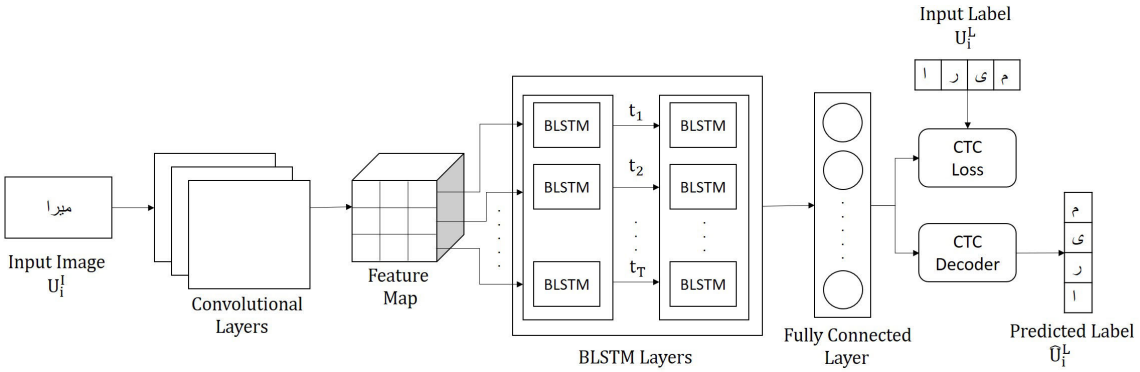

FIGURE 5. Architecture of CNN + BLSTM + CTC.

TABLE 4. CNN + BLSTM + CTC model details.

No	Layer Type	Layer Arguments
		CNN Part
1	Convolutional	Filters: 64, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
2	Max Pool	Pool Size: (2,2), Padding: Same, Stride: (2,2)
3	Convolutional	Filters: 128, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
4	Max Pool	Pool Size: (2,2), Padding: Same, Stride: (2,2)
5	Convolutional	Filters: 256, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
6	Convolutional	Filters: 256, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
7	Max Pool	Pool Size: (2,1), Padding: Same, Stride: (2,2)
8	Convolutional	Filters: 512, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
9	Batch Normalization	Momentum: 0.99, Epsilon: 0.001
10	Convolutional	Filters: 512, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
11	Batch Normalization	Momentum: 0.99, Epsilon: 0.001
12	Max Pool	Pool Size: (2,1), Padding: Same, Stride: (2,1)
13	Convolutional	Filters: 512, Kernel Size: (2,2), Padding: Same, Stride: (1,1), Activation: ReLU
14	Reshape	To map the feature map to feature sequences
		RNN Part
15	BLSTM	Units: 512, Activation: Tanh, Dropout: 0.2
16	BLSTM	Units: 512, Activation: Tanh, Dropout: 0.2
17	Fully Connected	Units: Number of Unique Characters/Words +1 (CTC Blank Token), Activation: Softmax

Finally, we can pass the best path through our filter function G so that the output can again be filtered, collapsing the repeating characters and removing the CTC blank token to get our prediction:

$$\widehat{U}_i^L = G(O) \tag{12}$$

A. CNN + BLSTM + CTC MODEL (CBC)

The architecture of this technique is inspired by an existing architecture [27]. An overview of the architecture is presented in Figure 5.

In particular, two variations of the same model were trained, one for word and the other for text line image recognition. In the first variation, the model was trained to predict the sequence of Urdu text characters at each timestamp. Similarly, in the second variation, a single word was predicted at every timestamp by giving a sequence of words forming a complete text line. The details of the model are given in Table 4.

B. VGG-16 + BLSTM + CTC (VBC)

The aforementioned architecture [27] was also used for this technique. However, this model had its CNN layers replaced with a pre-trained VGG-16 using imagenet weights. Here, the idea was to exploit the already trained CNN for better image feature extraction and then to fine-tune the complete model for Urdu text recognition. The architecture of the model is given in Figure 6.

Similar to the previous architecture, two variations of this transfer learning-based model were trained, one for Urdu word images and other for the Urdu text line images. The details of network are given in Table 5.

C. ENCODER DECODER MODEL (EDM)

The encoder decoder model [73] has been used in predicting sequences with its wide applicability in tasks such as machine translation. An OCR model [74] was used for printed text using the same encoder decoder approach. Where, the encoder part consists of CNN + LSTM layers that extracted the image features which were then passed to the decoder as the initial state which generated the most probable text sequence. This architecture has been used in this study, where Urdu text line image was passed as input to the encoder and a sequence of most probable Urdu words at each timestamp t was outputted by the decoder. The encoder

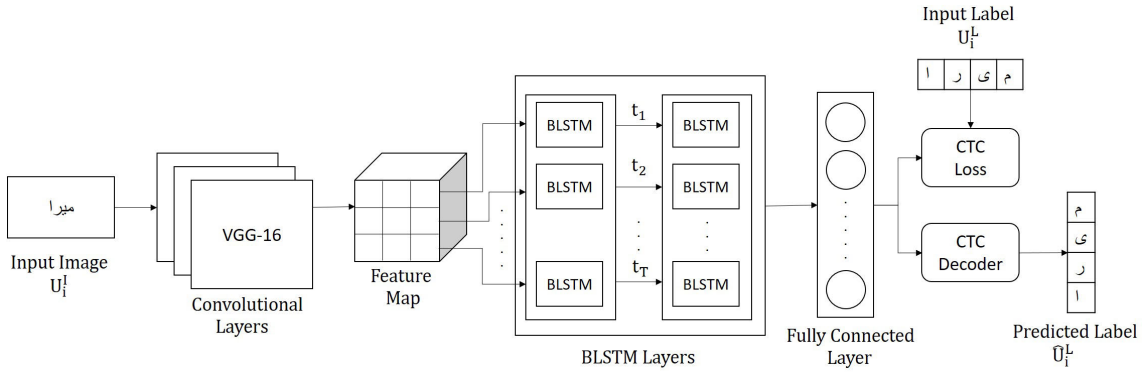

FIGURE 6. Architecture of VGG-16 + BLSTM + CTC.

TABLE 5. VGG-16 + BLSTM + CTC model details.

