



UrduDeepNet: offline handwritten Urdu character recognition using deep neural network

Faisal Mushtaq¹ · Muzafar Mehraj Misgar¹ · Munish Kumar² · Surinder Singh Khurana¹

Received: 3 June 2020 / Accepted: 21 May 2021 / Published online: 7 June 2021
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2021

Abstract

Handwritten Urdu character recognition system faces several challenges including the writer-dependent variations and non-availability of benchmark databases for cursive writing scripts. In this study, we propose a handwritten Urdu character dataset for *Nasta'liq* writing style covering isolated, positional characters as well as numerals. We also propose a convolutional neural network (CNN) architecture for the recognition of handwritten Urdu characters and numerals. CNN is a novel technique for image recognition that does not need explicit feature engineering and extraction and produces efficient results as compared to standard handcrafted feature extraction approaches. The proposed system was trained on a training dataset of 74, 285 samples and evaluated on a test dataset of 21, 223 samples and achieved a recognition rate of 98.82% for 133 classes, outperforming the results of all state-of-the-art systems for the Urdu language.

Keywords Handwritten Urdu character recognition · Urdu OCR · Convolutional neural network · Deep learning

1 Introduction

Urdu, the fifth most spoken language in the world catering to 4.7 percent of the world's population, is widely spoken in Pakistan where it is the national language and India where it is recognized as one of the 22 official languages [13]. Urdu language speakers reside in 20 different countries of the world like *India, Pakistan, Turkey, Saudi Arabia, Bangladesh, Afghanistan, Iran, Azerbaijan, Nepal* etc. [34]. The Urdu language is composed and influenced by different languages like *Arabic, Persian, Turkish* and *Hindi* languages. Urdu is a non-scripting language [20, 21] which is cursive in nature and has no concept of capital or small letters because it is highly influenced by Arabic and Persian Scripts [3, 4, 9] therefore, they share similarities at writing level. In the cursive script, individual characters are joined to form ligature, and ligatures are grouped to form words, such linguistics characteristic is called as a

compound character. Each Urdu ligature has two parts: *RASM* and *IJAM*. *RASM* is the main stroke without the associated diacritic and is also called as the main body, see Fig. 1c. *IJAM* are mandatory diacritics used to disambiguate the same *RASM* having different consonant behavior as can be seen in Fig. 1(d). A *RASM* can or cannot contain *IJAM* depending on consonantal behavior. The *RASM* of character *Reh* (ر) does not have any *IJAM*, whereas the *RASM* of character *Dhaal* (ذ) has an *IJAM* which is indicated in Fig. 1d. The Urdu character set has a total of 21 unique *RASMs* which are given in Fig. 2. Figure 2a specifies the name of *RASM*, Fig. 2b indicates the shape of *RASM* of Urdu character, and all characters with possible variations of diacritics of the same character class are shown in Fig. 2c [5]. In the modern Urdu language there are 38 basic characters [32] out of which 28 characters are adopted from the Arabic language [10]. Standard Urdu characters are shown in Fig. 1a.

Each character of the Urdu language has multiple forms or shapes (i.e., isolated, initial, medial, and final) depending upon its position within the word. For example, character *Beh* (ب), *Peh* (پ), *Teh* (ت), etc. can join with other characters from both directions i.e., from preceding and subsequent character as well in the same word, but some characters like *Alif* (ا), *Daal* (د), *Reh* (ر) only joins from

✉ Munish Kumar
munishcse@gmail.com

¹ Department of Computer Science and Technology, Central University of Punjab, Bathinda, Punjab, India

² Department of Computational Sciences, Maharaja Ranjit Singh Punjab Technical University, Bathinda, India

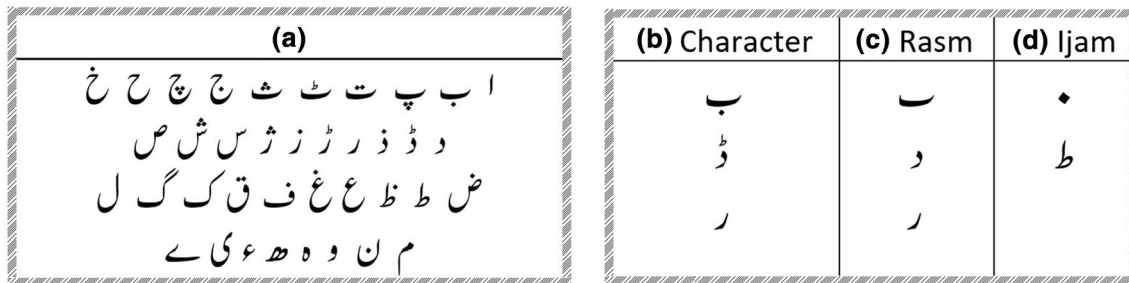


Fig. 1 a Urdu language basic character set, b Urdu character with Rasm and Ijam

RASM Class (a)	RASM Shape (b)	Characters with same RASM (c)	RASM Class (a)	RASM Shape (b)	Characters with same RASM (c)	RASM Class (a)	RASM Shape (b)	Characters with same RASM (c)
Alif Class	ا	ا	Toi Class	ط	ط ظ	Noon Class	ن	ن
Beh Class	ب	ب پ ت ٹ ث	Ayn Class	ع	ع غ	Waaw Class	و	و
Jeem Class	ج	ج چ ح خ	Fay Class	ف	ف	Hay Class	ہ	ہ
Daal Class	د	د ڈ ذ	Qaaf Class	ق	ق	Hay Do-Chashm Class	ھ	ھ
Reh Class	ر	ر ژ ز	Keef Class	ک	ک گ	Hamza Class	ء	ء
Seen Class	س	س ش	Laam Class	ل	ل	Yay Class	ی	ی
Suad Class	ص	ص ض	Meem Class	م	م	Yay-Bari Class	ے	ے

Fig. 2 Rasm classes of Urdu character set

right to left i.e., from preceding alphabet in a word only. Also, there exist some characters with no joining ability at all, for example, character *Hamza* (29) (ء). Therefore, based on the joining properties Urdu characters can be broadly divided into two categories: Joiners and non-joiners. The joiners acquire all the four shapes, depending on the neighboring character, while non-joiners can only acquire only isolated and final shape. Figures 3 and 4 show

all the possible shapes of a joiner and non-joiner Urdu character, respectively.

There are also two field characters in the Urdu language being excessively used in many Urdu ligatures but are not the members of the basic character set: *Alif-Madd* (آ) and *Noon-Ghunna* (ن). Also, like any other languages, such as, *English, Arabic, Persian*, etc. the Urdu language also

Final	Medial	Initial	Isolated	Final	Medial	Initial	Isolated	Final	Medial	Initial	Isolated
ق	ق	ق	ق	س	س	س	س	ب	ب	ب	ب
ک	ک	ک	ک	ش	ش	ش	ش	پ	پ	پ	پ
گ	گ	گ	گ	ص	ص	ص	ص	ت	ت	ت	ت
ل	ل	ل	ل	ض	ض	ض	ض	ٹ	ٹ	ٹ	ٹ
م	م	م	م	ط	ط	ط	ط	ث	ث	ث	ث
ن	ن	ن	ن	ظ	ظ	ظ	ظ	ج	ج	ج	ج
ھ	ھ	ھ	ھ	ع	ع	ع	ع	چ	چ	چ	چ
خ	خ	خ	خ	غ	غ	غ	غ	ح	ح	ح	ح
ی	ی	ی	ی	ف	ف	ف	ف	غ	غ	غ	غ

Fig. 3 Joiner Urdu characters

Final	Isolated	Final	Isolated	Final	Isolated	Final	Isolated	Final	Isolated
و	و	ز	ز	ر	ر	ذ	ذ	ا	ا
ے	ے	ژ	ژ	ڑ	ڑ	ڈ	ڈ	د	د

Fig. 4 Non-Joiner Urdu characters

contains ten numeral characters starting from 0 to 9 (۰, ۱, ..., ۹) which are shown in Fig. 5.