No	Layer Type	Layer Arguments
		CNN (VGG-16) Part
1	Convolutional	Filters: 64, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
2	Convolutional	Filters: 64, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
3	Max Pool	Pool Size: (2,2), Padding: Same, Stride: (2,2)
4	Convolutional	Filters: 128, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
5	Convolutional	Filters: 128, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
6	Max Pool	Pool Size: (2,2), Padding: Same, Stride: (2,2)
7	Convolutional	Filters: 256, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
8	Convolutional	Filters: 256, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
9	Convolutional	Filters: 256, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
10	Max Pool	Pool Size: (2,2), Padding: Same, Stride: (2,2)
11	Convolutional	Filters: 512, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
12	Convolutional	Filters: 512, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
13	Convolutional	Filters: 512, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
14	Max Pool	Pool Size: (2,2), Padding: Same, Stride: (2,2)
15	Convolutional	Filters: 512, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
16	Convolutional	Filters: 512, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
17	Convolutional	Filters: 512, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
18	Reshape	To map the feature map to feature sequences
		RNN Part
19	BLSTM	Units: 256, Activation: Tanh, Dropout: 0.5
20	BLSTM	Units: 256, Activation: Tanh, Dropout: 0.5
21	Fully Connected	Units: Number of Unique Characters/Words +1 (CTC Blank Token), Activation: Softmax

part consisting of CNN layers along with LSTM layer can be defined as:

$$X_i = F_{enc}(U_i^I) \tag{13}$$

For the input text image, it generates a feature vector X_i. The decoder part, consisting of LSTM cells, given this feature vector can then be used to generate the probability at each timestamp of every word.

$$P(U_i^L|U_i^I) = F_{dec}(X_i) \tag{14}$$

For the purpose of training, the decoder was initialized with state of the encoder and also fed with the actual labels so that it can maximize the probability distribution at each timestamp t.

$$P(U_i^L|U_i^I) = \prod_{t=1}^{T} P(c_t|c_1, \ldots\ldots c_{t-1}, X_i) \tag{15}$$

For the purpose of prediction, the maximum probability of a word at every timestamp t is generated as an output to generate a final sequence. Figure 7 illustrates the architecture of the model used in this study. The details of the model are given in Table 6.

$$\hat{U}_i^L = argmax(P(U_i^L|U_i^I)) \tag{16}$$

VI. EXPERIMENTS

We performed comprehensive experimentation using deep learning techniques to show the effectiveness of our developed resources. Firstly, we introduce the performance evaluation measures followed by the experimental setup. Secondly, results and discussion of the experiments are presented for multiple fonts under fixed settings. Thirdly, techniques for selecting optimal parameters per model along with the results achieved for optimal parameter settings is discussed. Finally, an error analysis of the results is performed.

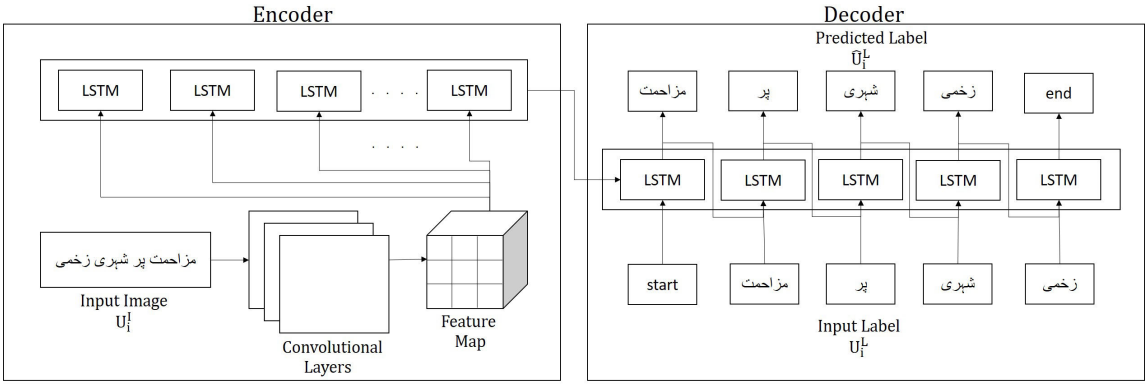

FIGURE 7. An illustration of Seq2Seq architecture.

TABLE 6. Encoder decoder model details.

No	Layer Type	Layer Arguments
		Encoder
1	Convolutional	Filters: 16, Kernel Size: (7,7), Padding: Same, Stride: (1,1), Activation: ReLU
2	Batch Normalization	Momentum: 0.99, Epsilon: 0.001
3	Max Pool	Pool Size: (2,2), Padding: Valid, Stride: (2,2)
4	Convolutional	Filters: 32, Kernel Size: (5,5), Padding: Same, Stride: (1,1), Activation: ReLU
5	Batch Normalization	Momentum: 0.99, Epsilon: 0.001
6	Max Pool	Pool Size: (2,2), Padding: Valid, Stride: (2,2)
7	Convolutional	Filters: 64, Kernel Size: (5,5), Padding: Same, Stride: (1,1), Activation: ReLU
8	Batch Normalization	Momentum: 0.99, Epsilon: 0.001
9	Max Pool	Pool Size: (2,2), Padding: Valid, Stride: (2,2)
10	Convolutional	Filters: 128, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
11	Batch Normalization	Momentum: 0.99, Epsilon: 0.001
12	Max Pool	Pool Size: (2,2), Padding: Valid, Stride: (2,2)
13	Convolutional	Filters: 256, Kernel Size: (3,3), Padding: Same, Stride: (1,1), Activation: ReLU
14	Batch Normalization	Momentum: 0.99, Epsilon: 0.001
15	Max Pool	Pool Size: (2,2), Padding: Valid, Stride: (2,2)
16	Reshape	To map the feature map to feature sequences
17	LSTM	Units: 256, Activation: Tanh, Dropout: 0.5
		Decoder
18	LSTM	Units: 512, Activation: Tanh
19	Fully Connected	Units: Number of Unique Words, Activation: Softmax

A. PERFORMANCE EVALUATION MEASURES

Two established measures to evaluate the effectiveness of deep-learning techniques for their ability to identify the text sequence from input images are used. These measures are, Character Error Rate (CER) and Word Error Rate (WER). A lower value of these measures represents a higher effectiveness of a technique and vice versa. The reason for the choice of these measures stems from the fact that these measures have been widely used to evaluate the effectiveness of OCR and speech recognition systems [59], [61], [66]. These measures are based on the Levenshtein distance [75] which measures similarity between two strings. These measures are defined as follows:

CER is the ratio between the Levenshtein distance of the actual and predicted characters and the maximum length of the actual and predicted excerpt of characters. For the image I, if G(C, I) is the character level ground truth, P(C, I) is the character level excerpt predicted, and Lev(G, P) is the Levenshtein distance between G(C, I) and P(C, I). The CER of I is defined as follows:

$$CER_I = \frac{LEV_{(C,I)}}{max(|G_{(C,I)}|, |P_{(C,I)}|)} \tag{17}$$

A set S for testing is used having N number of text-line or word images, the CER_M is defined as follows:

$$CER_M = \frac{1}{N}\sum_{I=0}^{n} CER_I \tag{18}$$

Similarly, WER is the ratio between the Levenshtein distance of the actual and predicted words and the maximum length of actual and predicted excerpt of words. Consider an input image I, if G(W, I) is the character level ground truth, P(W, I) is the character level prediction, and Lev(G, P) is the Levenshtein distance between G(W, I) and P(W, I). The WER of I is defined as follows:

$$WER_I = \frac{LEV_{(W,I)}}{max(|G_{(W,I)}|, |P_{(W,I)}|)} \tag{19}$$

TABLE 7. Results of comparative analysis under fixed settings.

Input corpus	Model	Training Loss	Validation Loss		CER	WER
Text-line level	CNN+BLSTM+CTC	0.166	4.335	Train	0.002	0.003
				Validation	0.097	0.128
				Test	0.099	0.158
	VGG-16 + BLSTM + CTC	10.238	19.322	Train	0.501	0.669
				Validation	0.597	0.785
				Test	0.541	0.732
	Encoder decoder technique	0.004	0.6074	Train	0.001	0.002
				Validation	0.119	0.152
				Test	0.123	0.166
Word level	CNN+BLSTM+CTC	0.276	0.327	Train	0.005	0.040
				Validation	0.016	0.092
				Test	0.018	0.106
	VGG-16+BLSTM+CTC	0.046	0.058	Train	0.004	0.011
				Validation	0.018	0.063
				Test	0.016	0.065

A set of S is used for testing having N number of text-line or word images. The WER_M is defined as follows:

$$WER_M = \frac{1}{N}\sum_{I=0}^{n} WER_I \tag{20}$$

B. EXPERIMENTAL SETUP

MMU-OCR-21 corpus is composed of text-line, word, and character level images in three different fonts, Naskh, Nastaleeq and Tehreer. Experiments were performed using all the word and text-line images. More specifically, 300,000 word images and 301,623 text-line images were used for the experiments. However, experiments were not performed on the 849 character images due to the following reasons: (a) the count of character level images is very small having inadequate variation. Therefore, it cannot be used for the training of deep learning techniques. And, (b) an Urdu OCR that takes the images having merely single isolated characters do not have real-world applications and such type of OCR is not of higher value for Urdu language due to the joining nature of its alphabets.

Experiments are performed using two state-of-the-art deep learning techniques, CNN + BLSTM + CTC (CBC) and VGG-16 + BLSTM + CTC (VBC). The detailed architecture of each technique and the optimal settings of the parameters that were used in this study are presented in the preceding section. In addition to these two techniques, Encoder Decoder Model (EDM) architecture was also used for experimentation at text-line level due to two reasons. Firstly, because EDM is a sequence-to-sequence model that has the ability to effectively capture contextual information, where the preceding words determine the probability of the subsequent word in addition to the features of the input images. Secondly, the EDM architecture allows to work with varying length sequences and it has been widely used for similar tasks, such as machine translation [73]. All the experiments were implemented using Tensorflow2 Keras API.

For the experimentation, 301,623 text-line images were randomly divided into three parts. Where, 80% of the images were used for training, 10% were used for validation, and the remaining 10% were used for testing. The exclusively random choice of samples without considering the font in the image was performed in order to ensure that the learning can be performed in the diverse settings. For an unbiased comparative analysis, firstly, all the models were trained using fixed settings. Under these fixed settings all of the models were trained for 500 epochs with a mini-batch size of 64 using Adam for optimizing the objective functions [76] and with the default learning rate of 0.001. The choice of Adam optimizer was made considering its benefits over the conventional Stochastic Gradient Descent [76], along with considering the fact that it has been widely used for training models for text recognition and other image processing tasks [77]–[79]. Table 7 presents a comparative analysis of the performance of all the models under fixed settings using equations 18 and 20 in order to calculate CER_M and WER_M, respectively. A comparative analysis of the results revealed that the CER and WER scores were lowest for the CBC model at text line level. Also, at word level, VBC model outperformed the other models as it achieved the lowest CER and WER scores of 0.016 and 0.065, respectively.

Figure 8 illustrates the details of training the models. The training and validation loss achieved at the end of each epoch is plotted in the figure. It can be observed from the figure that different models tend to achieve a combination of the best validation and training loss at different epochs.

C. BEST FIT MODELS

We used early stopping [80] to achieve the optimal parameters for each of the aforementioned techniques for which the validation loss was minimum. The details of the best fit model attained using early stopping are presented in Table 8. The results and analysis in the subsequent sections are all performed on the best version achieved for each model.

1) MIXED FONT

Table 9 provides an overview of the results achieved by our best fit models on randomly sampled test, validation and training datasets. It can be observed that at word level the

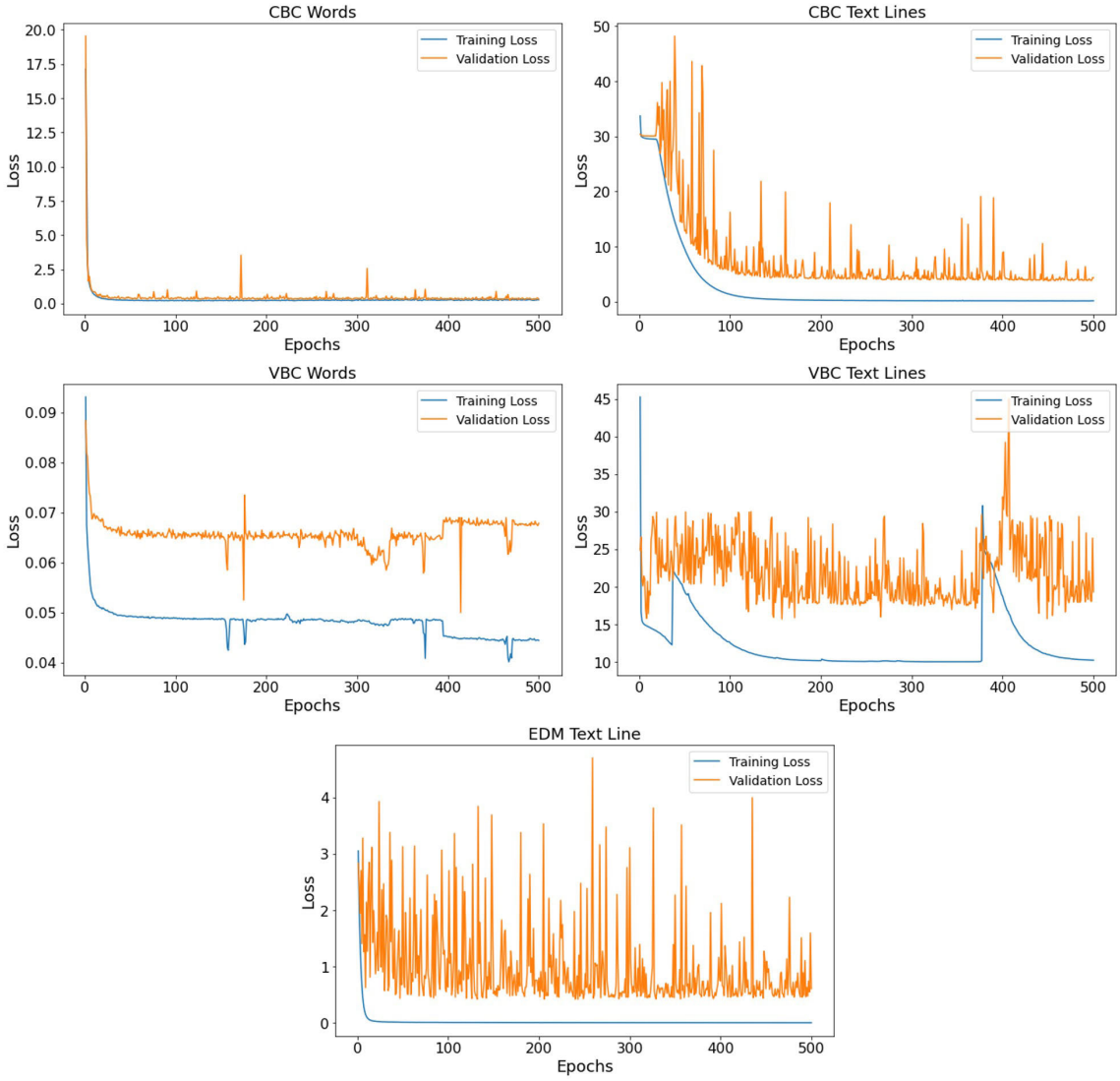

FIGURE 8. Training and validation loss at each epoch.

TABLE 8. Early stopping based best validation loss for each technique.

Input corpus	Model	Validation loss	Epoch
Text-line level	CNN+BLSTM+CTC	3.801	451
	VGG-16 + BLSTM + CTC	15.708	157
	Encoder Decoder model	0.421	243
Word level	CNN+BLSTM+CTC	0.239	468
	VGG-16+BLSTM+CTC	0.050	414

CER and WER scores of CBC and VBC are either identical or comparable. However, at text line level, CBC outperforms VBC. Also, at text line level EDM has the lowest WER which very much conforms to the ability of EDM to better cope with sequences of words.

Figure 9 shows a pictorial representation of the results. It can be observed from the figure that there is a little difference in CER and WER for the training, validation and test sets. This validates the fact that the trained models are not overfitting and they generalized at both word and text line levels. Also, it can be observed from the figure that WER is higher than CER in every case [61], [66]. It is due to the fact that WER considers words as a single unit and a word is marked as an error even if there is a variation of a single character in it. Contrary to this, the error count is computed at character level in case of CER, therefore the value of CER increases gradually.

2) INDIVIDUAL FONTS

To further analyze that our models have generalized well for all fonts, and that none of the models overfits for a single font, we tested the performance of each trained model on a sample consisting of single font images. This was done by randomly selecting a sample of 2% of the total images per font at word and text line level and then using each sample for testing all the techniques discussed earlier. The process was repeated

TABLE 9. Results of the best fit models for each technique.

Input corpus			CER	WER
Text-line level	CNN+BLSTM+CTC	Train	0.001	0.001
		Validation	0.074	0.105
		Test	0.072	0.103
	VGG-16 + BLSTM + CTC	Train	0.355	0.492
		Validation	0.490	0.660
		Test	0.490	0.662
	Encoder decoder technique	Train	0.001	0.002
		Validation	0.074	0.089
		Test	0.073	0.088
Word level	CNN+BLSTM+CTC	Train	0.005	0.020
		Validation	0.012	0.047
		Test	0.011	0.047
	VGG-16+BLSTM+CTC	Train	0.005	0.023
		Validation	0.011	0.050
		Test	0.011	0.050

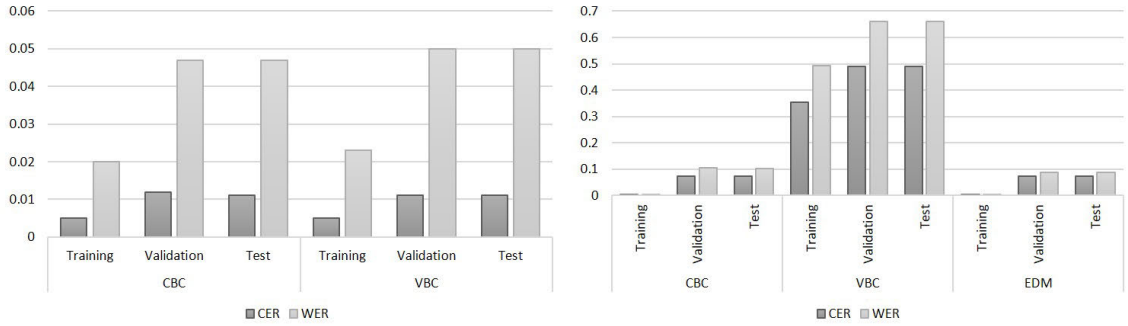

FIGURE 9. Results of mixed fonts. Word level graph on left and text line graph on right.

TABLE 10. Text-line level and word level results for individual fonts.