In this paper, we proposed a new and comprehensive handwritten dataset for *Nasta'liq* Urdu script named Handwritten Urdu Character dataset (HUCD) which consists of isolated and positional characters as well as numerals collected from 750 individuals of Kashmir valley. We also proposed a novel deep learning model based on CNN for recognition of the handwritten Urdu characters and numerals. The proposed analytical model achieved state-of-the-art recognition accuracy of 98.82% with a root-mean-square error rate of 4.24.

1.1 Optical character recognition

The Optical Character Recognition (OCR) is a classical pattern recognition problem which takes handwritten or printed text as an input and regenerates an editable machine-understandable format of text extracted from the scanned image. In other words, OCR is an electronic and mechanical conversion of an image into a textual form like ASCII or UNICODE. OCR finds its applications in translation tools, document automation, advanced document scanning, document verification, business applications, data entry, systems for visually challenged persons [29], data mining, biometrics, text storage optimization and electronic data searching, etc. [39]. Over the past recent years, there has been unending demand for Urdu-based OCR, not only to facilitate the native speakers to readily use it for their pocket device requirements, but also for digitizing historical documents, holy books, and poetry books to preserve the literary heritage. Various organizations and libraries throughout the world have digitized their collections of documents and have made them available online facilitating their retrieval. A major shortcoming with these document archives is that they are stored as digitized images that are neither searchable nor editable [10]. In addition, such digitized images require more storage space as compared to text [27] and more bandwidth for their transfer or retrieval over the internet. Conversion of these digitized images into text is therefore highly desirable.



Fig. 5 Urdu language numerals from 0 to 9

There are two types of OCR: *Offline* and *Online* OCR based on the input acquisition which can be printed, typewritten or handwritten [39]. Offline OCR system deals with the input image which is already written i.e., handwritten or machine-printed format. Whereas input to the Online OCR is directly written by the user using any smart device.

1.2 Peculiarities and challenges in Urdu language

Urdu OCR is a challenging task due to the following complexities of cursive scripts:

1.2.1 Ligature

In the Urdu language, two or more individual characters are combined to form ligatures (sub-word) and ligatures are grouped to form a word. Figure 6a, b shows two ligature based and single ligature based Urdu word, respectively. This complex combination of ligatures is a major challenge for OCR.

1.2.2 Diacritics

The characters of the Urdu language are not single but are enriched with some marks (i.e., *Chota Toi*, *Nuqta (dots)*, *Hamza*, *Madaa*, *Shud*, *Paish*, *Zabr*, *Zair*, *Juzm* and *Khari Zab*) below or above the primary character as represented in Fig. 7. These marks are called secondary characters or *diacritics*. Recognition of these *diacritics* produces problems in segmentation and then in recognition stages [2, [24].

1.2.3 Script

The Urdu language has 12 different writing styles like *Naskh*, *Nasta'liq*, *Kofi*, *Riqa*, *Aswad*, *Taleeq*, *Batool* etc., but two styles *Naskh* and *Nasta'liq* are the most commonly used. *Nasta'liq* style is mostly followed by printed media like magazines, newspapers, whereas online material is generally available in the *Naskh* style of writing. Both of these styles are written in a semi-cursive fashion from right to left, similar to *Arabic* script.



Fig. 6 Example of Urdu words. **a** Separate letters of “Bandagi,” **b** Separate letters of “Tasbih”

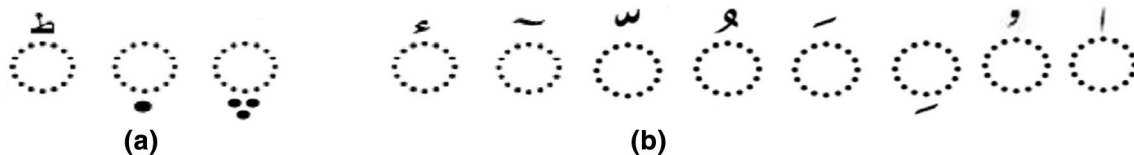


Fig. 7 Urdu diacritics. **a** Common: *Chota Toi, Dots*, **b** Uncommon: *Hamza, Madaa, Shud, Paish, Zab, Zair, Juzm and Khari Zab*

1.2.4 Diagonality and baselines

A prominent distinction between the *Naskh* and *Nasta’liq* styles is the flow in which these scripts are written. *Naskh* has a horizontal writing flow from right to left with a flat baseline, i.e., words are spread horizontally along the baseline taking more space, while *Nasta’liqs* flow is diagonal from right-top to left-bottom with multiple baselines. Although diagonality economizes the horizontal space, the however vertical overlap of characters is possible with the words of the underneath line when a word in a given line contains a large number of letters or vice versa (see Fig. 8). The *Nasta’liq* style was evolved from two other writing styles namely *Naskh* and *Taleeq*. The *Nasta’liq* style of writing is considered as one of the sophisticated scripts to be used digitally [36]. Figure 9a shows the style, diagonal flow, and multiple baselines for *Nasta’liq* script, whereas Fig. 9b represents the style, horizontal flow, and single baseline for *Naskh* script.

1.2.5 Direction of writing

The Urdu language is bidirectional in nature. Generally, the right to left direction is used for reading and writing the Urdu text, whereas the left to the right direction is used for reading and writing Urdu numerals. The bidirectional behavior of cursive scripts is illustrated in Fig. 10a.



Fig. 8 Red circles and green boxes represent inter-word and inter-ligature variable spacing; sky blue circles show vertical overlapping

1.2.6 Overlapping

Words and characters in *Nasta’liq* style are written from the right diagonal at different angles with the baselines for beauty and sufficient space (called *kerning*). On one hand, it creates beauty in writing style but on the other hand, it creates vertical overlapping problems of characters. The *Naskh* style can be easily segmented due to its non-overlapping of characters, linearity in writing, and single baseline. On the other hand, *Nasta’liq* style of writing is highly cursive with multiple baselines and therefore, making the segmentation process a difficult task. Figure 9c shows *intra-ligature* and *inter-ligature* overlapping of characters in *Nasta’liq* style of writing [25, 36].

1.2.7 Stacking

The diagonality of the *Nasta’liq* style results in variable height and variable width ligatures as shown in Fig. 10b. Since there is no limit on a number of characters to be joined in a ligature, thus giving rise to stacking problems (multiple baselines), the stacking of characters makes some ligatures higher than the usual ones, thus, consuming less horizontal space as shown in Fig. 9a [25].

1.2.8 Context sensitivity/graphism multiplication

Each character in a ligature has a different shape depending on the position of the character and the neighboring character to which it is connected. The character may occur at initial, medial, final, and isolated positions. Due to the context sensitivity, the Urdu language is completely different from other languages. Figure 11 shows some sample alphabets and their shapes with respect to their position in a word.