Input Corpus	Technique	Font	Average CER	Average WER
Text-line level	CNN+BLSTM+CTC	Naskh	0.0339	0.0562
		Nastaleeq	0.0473	0.0637
		Tehreer	0.0351	0.0575
	VGG16+BLSTM+CTC	Naskh	0.3738	0.4805
		Nastaleeq	0.4338	0.5616
		Tehreer	0.3609	0.4708
	Encoder Decoder	Naskh	0.0294	0.0498
		Nastaleeq	0.0418	0.0570
		Tehreer	0.0305	0.0506
Word level	CNN+BLSTM+CTC	Naskh	0.0102	0.0443
		Nastaleeq	0.0130	0.0510
		Tehreer	0.0116	0.0473
	VGG16+BLSTM+CTC	Naskh	0.0104	0.0476
		Nastaleeq	0.0129	0.0558
		Tehreer	0.0115	0.0499

five times and the average CER and WER were calculated per font. The detailed results are presented in Table 10.

Figure 10, Figure 11 and Figure 12 illustrate the results that were achieved per sample for each font, whereas Figure 13 gives an overview of the average results. The key observation here is that, there is no significant variation in the performance of all the models for all the fonts. This validates the hypothesis that our models generalized well regardless of the fonts. It can also be observed from the figure that the lowest error rates for all the techniques were achieved for Naskh font images, whereas the error rates for the other two fonts were slightly higher. Furthermore, the observation is valid for both word level and text line level. A possible reason for this is that the writing style of Naskh is simpler than Nastaleeq and Tehreer, and is easier for the learning of models. However, the variation is quite negligible.

D. ERROR ANALYSIS

Recall from the results that a very high accuracy was achieved at word level, whereas a below par accuracy was achieved

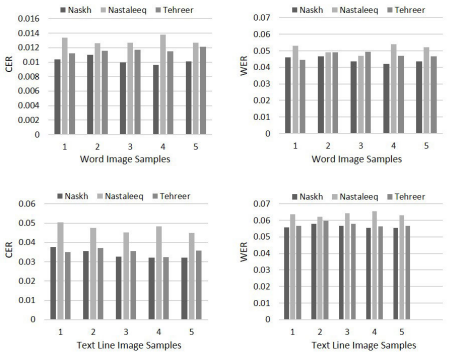

FIGURE 10. Per font evaluation results for CBC technique.

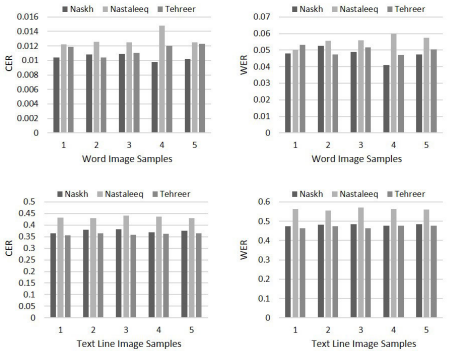

FIGURE 11. Per font evaluation results for VBC technique.

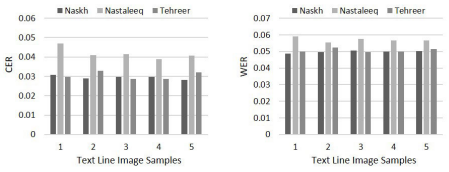

FIGURE 12. Per font evaluation results for EDM technique.

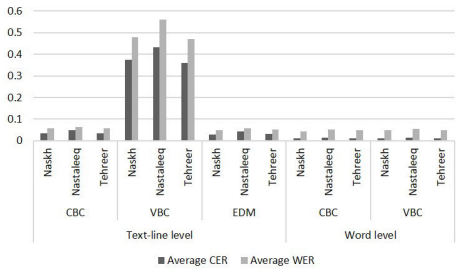

FIGURE 13. Average error rates for individual fonts.

for text line level by all the techniques. To understand the underlying reason for the higher WER, all images in the testing dataset that erred for either EDM or CBC technique, were separated. Subsequently, the intersection of the set of

TABLE 11. Character count distribution of error set.

Character Count	Total Example Points	Percentage
11	41	0.74
12	84	1.51
13	156	2.81
14	321	5.77
15	547	9.84
16	817	14.7
17	1242	22.34
18	1586	28.53
19	765	13.76

TABLE 12. Word count distribution of error set.

Word Count	Total Example Points	Percentage
4	4378	78.76
5	1109	19.95
6	70	1.26
7	2	0.04

images with EDM errors and CBC errors was taken for further analysis. The generated Error Set consisted of a count of 5,559 example points having 5,680 unique words. The reason to generate the Error Set was to identify the root causes of the erroneous performance of both the EDM and CBC techniques on these example points. Analyses on the length variation of the example points and count variation of words and bigrams in the example points, were performed. The details of the analyses are as follows:

For the length variation analysis, our Error Set was divided based on the number of characters that exist in an example point. Where, the character count consists of the number of characters in the text line, including the space character and the repeated characters. This was done to find out if the number of characters in a text line had some impact on the performance of our two techniques. It was observed from the results that a large number of text lines in the Error Set had characters count between 15 and 19. This indicates that higher number of characters in a text line causes errors in prediction.

For the count variation analysis, the Error Set was divided based on the Count of Words in each example point. It was observed that a large portion of example points had a word count value of 4. Based on these results we deduce that text lines with a fewer but longer words tend to have a higher chance of error in prediction than the ones with larger number words having a smaller length. The possible reason for this behavior is that longer words are usually uncommon and a combination of such longer words may not be easy for the models to learn properly. For example, a word consisting of 12 characters occurs only once in the corpus. Further details of the two analyses are given in Table 11 and Table 12, respectively.

Figure 14 shows the 20 most confused words and their counts in the text line level corpus. It can be observed that most of the error words have similarity in terms of characters. That is, many incorrectly predicted words have similar characters, and they have the same order.

Most Confused Words				Most Confused Bigram			
Actual		Predicted		Actual		Predicted	
Word	Occurrences	Word	Occurrences	Bigram	Occurrences	Bigram	Occurrences
سپیشل	42	سیشن	41	نے:لکھا	94	نے:کہا	719
گئے	968	ہوگئے	127	کے:محتاج	2	کے:مطابق	587
نو	171	تو	925	کے:حصول	3	کے:مطابق	587
بی	958	پی	2745	سے:بڑھا	7	سے:بڑھ	492
ہو	1915	ہوگا	507	ان:بچوں	5	ان:کی	406
قدر	58	قد	13	کا:کہا	6	کا:کہنا	254
سنچری	30	سینچری	5	اس:امر	4	اس:کا	252
سر	198	پر	5053	کم:یہ	17	کیا:یہ	53
بجے	64	ہو	1915	کے:بنے	3	کے:لیے	219
کار	258	کا	7915	کے:پے	1	کے:لیے	219
سو	121	ہو	1915	کے:گندے	1	کے:لیے	219
خود	195	خودکش	14	علم:کی	2	فلم:کی	33
بعد	390	بند	684	خون:کا	1	ان:کا	360
کی	7627	کمی	266	کیا:غلط	3	کیا:گیا	195
سیکورٹی	26	سیکیورٹی	17	کے:علاہ	1	کے:علاوہ	193
ہوتی	107	ہوئی	378	اس:غلطی	1	اس:سے	188
نائب	312	عرب	92	کی:بجلی	3	کی:گئی	161
دھیان	3	خیال	89	کی:آپریٹر	1	کی:گئی	161
دیا	787	گیا	1719	کی:فکر	12	کی:گئی	161
اکرام	55	اکرم	131	سے:بڑا	16	سے:زائد	142

FIGURE 14. Most confused words/bigrams and their counts.

Another important aspect of our RNN based models is their ability to learn from the context. Keeping that in mind, a further analysis was performed. That is, we generated bigrams for each of the confused words from the Error Set and compared the number of occurrences of each bigram to see if common bigrams are equally confused. We generated bigram pairs for the actual and the confused words from the actual and predicted text lines, respectively. For that, bigram pairs were generated for both the previous and next word in the sequence, making two pairs per confused word in an example point. Subsequently, we identified the 20 most confused bigram pairs and calculated their count in the text line corpus. The Figure 14 shows the 20 bigram pairs that are mostly confused. It can be observed from the results that the less occurring bigrams are mostly confused with bigrams that occur frequently in the corpus. The possible reason for such behavior is that the model generalizes better for higher count bigrams than those that occurs less frequently. However, this reasoning may very much be challenged by the fact that our models rely on much larger contexts rather than only on the context of the previous word, thus this does not have too much an effect on the overall learning of the models.

VII. CONCLUSION

Urdu OCR is a key topic of research, as Urdu is a prominent language having several application areas. Despite the presence of several Urdu corpora, none was found to be adequate in-terms of size. That is, the existing datasets are not sufficiently large, neither do they support the diversity of fonts nor the varying levels, Text Line and Word Level. To that end, we have developed a printed Urdu text corpus along with the corresponding OCR benchmark. In particular, the corpus supports multiple levels and multiple fonts, hence fulfilling the minimum requirements for developing end-to-end deep learning models. We also evaluated our proposed corpus using deep learning models for word and text line level OCRs. The study concludes that most of the deep learning models used for experimentation generalized well on our developed corpus and they were able to achieve remarkable CER and WER scores. We also analyzed the performance of our models for individual fonts to establish that the models were effectively generalized for all the fonts.

Despite the fact that we have generated the largest-ever Urdu OCR corpus, a further expansion of our dataset in terms of both volume and variety is desired. For instance, it is desired to introduce other Urdu fonts, such as Tuluth, Kofi and Riqa. Furthermore, other variations can be made to the rendering settings of images, such as changing font size, foreground colour, and background colour. Apart from that, certain data augmentation techniques can be used for training our models which in turn can further contribute towards better generalization of deep learning models. Other directions for future research include the use of ensembled and attention-based models to build even better performing end-to-end segmentation free Urdu OCR systems. Finally, our proposed technique can be used to solve the OCR problem for other South Asian languages, such as Punjabi and Sindhi.

REFERENCES

[1] R. H. Davis and J. Lyall, "Recognition of handwritten characters—A review," *Image Vis. Comput.*, vol. 4, no. 4, pp. 208–218, 1986.

[2] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: A comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 63–84, Jan. 2000.

[3] N. A. M. Isheawy and H. Hasan, "Optical character recognition (OCR) system," *IOSR J. Comput. Eng. (IOSR-JCE), e-ISSN*, vol. 17, no. 2, pp. 22–26, Mar. 2015.

[4] G. R. Goncalves, M. A. Diniz, R. Laroca, D. Menotti, and W. R. Schwartz, "Real-time automatic license plate recognition through deep multi-task networks," in *Proc. 31st SIBGRAPI Conf. Graph., Patterns Images (SIBGRAPI)*, Oct. 2018, pp. 110–117.

[5] M. T. Qadri and M. Asif, "Automatic number plate recognition system for vehicle identification using optical character recognition," in *Proc. Int. Conf. Educ. Technol. Comput.*, 2009, pp. 335–338.

[6] H. Fujisawa, "Forty years of research in character and document recognition—An industrial perspective," *Pattern Recognit.*, vol. 41, no. 8, pp. 2435–2446, 2008.

[7] S. España-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez, "Improving offline handwritten text recognition with hybrid HMM/ANN models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 4, pp. 767–779, Apr. 2011.

[8] M. Mohamed and P. Gader, "Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 5, pp. 548–554, May 1996.

[9] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 8, pp. 787–808, Aug. 1990.

[10] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 545–552.

[11] T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait, "High-performance OCR for printed English and fraktur using LSTM networks," in *Proc. 12th Int. Conf. Document Anal. Recognit.*, Aug. 2013, pp. 683–687.

[12] C. E. Dunn and P. S. P. Wang, "Character segmentation techniques for handwritten text—A survey," in *Proc. 11th IAPR Int. Conf. Pattern Recognit. Conf. B, Pattern Recognit. Methodol. Syst.*, vol. 1, 1992, pp. 577–578.

[13] M. Cesar and R. Shinghal, "An algorithm for segmenting handwritten postal codes," *Int. J. Man-Mach. Stud.*, vol. 33, no. 1, pp. 63–80, Jul. 1990.

[14] M. Cheriet, Y. S. Huang, and C. Y. Suen, "Background region-based algorithm for the segmentation of connected digits," in *Proc. 11th Int. Conf. Pattern Recognit. Conf. B, Pattern Recognit. Methodol. Syst. (IAPR)*, vol. 1, 1992, pp. 619–620.

[15] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 7, pp. 690–706, Jul. 1996.

[16] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1764–1772.

[17] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[18] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand. (ASRU)*, Dec. 2015, pp. 167–174.

[19] E. Greenstein and D. Penner, "Japanese-to-English machine translation using recurrent neural networks," *Retrieved*, vol. 19, p. 2019, Aug. 2015.

[20] D. Hingu, D. Shah, and S. S. Udmale, "Automatic text summarization of Wikipedia articles," in *Proc. Int. Conf. Commun., Inf. Comput. Technol. (ICCICT)*, Jan. 2015, pp. 1–4.