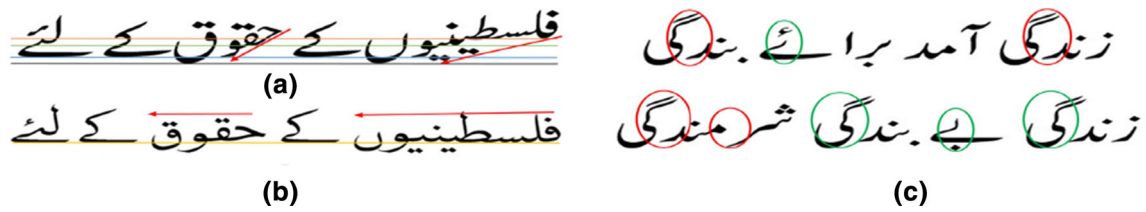


Fig. 9 **a** *Nasta'liq* style of writing, diagonal flow, and multiple baselines, **b** *Naskh* style of writing, horizontal flow and single baseline, (c) overlapping: Red ovals represent inter-ligature and green ovals represent intra-ligature overlapping

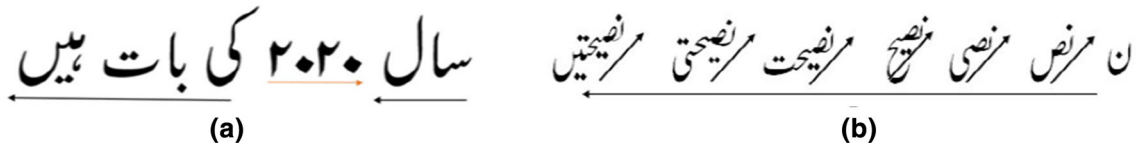


Fig. 10 **a** Bi-directional nature of Urdu language, **b** dimensions of ligature increase with stacking

Final	Medial	Initial	Isolated	Alphabets
صا بن	بنا وں	نا زک	ایمان	ن (Noon)
وہم	سمرات	مراد	آم	م (Meem)
برما	ملاح	امرو د	شعرا	ا (Alif)

Fig. 11 Different shapes of alphabets with respect to their position in a Word

1.2.9 Dotting problem

The diacritics ‘*Nuqta (dots)*’ and ‘*Chota Toi*’ are a requisite part of many Urdu characters. The position and number of ‘*Nuqtas*’ associated with a character are vital for differentiating the characters from the same class. Therefore, in Urdu script, the characters are differentiated from each other using *dots*. All the diacritics except ‘*Nuqta*’ and ‘*Chota Toi*’ helps in pronouncing the Urdu words and are called ‘*Aerab*’ [2, 24]. Figure 7a and b shows the commonly and rarely used Urdu diacritics.

1.2.9.1 Non-monotonicity Cursive scripts are non-monotonic in nature meaning that the stroke of some characters frequently goes back beyond the previously written character. The non-monotonic nature of the cursive script is illustrated in Fig. 12a where ‘*Bari-Yay*’ goes backward beyond the previous character ‘*Beh*’.

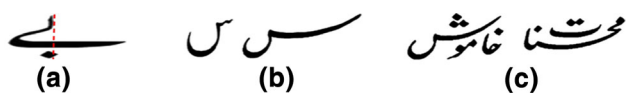


Fig. 12 **a** Non-monotonicity, **b** stretching, **c** positioning in Urdu Script

1.2.9.2 Stretching and positioning In *Nasta'liq*, some characters frequently change their standard version to a longer version, if space is to be occupied which causes certain characters to change their default shape, while others change their width only. The characters *Seen*, *Sheen*, *Beh*, *Keef*, and *Gaaf* etc. often exhibit such phenomenon. Figure 12b shows such a phenomenon for the character ‘*Seen*’. On the other hand, *positioning* refers to a situation when a character or ligature is placed on top of the previous ligature of the same word or an adjacent word in order to economize the space. Figure 12c illustrates such a phenomenon.

1.2.9.3 Spacing In the cursive script, variable spacing may occur between two words or between the ligatures just to justify the text in Urdu script. Figure 8 shows various *inter-word* and *inter-ligature* spacing in Urdu text as represented by red circles and green boxes, respectively.

1.3 Artificial neural network

Artificial neural network (ANN) is a machine learning (ML) approach comprised of many interconnected, simple functional units called *neurons* organized in input, hidden, and output layers that act together as parallel information-

processors, to solve regression or classification problems. The ANN is inspired by the structure of the brain and resembles the biological nervous system. Therefore, if the network is created by stacking layers of these multiple connected neurons the resulting computational system can separate the input space (that receives the input information) into a discrete number of classes (that relay the processed information out called output layer) or they can approximate the function (black-box) that maps inputs to outputs (that pass the information back and forth between layers for processing by invoking certain goals and learning rules called hidden layers). The units in each layer are interconnected to the nodes or units in surrounding layers with each connection having a weight value. Each input is then multiplied by a corresponding weight value and then summed up. This computation is the dot product between the input vector and the weight vector which can be thought of as a measure of similarity between the two. The summed value then undergoes a transformation based on activation function, which can be *sigmoid*, *hyperbolic (tanh)* or *rectified linear unit (ReLU)* function. These functions are used because they make it easier to compute the partial derivative of the error delta with respect to individual weights. Both *sigmoid* and *tanh* functions implement nonlinearity before and after the respective thresholds as output plateaus and normalize the input into a narrow output range i.e., 0/1 and $1/-1$, respectively. On the other hand, *ReLU* exhibits both saturating and non-saturating behaviors. Finally, the output of the activation function is then fed as input to the subsequent unit into the next layer. The result of the final output layer is used as the solution for the problem. The ANN can be used to solve various day today problems including pattern recognition, classification, computer vision, dimensionality reduction, regression, and natural language processing, etc. [38].

1.3.1 Deep learning

Deep Learning (DL) or hierarchical learning is a type of ANN based on a set of algorithms that attempt to model high-level representations or abstractions from datasets without any manual design of feature extractors. Deep neural network consists of several nonlinear processing layers with complex structures in between the input and output layer for feature learning and pattern classification. Deep Learning can be *Supervised*, *Unsupervised*, and *Semi-Supervised*. There exists another category of learning approach called *Reinforcement Learning (RL)* or *Deep RL (DRL)* which is often discussed under the scope of semi-supervised or unsupervised learning approaches. In the case of supervised DL, the environment has a set of inputs and the corresponding set of outputs. Therefore, in supervised learning, there is an output label associated with

every input data. In other words, supervised learning uses the labeled data to train the network. Different supervised DL learning approaches include deep neural network (DNN), convolutional neural network (CNN), and recurrent neural network (RNN) including long short-term memory (LSTM) and Gated Recurrent Units (GRU). In unsupervised DL, the algorithms are left to themselves to discover interesting structures or patterns in the data. Therefore, there are no output labels associated with the given input data. Clustering, dimensionality reduction and generative techniques like Auto-Encoders (AE), Restricted Boltzmann Machines (RBM) and Generative Adversarial Networks (GAN) are considered as unsupervised learning approaches. In addition, RNNs (LSTM) and RL are used for unsupervised learning in various application domains. Semi-supervised learning is a learning approach that occurs on partially labeled datasets, while DRL is a learning technique in which the model is exposed to an environment where it continuously trains itself using the trial and error method [7].

1.3.1.1 Convolutional neural network CNN is a type of deep learning based on the organization of the animal visual cortex for processing data of grid pattern, such as images. CNN is the neural network of choice for computer vision (image recognition) and is designed to automatically and adaptively learn spatial hierarchies of features (low to high-level patterns) through a backpropagation algorithm [38, 40]. CNN is a mathematical construct that is composed of a series of convolution and pooling layers followed by a fully connected layer and a normalizing (softmax) layer. Each node of the convolution layer extracts features from the input images by performing the convolution operations on input nodes and preserves the spatial relationship between pixels by learning image features using matrices of input data. A fully connected layer performs classification by mapping the extracted features into the final output as class probabilities prediction. Subsampling or pooling layer performs the down-sampling operation on input maps. The down-sampling operation reduces the size of each dimension of output maps depending on the size of down-sampling mask.