[21] S. Sah, S. Kulhare, A. Gray, S. Venugopalan, E. Prud'Hommeaux, and R. Ptucha, "Semantic text summarization of long videos," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 989–997.

[22] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2048–2057.

[23] A. Wang, H. Hu, and L. Yang, "Image captioning with affective guiding and selective attention," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 14, no. 3, pp. 1–15, Aug. 2018.

[24] P. Doetsch, M. Kozielski, and H. Ney, "Fast and robust training of recurrent neural networks for offline handwriting recognition," in *Proc. 14th Int. Conf. Frontiers Handwriting Recognit.*, Sep. 2014, pp. 279–284.

[25] N. Malik and A. Singh, "Character recognition of offline handwritten devanagari script using artificial neural network," *Int. J. Adv. Comput. Res.*, vol. 2, no. 2, pp. 35–41, 2016.

[26] M. Namysl and I. Konya, "Efficient, lexicon-free OCR using deep learning," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2019, pp. 295–301.

[27] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, Nov. 2016.

[28] P. Natarajan, I. Bazzi, Z. Lu, J. Makhoul, and R. Scwhartz, "Robust OCR of degraded documents," in *Proc. 5th Int. Conf. Document Anal. Recognit. (ICDAR)*, 1999, pp. 357–361.

[29] T. V. Ashwin and P. S. Sastry, "A font and size-independent OCR system for printed Kannada documents using support vector machines," *Sadhana*, vol. 27, no. 1, pp. 35–58, Feb. 2002.

[30] A. Ul-Hasan, S. B. Ahmed, F. Rashid, F. Shafait, and T. M. Breuel, "Offline printed Urdu nastaleeq script recognition with bidirectional LSTM networks," in *Proc. 12th Int. Conf. Document Anal. Recognit.*, Aug. 2013, pp. 1061–1065.

[31] M. Jain, M. Mathew, and C. V. Jawahar, "Unconstrained OCR for Urdu using deep CNN-RNN hybrid networks," in *Proc. 4th IAPR Asian Conf. Pattern Recognit. (ACPR)*, Nov. 2017, pp. 747–752.

[32] S. Naz, K. Hayat, M. I. Razzak, M. W. Anwar, S. A. Madani, and S. U. Khan, "The optical character recognition of Urdu-like cursive scripts," *Pattern Recognit.*, vol. 47, no. 3, pp. 1229–1248, Mar. 2014.

[33] N. Sabbour and F. Shafait, "A segmentation-free approach to Arabic and Urdu OCR," in *Document Recognition and Retrieval XX*, vol. 8658. Bellingham, WA, USA: SPIE, 2013, Art. no. 86580N.

[34] Q. U. A. Akram and S. Hussain, "Improving Urdu recognition using character-based artistic features of Nastalique calligraphy," *IEEE Access*, vol. 7, pp. 8495–8507, 2019.

[35] M. J. Rafeeq, Z. U. 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, 2019.

[36] A. Qurrat-Ul, A. Niazi, F. Adeeba, S. Urooj, S. Hussain, and S. Shams, "A comprehensive image dataset of Urdu Nastalique document images," in *Proc. Conf. Lang. Technol.*, 2016, pp. 81–88.

[37] S. B. Ahmed, S. Naz, S. Swati, I. Razzak, A. I. Umar, and A. A. Khan, "UCOM offline dataset-an Urdu handwritten dataset generation," *Int. Arab J. Inf. Technol.*, vol. 14, no. 2, pp. 1–7, Mar. 2017.

[38] M. W. Sagheer, C. L. He, N. Nobile, and C. Y. Suen, "A new large Urdu database for off-line handwriting recognition," in *Proc. Int. Conf. Image Anal. Process.* Berlin, Germany: Springer, 2009, pp. 538–546.

[39] S. B. Ahmed, S. Naz, S. Swati, and M. I. Razzak, "Handwritten Urdu character recognition using one-dimensional BLSTM classifier," *Neural Comput. Appl.*, vol. 31, no. 4, pp. 1143–1151, 2019.

[40] P. Choudhary and N. Nain, "An annotated Urdu corpus of handwritten text image and benchmarking of corpus," in *Proc. 37th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*. USA: IEEE, 2014, pp. 1159–1164.

[41] P. Choudhary and N. Nain, "A four-tier annotated Urdu handwritten text image dataset for multidisciplinary research on Urdu script," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 15, no. 4, pp. 1–23, Jun. 2016.

[42] S. A. Malik, M. Maqsood, F. Aadil, and M. F. Khan, "An efficient segmentation technique for Urdu optical character recognizer (OCR)," in *Proc. Future Inf. Commun. Conf.* San Fransisco, CA, USA: Springer, 2019, pp. 131–141.

[43] 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.

[44] D. Firmani, P. Merialdo, E. Nieddu, and S. Scardapane, "*In codice ratio*: OCR of handwritten Latin documents using deep convolutional networks," in *Proc. AI CH AI IA*, 2017, pp. 9–16.

[45] S. Srivastava, J. Priyadarshini, S. Gopal, S. Gupta, and H. S. Dayal, "Optical character recognition on bank cheques using 2D convolution neural network," in *Applications of Artificial Intelligence Techniques in Engineering*. Singapore: Springer, 2019, pp. 589–596.

[46] J. Adriano, K. Calma, N. Lopez, J. Parado, L. Rabago, and J. Cabardo, "Digital conversion model for hand-filled forms using optical character recognition (OCR)," in *Proc. IOP Conf., Mater. Sci. Eng.*, vol. 482, 2019, Art. no. 012049.

[47] M. W. Sagheer, C. L. He, N. Nobile, and C. Y. Suen, "Holistic Urdu handwritten word recognition using support vector machine," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 1900–1903.

[48] I. Shamsher, Z. Ahmad, J. K. Orakzai, and A. Adnan, "OCR for printed Urdu script using feed forward neural network," *World Acad. Sci., Eng. Technol.*, vol. 23, pp. 172–175, Aug. 2007.