Convolution layer: A CNN implicitly breaks down an image in terms of spatial properties like edges, strokes, contours, textures, gradients, orientation, and color and learns them as representations in different layers. A convolution layer typically consists of a combination of linear (convolution) and nonlinear (activation function) operations to perform feature extraction. In a Convolution operation, a small array of numbers, called a kernel is applied on an input array of numbers, called a tensor by performing the summation of the element-wise product of each kernel element and input tensor to obtain the output

value in the corresponding position of output tensor, called as feature map (see Fig. 13). The process is repeated on multiple kernels to generate an arbitrary number of feature maps, each representing different characteristics of the input tensor. Thus, different kernels can be considered as different feature extractors of the convolutional layer. Therefore, the size and number of kernels (filter size) define the two main hyper-parameters of the convolution layer. The size of the kernel is a tuple of 3×3 or sometimes 5×5 or 7×7 , whereas the number of kernels is arbitrary and determines the depth of output feature maps.

There are two other parameters in convolution operation i.e., *padding* and *strides*: *Padding* can be ‘*same*’ or ‘*valid*’. Setting *Padding* to ‘*valid*’ means that the spatial dimensions of the output feature map are allowed to reduce compared to the input tensor via the natural application of convolution because it does not allow the center of each kernel to overlap the outermost element of the input tensor as described in Fig. 13. With ‘*valid*’ padding each successive feature map would get smaller after the convolution operation. To preserve spatial dimensions of the output feature map with input tensor, ‘*same*’ padding is used. This is achieved by adding rows and columns of zeros on each side of the input tensor, so as to fit the center of a kernel on the outermost element of the input tensor and keep the same in-plane dimension through the convolution operation (see Fig. 14). The distance between two successive kernel positions is called a *stride*. In other words, *stride* denotes the number of pixels by which the kernel window moves after each convolution operation (see Fig. 15). The typical choice of a stride is 1. However, in order to achieve down-sampling of the feature maps a stride larger than 1 or alternatively pooling operation can be used.

Thus, with regard to the convolution layer training, a CNN model is to identify the kernels that are effective for a given task based on a given training dataset. During the training process kernels are the only parameters automatically learned in the convolution layer, whereas the hyper-parameters such as number of *kernels*, *size of the kernels*,

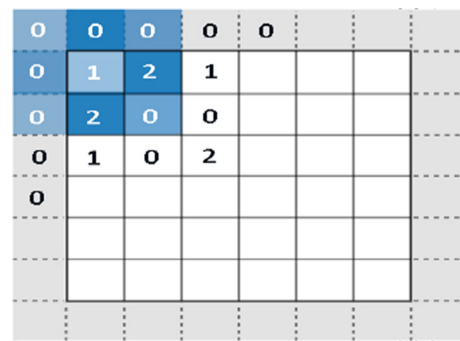


Fig. 14 Example of zero padding



Fig. 15 Movement of kernel window for Stride of 1

padding, and *stride* are needed to be set before the training process starts.

Nonlinear activation function: ReLU: Once the convolved outputs are obtained they undergo a nonlinear transformation by passing through a nonlinear activation function. The only purpose of the activation function is to introduce nonlinearity in the network. Different types of nonlinear activation functions include *sigmoid*, *tanh* and *rectified linear unit (ReLU)*. *ReLU* is less computationally expensive as compared *tanh* and *sigmoid* because it involves simpler mathematical operations and only a few neurons are activated at a time making the network sparse efficient and easy for computation. In simple words, *ReLU* learns much faster than *tanh* and *sigmoid* function and simply computes the function: $f(x) = \max(0, x)$ (see Fig. 16).

In other words, the activation function decides, whether a neuron should be excited by calculating a weighted sum and further adding bias with it. A neural network generally updates the weights and biases of the neurons on the basis of the output error. This process is called as *backpropagation*. The backpropagation is only possible by activation

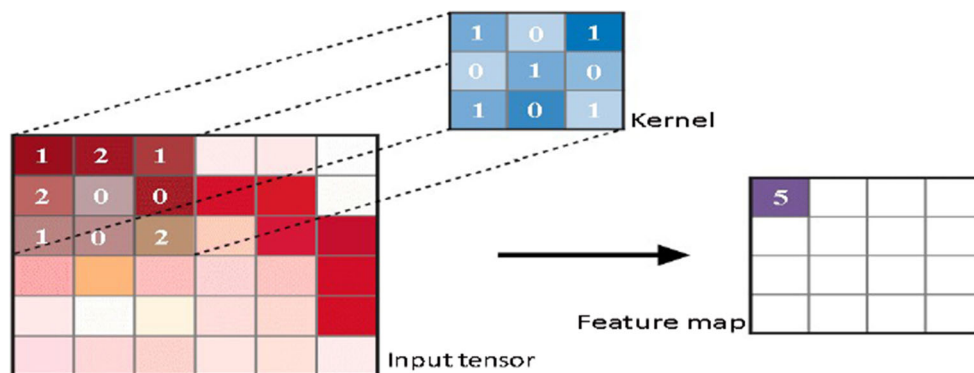


Fig. 13 An example of convolution operation with a kernel size of 3×3

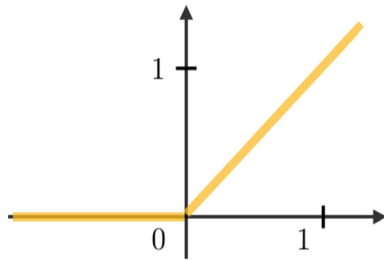


Fig. 16 ReLU activation function

functions since the gradients are supplied along with the error to update the weights and biases.

Pooling Layer: A pooling layer is a simple down-sampling operation performed in order to introduce a translation invariance to small shifts and distortions by reducing the size of each dimension of output feature maps, which in turn decreases the number of subsequent learnable parameters. Like convolution operation, *pool size*, *stride*, and *padding* are the hyper-parameters in pooling operations. However, there is no learnable parameter in any of the pooling layers.

- *Max pooling and average pooling*

In max pooling, each pooling operation extracts patches from the input feature maps and then selects the maximum value of the current patch, whereas in average pooling, the average of all the elements of the current patch is taken (see Fig. 17). A max pooling or average pooling operation down-samples of the in-plane dimension (height \times width) of feature maps by a factor of 2. However, the depth dimension of feature maps remains unchanged.

- *Global max pooling and global average pooling*

A global max pooling or global average pooling operation down-samples the in-plane dimension (height \times width) of a feature map into a 1D array by selecting the maximum value or by taking the average of all the elements in each feature map, respectively, whereas the depth of feature maps is retained. These types of pooling

operations reduce the number of learnable parameters, enable the CNN to accept variable size input, and are only applied once before the fully connected layers.

Fully Connected layer: The fully connected layer (FC) operates on a flattened input as shown in Fig. 18, where each input feature map is mapped by a subset of fully connected layers to the final outputs of the network, as the predicted probabilities of each class for final classification. In other words, the output feature maps of the final convolutional or pooling layer are flattened and connected to one or more FC layers, also called as *dense layers* in which every input is connected to every output by a learnable weight. Flattening is the process of transforming the multi-dimensional feature maps into a one-dimensional array of numbers. The final dense layer typically has the same number of output nodes as the number of classes. Each fully connected layer is followed by a nonlinear function.

Last layer activation function (softmax): A softmax function can be seen as a generalized logistic function that takes as input a vector of scores and produces as output a vector of output probabilities. In other words, a softmax function is an activation function for performing multiclass classification which normalizes the output of the last dense layer to target class probabilities, where each value ranges between 0 and 1 and all values sum to 1. For a given class, the softmax function can be computed as:

$$f(x_i) = \frac{e^{x_i}}{\sum_{k=1}^N e^{x_k}}$$

where x_i is the output CNN Score for each class i .

2 Related work

Recognition techniques for cursive scripts are generally grouped into two major classes, segmentation-based and segmentation-free methods.