[49] J. Tariq, U. Nauman, and M. U. Naru, "Softconverter: A novel approach to construct OCR for printed Urdu isolated characters," in *Proc. 2nd Int. Conf. Comput. Eng. Technol.*, vol. 3, 2010, p. V3-495.
[50] Z. Ahmad, J. K. Orakzai, I. Shamsher, and A. Adnan, "Urdu Nastaleeq optical character recognition," *World Acad. Sci., Eng. Technol.*, vol. 26, pp. 249–252, Dec. 2007.
[51] Q. Akram, S. Hussain, F. Adeeba, S. Rehman, and M. Saeed, "Framework of Urdu Nastalique optical character recognition system," in *Proc. Conf. Lang. Technol. (CLT)*, 2014, pp. 23–30.
[52] S. T. Javed, S. Hussain, A. Maqbool, S. Asloob, S. Jamil, and H. Moin, "Segmentation free Nastalique Urdu OCR," *World Acad. Sci., Eng. Technol.*, vol. 46, no. 10, pp. 456–461, 2010.
[53] 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.
[54] G. S. Lehal and A. Rana, "Recognition of Nastalique Urdu ligatures," in *Proc. 4th Int. Workshop Multilingual (OCR MOCR)*, 2013, pp. 1–5.
[55] T. Ali, T. Ahmad, and M. Imran, "UOCR: A ligature based approach for an Urdu OCR system," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, 2016, pp. 388–394.
[56] M. K. Hayat, A. Daud, A. A. Alshdadi, A. Banjar, R. A. Abbasi, Y. Bao, and H. Dawood, "Towards deep learning prospects: Insights for social media analytics," *IEEE Access*, vol. 7, pp. 36958–36979, 2019.
[57] W. Khan, A. Daud, F. Alotaibi, N. Aljohani, and S. Arafat, "Deep recurrent neural networks with word embeddings for Urdu named entity recognition," *ETRI J.*, vol. 42, no. 1, pp. 90–100, Feb. 2020.
[58] M. Busta, L. Neumann, and J. Matas, "Deep TextSpotter: An end-to-end trainable scene text localization and recognition framework," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2204–2212.
[59] T. C. Wei, U. U. Sheikh, and A. A.-H.-A. Rahman, "Improved optical character recognition with deep neural network," in *Proc. IEEE 14th Int. Colloq. Signal Process. Appl. (CSPA)*, Mar. 2018, pp. 245–249.
[60] R. Achanta and T. Hastie, "Telugu OCR framework using deep learning," 2015, *arXiv:1509.05962*. [Online]. Available: http://arxiv.org/abs/1509.05962
[61] D. Paul and B. B. Chaudhuri, "A BLSTM network for printed Bengali OCR system with high accuracy," 2019, *arXiv:1908.08674*. [Online]. Available: http://arxiv.org/abs/1908.08674
[62] A. Daud, W. Khan, and D. Che, "Urdu language processing: A survey," *Artif. Intell. Rev.*, vol. 47, no. 3, pp. 279–311, 2017.
[63] I. Supriana and A. Nasution, "Arabic character recognition system development," *Proc. Technol.*, vol. 11, pp. 334–341, Jan. 2013.
[64] A. Zidouri, "On multiple typeface Arabic script recognition," *Res. J. Appl. Sci. Eng. Technol.*, vol. 2, no. 5, pp. 428–435, 2010.
[65] C. J. Hilditch, "Linear skeletons from square cupboards," in *Machine Intelligence IV*, B. Meltzer and D. Michie, Eds. U.K.: Edinburgh Univ. Press, 1969, pp. 403–420.
[66] S. Rawls, H. Cao, E. Sabir, and P. Natarajan, "Combining deep learning and language modeling for segmentation-free OCR from raw pixels," in *Proc. 1st Int. Workshop Arabic Script Anal. Recognit. (ASAR)*, Apr. 2017, pp. 119–123.
[67] A. Pal, "Bengali handwritten numeric character recognition using denoising autoencoders," in *Proc. IEEE Int. Conf. Eng. Technol. (ICETECH)*, Mar. 2015, pp. 1–6.
[68] I. Ahmad, X. Wang, R. Li, and S. Rasheed, "Offline Urdu Nastaleeq optical character recognition based on stacked denoising autoencoder," *China Commun.*, vol. 14, no. 1, pp. 146–157, Jan. 2017.
[69] 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.
[70] S. Naz, A. I. Umar, R. Ahmed, M. I. Razzak, S. F. Rashid, and F. Shafait, "Urdu Nasta'liq text recognition using implicit segmentation based on multi-dimensional long short term memory neural networks," *SpringerPlus*, vol. 5, no. 1, p. 2010, Dec. 2016.
[71] S. Naz, S. B. Ahmed, R. Ahmad, and M. I. Razzak, "Zoning features and 2DLSTM for Urdu text-line recognition," in *Proc. KES*, 2016, pp. 16–22.
[72] 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.
[73] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[74] D. K. Sahu and M. Sukhwani, "Sequence to sequence learning for optical character recognition," 2015, *arXiv:1511.04176*. [Online]. Available: http://arxiv.org/abs/1511.04176
[75] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys.-Dokl.*, vol. 10, no. 8, pp. 707–710, 1966.
[76] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[77] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence-video to text," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4534–4542.
[78] J. Hu, T. Guo, J. Cao, and C. Zhang, "End-to-end Chinese text recognition," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2017, pp. 1407–1411.
[79] A. Chowdhury and L. Vig, "An efficient end-to-end neural model for handwritten text recognition," 2018, *arXiv:1807.07965*. [Online]. Available: http://arxiv.org/abs/1807.07965
[80] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 1998, pp. 55–69.

TAYYAB NASIR received B.S. and M.Phil. degrees in computer science from Punjab University College of Information Technology (PUCIT), University of the Punjab, Lahore, Pakistan, in 2017 and 2020, respectively. He has more that five years of industry experience working as a Software Engineer and a Data Scientist in some of the leading software firms. His research interests include image processing, natural language processing, deep neural networks, and machine learning.

MUHAMMAD KAMRAN MALIK received the Ph.D. degree in computer science. He is currently an Associate Professor with the Department of Information Technology, Faculty of Computing and Information Technology, University of the Punjab, Lahore, Pakistan. He has more than 15 years of teaching and development experience. He has provided consultancy to many multinational firms on natural language processing, machine learning, and data science tasks. He is the author of one U.S. patent and 30 journals and conference papers. His research interests include natural language processing, machine learning, and data science.

KHURRAM SHAHZAD received the master's and Ph.D. degrees from the KTH-Royal Institute of Technology, Stockholm. He is currently a Tenured Associate Professor with the Department of Data Science, University of the Punjab. He has been associated with the Information Systems Groups, Eindhoven University of Technology, Eindhoven, and the University of Fribourg, Fribourg. He has published more than 50 papers in international conferences and journals.