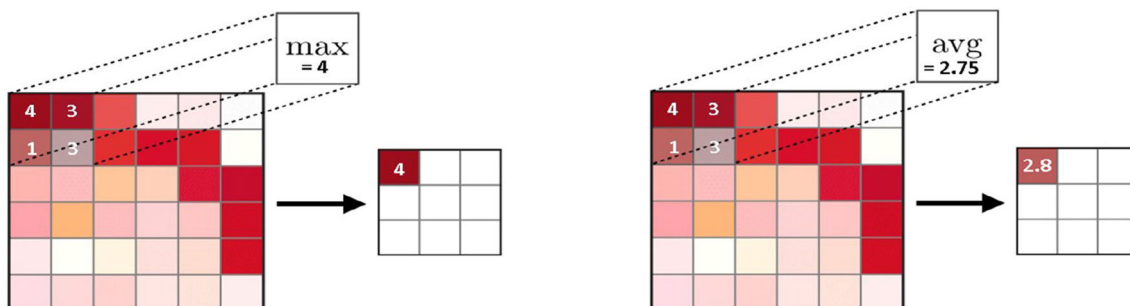


Fig. 17 Example of max and average pooling

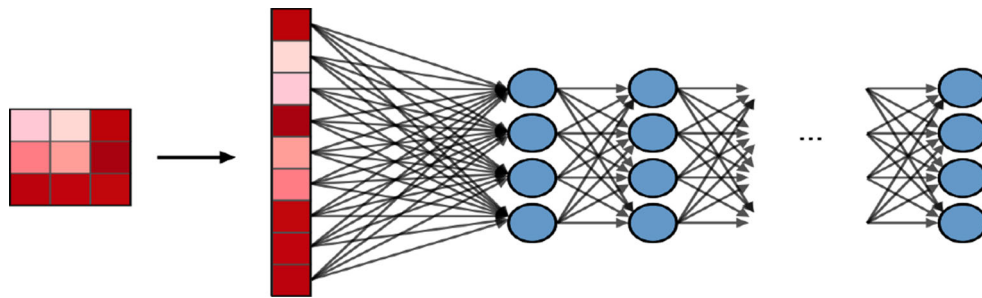


Fig. 18 CNN's fully connected layer

2.1 Segmentation-based approaches

Segmentation-based (or analytical) methods employ individual characters as recognition units which are segmented either implicitly [31] or explicitly [2]. Segmentation of Urdu text into characters is itself a very challenging task because of various complexities inherited to cursive scripts.

In implicit segmentation-based techniques, the learning algorithm is provided with the complete transcription of text lines as an input to determine and learn the character boundaries and shapes by itself. Among well-known analytical methods, an implicit segmentation-based Urdu character recognition technique for *Nasta'liq* writing style was presented by Akram and Hussain [5], based on recognition of characters and joiners using Hidden Markov Models (HMMs). At first, the artistic features of characters and joiners were extracted using robust stroke-based traversal of Urdu ligatures in order to analyze their shapes, train, and recognize the sequence of characters and their joiners. The system was tested on standard Urdu Printed Text Images (UPTI dataset) Dataset-1 comprising of 3, 309, 762 ligature instances with 1446 ligature classes and 91, 129 unique Urdu words, based on the recognition of character Unicode using the recognized labels of character core shapes, giving the recognition accuracy of 95.58%. In addition, the system was also tested on a UPTI dataset called Dataset-2 comprising of 1600 text lines, giving character recognition accuracy of 98.37%.

In explicit segmentation-based techniques, either ligature is divided into characters or isolated characters are considered as recognition units. Among the significant works, Rizvi et al. [34] proposed a supervised learning-based OCR system for the *Nasta'liq* Urdu language. The proposed system apprehends 98.4% accuracy using Random Forest and Logistics Algorithm. Other supervised learning algorithms like Sequential Minimal Optimization (SMO), Multilayer Perception resulted in the accuracy of 94.57% and 97.67%, respectively. The proposed technique also provides 89.92% recognition accuracy using Naïve Bayes Algorithm. Similarly, Naz, Ahmed, Ahmad, and Razzak [28, 30] proposed an OCR system based on zoning

features and 2-dimensional long short-term memory networks (2DLSTM) for classification of *Nasta'liq* Urdu text. The proposed system considers lines of text for classification as they represent significant information with low complexity and high speed. The system was evaluated on a standard UPTI database and achieved a character recognition accuracy of 93.39%. In another study, [19] proposed an approach for the development and experimental analysis of the Urdu OCR system. First a medium-sized database of 441 handwritten and machine-written Urdu characters were created, then features were extracted with three different methods like Hu moments, Zernike moments, and Principal Component Analysis (PCA) and finally Decision tree algorithm J-48 was used for classification. Overall recognition accuracy of 92.06% was achieved using the Hu moments. Kaushal, Khan, and Varma [18] proposed a novel and robust technique for handwritten Urdu numerals and digits recognition based on Zernike Invariants and SVM as a classifier. This hybrid approach is used for feature extraction and is independent of basic information and other variations in handwritten numerals and overall 22 features corresponding to each numeral proceed for classification using Support Vector Machine (SVM) classifier, and the accuracy of 96.29% was achieved by the proposed technique. Pal and Sarkar [32] suggested an individual character recognition-based system using a combination of topological, contour, and water reservoir concept-based features. The system was evaluated on the Small Variety Character dataset and achieved the recognition accuracy of 97.8% on average for *Nasta'liq* Urdu characters. It also identified individual text lines with an accuracy of 98.3%. Pal and Sarkar [32] used a horizontal and vertical projection profile, component labeling method for character segmentation and achieved the character segmentation accuracy of 96.9%.

2.2 Segmentation-free methods

Segmentation-free (or holistic) methods consider partial words or ligatures as a unit of recognition rather than individual characters. Among significant holistic

techniques, N. H. Khan, Adnan, and Basar [20, 21] proposed a recognition system based on semi-supervised multi-level agglomerative hierarchical clustering for the classification of the ligatures. The classification and recognition were performed using four machine learning techniques i.e., K-Nearest Neighbor (K-NN), Naïve Bayes, Decision Trees and Linear Discriminant Analysis and achieved the maximum accuracy of 100% for K-NN. They also analyzed geometrical features which are responsible for most misclassifications and applied noise removal pre-processing to improve the overall accuracy. In another study, Din, Siddiqi, Khalid, and Azam [11] presented a holistic OCR system for printed Urdu *Nasta'liq* font based on statistical features and employing HMMs for classification. The system was trained on 1,525 unique high-frequency Urdu ligature clusters among which 16 represent dots and diacritics and evaluated on random 6,187 ligatures from a standard UPTI database containing 7,063 text lines. The extracted ligatures were first split into 9,778 primary and secondary ligatures and achieved the recognition rate of 95.24% and 93.30%, respectively, and after association by sequential clustering algorithm the recognition rate decreases to 92.26% on complete ligatures (partial words). In another study, Ahmed, Iqbal, Mehmood, and Ayub [4] proposed a correlation-based framework for the *Nasta'liq* Urdu, where each ligature was categorized according to segmented last character by finding statistical similarity correlation with corresponding one-, two-, and three-character ligatures in sequence. The system was evaluated on a ligature image bank comprising of 3500 images and an accuracy of 97.4%, 82.3%, and 80.6% was achieved for isolated characters two- and three-character ligatures, respectively. They also found that the accuracy was affected after diacritic association and by similar shaped characters at smaller fonts such as ۛ, ۛ, and ۛ. Kadhm and Abdul [17] proposed an architecture based on a Support Vector Machine (SVM) classifier for handwritten word recognition depending on the handwriting word level. The system was trained and tested on an Arabic handwritten dataset (AHDB) comprising of 2913 handwriting word images and achieved the recognition accuracy of 96.317% based on Discrete Cosine Transform (DCT) and Histogram of Oriented Gradient (HOG) feature extraction methods and SVM classifier with the polynomial kernel. They also found that the polynomial kernel is convergent and more accurate than linear or Radial Basis Function (RBF) SVM kernel. Sobia T Javed et al. [16] suggested a holistic OCR model for *Nasta'liq* Urdu script in which global transformational features were extracted from non-segmented ligatures and were given as input to Hidden Markov Model (HMM) recognizer because of ease and efficiency in identification. The process of HMM was divided into two tasks of training (carried out on 1500 high-frequency

ligatures which vary from one character to seven characters in ligature) and recognition, which resulted in text recognition accuracy of 92%.

2.3 Deep learning-based approaches

Deep learning is a state-of-the-art machine learning algorithm but it has not been extensively applied for recognition of cursive like scripts such as Urdu until recently with advances in technology, computing power, and computing resources. Among the notable implementation of such algorithms Latif, Alghazo, Alzubaidi, Naseer, and Alghazo [23] proposed a deep CNN architecture with two hidden layers for the recognition of Multilanguage handwritten numerals. The system was evaluated on datasets of 5 different language numerals i.e., Eastern Arabic, Persian, Devanagari, Urdu and Western Arabic, and achieved the average recognition accuracy of 99.26% with a precision of 99.29% for the combined Multilanguage database and 99.322% for each individual language. They also found that the proposed system produced better results on datasets having similar geometrical features or have combined different geometrical features. Similarly, I. Ahmad et al. [1] proposed a novel case of recurrent neural network, i.e., gated bidirectional LSTM (GBLSTM) based on ligature information dissimilar to character-based LSTM strategies. The proposed single layer GBLSTM system was trained on un-degraded and evaluated on unseen artificially degraded samples of UPTI database which recognized 3604 ligature classes and 191 character classes with an accuracy of 96.71% and 86.4%, respectively. They used character raw pixels as features instead of traditional human-crafted features because of the latter being more error prone. In another study, N. Javed, Shabbir, Siddiqi, and Khurshid [14] adopted the convolutional neural network-based system for recognition of Urdu ligatures. The system was evaluated on 18,000 Urdu ligatures with 98 different classes, realized a recognition rate of 95%. The input to CNN was fixed sized ligature images which automatically extract features from raw pixels. CNN was found to be an effective feature extractor as compared to conventional human-crafted feature extraction methods. They also investigated different CNN architectures and realized that the smaller the kernel size and the deeper the network, the better are the recognition results. Similarly, Naz et al. [31] proposed a novel technique for Urdu OCR system based on CNN and RNN for feature extraction, learning, and classification of cursive *Nasta'liq* Urdu script. At first, the low-level translational invariant features were extracted using CNNs and are then forwarded to multi-dimensional long short-term memory (MDLSTM) for contextual feature extraction and learning. The proposed system when evaluated on the publicly available UPTI dataset, a recognition

rate of up to 98.12% for 44-classes was achieved. In another study, Naz, Umar, et al. [28, 30] suggested a segmentation-based technique for the *Nasta'liq* script based on multidimensional recurrent neural networks by extracting statistical features of characters and feed them to a MDLSTM recurrent neural network (MDLSTM RNN). The proposed technique was carried out on the standard UPTI database comprising of 10, 000 *Nasta'liq* font text lines and evaluation of the proposed system resulted in a promising character recognition rate accuracy of 96.40%.

2.4 Study on various challenges in Urdu script

Having similarly shaped and a large set of character classes makes the case of Urdu like cursive script challenging and more complex. The issues related to the Urdu language require more considerable efforts for the *Nasta'liq* style of writing as compared to *Naskh* because it poses even greater challenges for the development of the Urdu OCR system as compared to the later one. Such numerous challenges in Urdu and Urdu like script languages are explained by Naz et al. [29] with the emphasis being on the peculiar nature of *Nasta'liq* script and explained the detailed overview of existing OCR Systems and also suggested that there is a need to develop a model which overcomes the challenges inherited to Urdu like script languages. Further, Naz et al. [29] also proposed a multilingual character recognition system based on a ghost character theory, which states on the basis of a proponent of this theory that there are some problems in Urdu ASCII code plate. Similarly, Raza, Habib, Ashraf, and Javed [33] presented a comprehensive review of various parsing techniques used in natural language processing to extract syntactic features of a sentence. Further Raza et al. [33] contributed past, present, and future of parsing techniques for the Urdu language. In a similar study, Daud et al. [9] contributed a broad survey regarding modern techniques and different linguistic resources available for Urdu language processing. Daud et al. [9] also discussed available datasets, characteristics, resource sharing between Hindi and Urdu language, orthography and morphology of Urdu language, pre-processing aspects like Diacritics removal, stop words removal, normalization, stemming etc. in detail. The overall goal of their survey was to organize the past and present of Urdu language processing (ULP) so that it can provide a vital platform to ULP researchers in the future using statistical learning.

2.5 Pre-processing approaches

For improving the performance of an OCR system, the pre-processing step plays an important role in the development of an OCR engine. Pre-processing the input data directly

affects the readability and efficiency in the rest of the steps which may involve binarization, noise filtering, smoothing, segmentation, skew correction, and thinning (skeletonization) [6, 12]. Among the preliminary works, Nautiyal, Singh, and Rana [26] presented a supervised single layer perceptron learning algorithm which was used to detect noisy characters in the given texts, the main aim of the study was to improve the accuracy and performance of the recognition system, but he suggested his approach using English language characters and the algorithm recognizes 85% of the characters in a very noisy environment having 87.66% noise. Similarly, Sobia Tariq Javed, Fasihi, Khan, and Ashraf [16] proposed a model to remove punch-hole and background noise from handwritten Urdu text images. For background noise removal, the difference between the meaningful foreground text and background noise was exploited using median color intensities (threshold). For punch-hole noise removal, iterators across the outer regions were launched which look for a series of dark pixels. The dark pixels were brushed off with the median color value based on the local median color intensities. The proposed system was able to remove 93% background and 96% punch-hole noise. Kadhm and Abdul [17] used Fuzzy C-Means clustering (FCM) for the removal of unwanted space (noise) by defining the bounding box around the Urdu ligatures. They also performed image thinning for removing the redundant pixels around the edges of text and finally normalized the images to fix size for fast image recognition. In another study, K. Khan et al. [19] used filtering and morphological operations for noise removal, global and adaptive thresholding for binarization, the correlation for skew correction, thinning for removal of redundant pixels, and normalization for fixed sized images.

2.6 Knowledge gap and motivation

Although OCR, a computational intensive field, has witnessed a significant improvement over the years mainly due to the advancement in the computational intelligence algorithms. The OCR for Urdu like scripts is still in its infancy stage due to its complex cursive nature. This massive lag of research is mainly due to segmentation errors because of various complexities associated with its cursive script as well as the inefficiencies in different fields, such as benchmark datasets, dictionaries, research funding's, equipment's and other necessary utilities. The development of OCR for Urdu script has remained a challenging task for Urdu researchers for last few years due to the complexity of *Nasta'liq* Urdu writing style (i.e., dots, diacritics, diagonal writing style etc.). In addition, the literature review also showed that the accuracy of Urdu OCR system directly depends upon the quality of the input

images. The main difficulties encountered in different document images were as follows:

- Writer-dependent style variations in shape.
- Distortions, caused by broken, connected and smudged characters, ink spread and speckle.
- Space variations, due to skew and variable spacing.

These imperfections introduce numerous problems in different parts of the recognition process of an OCR system, resulting in rejections or misclassifications. The majority of errors were often due to the acquisition and segmentation process. Errors in the acquisition process were the result of noise (dust) inserted during scanning which resulted in joining characters or association of noise as dots with the character core shapes. Errors in the segmentation process were the result of confusion between text and noise, space insertion or deletion, compound words and reduplication of words. Even if the character was written or printed, scanned and segmented properly, it was incorrectly classified. This happened mainly due to similar core shapes of different characters (such as ا , و , and ل , و) or confusing shapes of different joiners or the selected features were not sufficiently efficient in separating the different classes, or the features that were difficult to extract were computed incorrectly. Finally, errors were also introduced by the post-processing due to the incorrect association of secondary characters (diacritics) with the primary characters to reconstruct the original words or ligatures. These problems occurred if the text is skewed, or due to proportional spacing. Also, it has been observed that with the increase in a number of characters per ligature the recognition accuracy decreased. In addition, the existing models do not provide significant practical usability due to the following reasons:

- Due to the poor design of classifiers where the classifier was trained and tested on smaller datasets, however, larger datasets are needed to include all the possible forms of each character as well as to incorporate Special characters in order to develop a complete Urdu OCR engine.
- The existing OCR models were mostly tested on trained ligature level datasets and not for recognition of text in real documents which require efficient text word segmentation.
- Most of the existing systems have not included the Aerabs (Zair, Zabar, and Paish) in the OCR system, which plays an important role in correct character recognition and preserving the meaning of Urdu words.

The main motivation of character recognition is to imitate the human reading ability, with human accuracy but

at a far higher speed. Therefore, the main aim of this study is to solve some of these imperfections by proposing a new Handwritten Urdu Character dataset (HUCD) and exploring advanced computational algorithms mainly for feature extraction and recognition purposes.

3 Proposed methodology

This section details the methodology proposed for recognition of isolated Handwritten Urdu characters, numerals as well as the positional characters of each individual character written in a different context (i.e., initial, medial, and final) using a deep neural network-based approach. The technique relies on an explicit segmentation-based approach where features are extracted from individual characters using convolutional neural networks. The overall structure followed to reach the final solution is shown in Fig. 19a.

3.1 Dataset collection

In order to perform the evaluation and development of Handwritten Urdu OCR, a self-collected dataset named Handwritten Urdu Character dataset (HUCD) containing a total of 106, 120 samples of Handwritten Urdu is used in the proposed CNN algorithm. In the Urdu language, there are a total of 38 basic characters, 10 numerals and two field characters and many special characters in Urdu language, out of 38 basic characters, 27 characters are joiners, 10 characters are non-joiners and 1 character has no joining property at all. Among two field characters, one character is a non-joiner and the other has no joining property. Similarly, 10 numerals do not have any joining property. Thus, making a total of 142 unique forms as shown in Fig. 20. With regard to this, the dataset was collected on an A4-sized paper, printed with rectangular boxes in landscape mode where each box corresponds to the unique Urdu character as shown in Fig. 21. Each page contained a total of 142 rectangular boxes to be used for writing 132

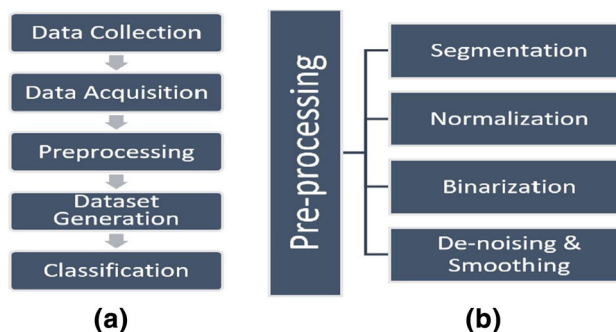


Fig. 19 a Flow of work to be followed, b pre-processing steps

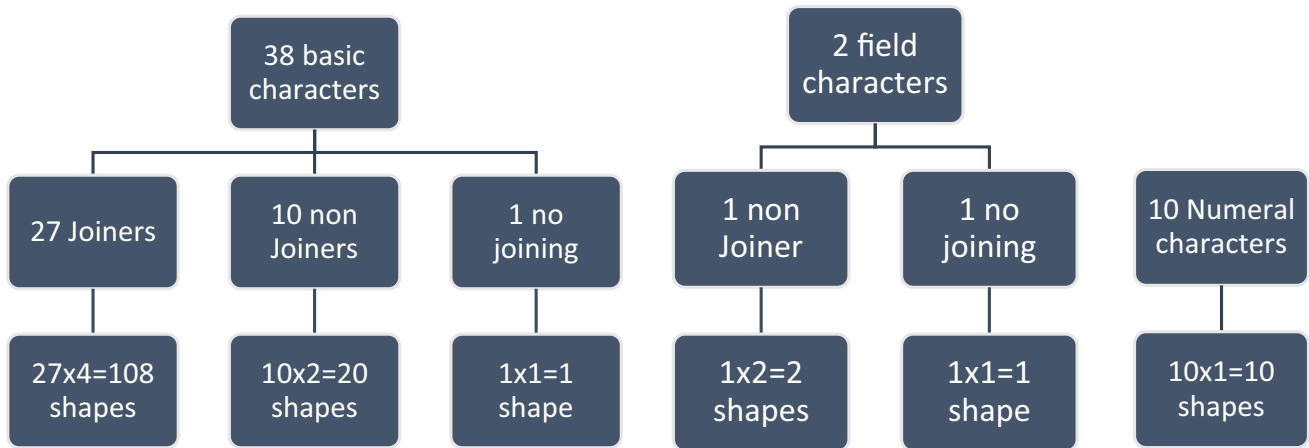


Fig. 20 Division of characters into joiners and non-joiners



Fig. 21 Scanned image of collected data

different individual and positional characters and 10 numeral characters. The dataset was acquired in different parts of Kashmir valley from 750 native writers with different age groups and different educational qualifications. The dataset documents were written by both male and female. Each individual writer was trained before gathering the dataset and was asked to write the provided Urdu characters within the respective box in neat and natural way while maintaining the appropriate number and placement of dots so that the character would not overlap with the bounding lines of the rectangle. Each writer was asked to write Urdu characters in an unconstrained environment in his or her natural handwriting with different pen, style, and inks. It must be noted that no special characters, ‘Aerabs’ or second form of ‘Hay final’ (i.e., ‘^ٲ’) character has been considered in this study.

3.2 Data acquisition

After dataset collection, paper-based handwritten documents are converted into electronic format by using a document scanner with 600 dpi for manipulation by digital computers. The document scanner scans a document and produces an electronic representation of the same in an image file format (.png in our case). Such an approach is basically termed as *offline* recognition where source images are obtained by scanning documents (handwritten, printed or typewritten,) or by using a digital camera for capturing a photograph. The digital representation of one such collected document is shown in Fig. 21.

3.3 Pre-processing

Pre-processing is an important phase in automatic handwritten character recognition which involves various operations that are carried out on the scanned input image in the sequence to make it effective for later recognition phases and to improve the overall performance [8, 29]. If the process of image acquisition is not carried out properly several issues like distortion, deformations, quality breakdown, orientation, and skewness may arise, which may introduce several difficulties in character recognition. Therefore, the phase of pre-processing is essential and plays a vital role in the development of an effective OCR engine. There are various pre-processing techniques to remove such image acquisition anomalies and the selection of such techniques depends on the nature and source of the images. These techniques include image thresholding (binarization), noise removal, smoothing, de-skewing, thinning (skeletonization), image dilation and normalization etc. [29]. The series of operations performed in the pre-processing phase of our proposed system with their implementation scenario are shown in Figs. 19b and 22, respectively, which include segmentation, normalization, binarization, and denoising and smoothing. Each of the pre-processing operations is discussed in detail as:

3.3.1 Segmentation

In this step, the rectangular bounding boxes of the input image within which individual Urdu characters are written are detected horizontally, and the characters within the box are then cropped (segmented) and each stored in the specific directory at the end of the pre-processing phase. The process is repeated for all 750 input images. In this way, all instances of each character are stored in the respective directory (class).

3.3.2 Normalization

In this step, each cropped character having different dimensions is normalized (resized) to a fixed size by padding each segmented character at the center of a 64×64 pixel image window so as to make identical input to the CNN.

3.3.3 Binarization

Binarization operation attempts to convert grayscale (RGB or) image to a binary-level image in such a way that foreground information is represented by black pixels and background by white pixels. Therefore, in a binary image, each pixel holds a value of 0 or 1 [35]. Binarization removes all the unnecessary pixel information (color, or gray shades) from the acquired image, which is not needed for an OCR system so that the resulting image is small, computationally fast, and easy to analyze. There are two types of binarization algorithms: *global* and *local adaptive thresholding*. In global thresholding, only a single threshold value is used for an entire image based on the background level estimation from the intensity of the image histogram, whereas in local adaptive, the threshold is computed for each pixel by considering neighborhood pixel information. If the pixel under observation is darker than its adjacent neighboring pixel, the pixel is converted into black, otherwise into white. Global thresholding is more feasible for non-varying background intensities, whereas local adaptive thresholding works well for degraded documents with uneven background illumination. [22, 37]. In the proposed system, each normalized character image undergoes *Otsu's method of global binarization* because the system is implemented on simple handwritten character images, where characters can be easily distinguished into the background and foreground pixels.

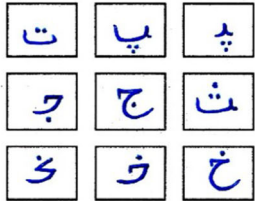




Scanned Image	Segmentation	Normalizing to Size: (64x64) pixels	Binarization	Denoising & Smoothing
				

Fig. 22 Pre-processing implementation

3.3.4 De-noising and smoothing

Digital images are prone to a variety of noise due to bad photocopying, scanning, or due to external factors (such as dust) that lead to the degradation of an image. The main objective of de-noising is to remove any unwanted bit patterns. De-noising techniques include filtering, morphological operation, and noise modeling. For smoothing, sharpening, thresholding, removing the slightly textured background and contrast adjustment process filters can be used. However, various morphological operations can be designed to smoothen contours, connect broken strokes, clipping unwanted points, thinning the characters, and extract the boundaries [22].

In this step, each binarized image is de-noised using a median filter of 3×3 pixel window of image and smoothed to preserve the contours of the character using morphological operation. The median filter is a nonlinear low pass filter which eliminates isolated pixel noise in gray level. The median filter takes an image window (3×3 , 5×5 etc.) and replaces the center pixel with the median value. If the area (neighborhood) under consideration contains an even number of pixels, the mean of the two middle pixel values is used. The median filter is effective for removing random occurrences of black and white pixel noise such as “salt and pepper noise.” A median filter is more effective than other filtering or morphological methods when the goal is to simultaneously reduce noise and preserve the edges [22].

3.4 Dataset generation

In order to remove the ambiguity in our OCR system in recognizing the handwritten Urdu characters, various character classes have been combined together to represent one class (instead of two), because of minor differences in the visual appearance of handwritten characters that could only be differentiated by the context in which these individual characters are written. Therefore, all the possible Urdu characters and numerals can be grouped into 133 unique classes rather than 142 classes. For example, *Alif* (ا) and numeral one (١) are written in the exactly the same way, and can only be differentiated by the context and direction of writing, and are thus combined. Similarly, other classes that have been merged together to represent a single unique class are:

- *Toi isolated* (ٹ) and *Toi initial* (ٹ)
- *Toi medial* (ٹ) and *Toi final* (ٹ)
- *Zoi isolated* (ظ) and *Zoi initial* (ظ)
- *Zoi medial* (ظ) and *Zoi final* (ظ)
- *Hay Do chasm isolated* (ح) and *Hay Do chasm initial* (ح)

- *Hay Do chasm medial* (ح) and *Hay Do chasm final* (ح)
- *Choti Yay isolated* (چ) and *Choti Yay final* (چ)
- *Yay-Bari isolated* (چ) and *Yay-Bari final* (چ)

The complete distribution of characters in 133 unique classes is shown in Fig. 23.

Once 133 unique classes have been created, the dataset is explicitly partitioned into train and test datasets, with the training dataset comprising of 80% samples, and the test dataset comprising of remaining 20% samples from each of the 133 classes. Furthermore, the training dataset is implicitly divided into two parts: a training set and a validation set. Finally, the training, validation, and testing dataset comprises of 70% (74, 285 samples), 10% (10, 612 samples) and 20% (21, 223 samples) of total samples (106,120 samples), respectively.

3.5 Classification

This phase presents the detailed overview of proposed CNN architecture, layers, activation functions, and various CNN hyper-parameters used for recognition of handwritten Urdu characters and numerals. The training dataset is used as an input to the proposed CNN to train our deep learning model, the validation dataset is used for hyper-parameter tuning during the training process and the test dataset is used to check how accurately the proposed CNN model is going to recognize the handwritten Urdu characters based on the training dataset.

3.5.1 Proposed CNN architecture

The algorithm used for our proposed OCR system is based on deep learning i.e., a CNN, a two-dimensional (2D) set of neurons, to classify handwritten Urdu characters and numerals. The overall CNN algorithm architecture used in the proposed OCR model is shown in Fig. 24, which comprises of the following layers: an input layer, 4 hidden convolutional layers each followed by a max pooling layer, and finally, two fully connected layers to perform classification. The input layer is equivalent to the input image size of 64×64 units. In the first convolution layer, we have used 64 arbitrary kernels each of 5×5 unit size (also called as a *local receptive field*) with default parameter values to produce 64 feature maps. The resulting feature maps are then passed to another convolution layer with a higher number of kernels to extract higher-level features of the input image. Therefore, the rest of the three convolutional layers have been designed with 128, 256, and 128 kernels, respectively, each having a fixed kernel size of 3×3 units, and zero padding. This helps CNN to learn few features for smaller receptive fields and more features for higher

Samples	Letter	Samples	Letter	Samples	Letter	Samples	Letter	Samples	Letter	Samples	Letter
1501	ھ, ھ	751	گ	1465	ط, ط	920	ژ	751	ج	1496	ا, ا
746	ء	746	گ	1479	ظ, ظ	614	ژ	749	ج	714	ا
1489	ی, ی	737	گ	1510	ظ, ظ	1025	ز	745	ج	766	ب
743	ی	749	گ	761	ع	722	ز	743	ج	746	ب
754	ی	750	ل	750	ع	972	ژ	733	ج	750	ب
1498	ے, ے	738	ر	750	ع	693	ژ	731	ج	708	ب
746	آ	749	ل	736	ع	766	س	747	ج	751	پ
744	ں	750	ل	736	غ	794	س	753	ج	751	پ
740	ں	744	م	738	غ	666	س	740	ج	762	پ
741	۰	739	م	750	غ	724	س	749	ج	751	پ
750	۲	737	م	749	غ	715	ش	732	ج	745	ت
743	۳	736	م	749	ف	741	ش	739	ج	752	ت
725	۴	769	ن	768	ف	731	ش	748	ج	751	ت
739	۵	749	ز	724	ظ	784	ش	740	ج	717	ت
743	۶	749	ذ	730	ظ	731	ص	740	ج	750	ث
750	۷	751	ن	751	ق	728	ط	762	د	752	ڈ
749	۸	751	و	747	ق	749	ط	723	د	752	ڈ
750	۹	736	و	746	ق	747	ص	751	ڈ	712	ٹ
		749	ہ	749	ق	711	ض	738	ڈ	752	ث
		734	پ	747	ک	725	ظ	731	ذ	738	ژ
		744	پ	753	ک	749	ظ	710	ذ	742	ژ
		730	ہ	716	ک	739	ض	993	ر	716	ش
		1417	ھ, ھ	747	ک	1500	ط, ط	694	ر	751	ج

Fig. 23 Distribution of handwritten Urdu character samples in each of 133 classes

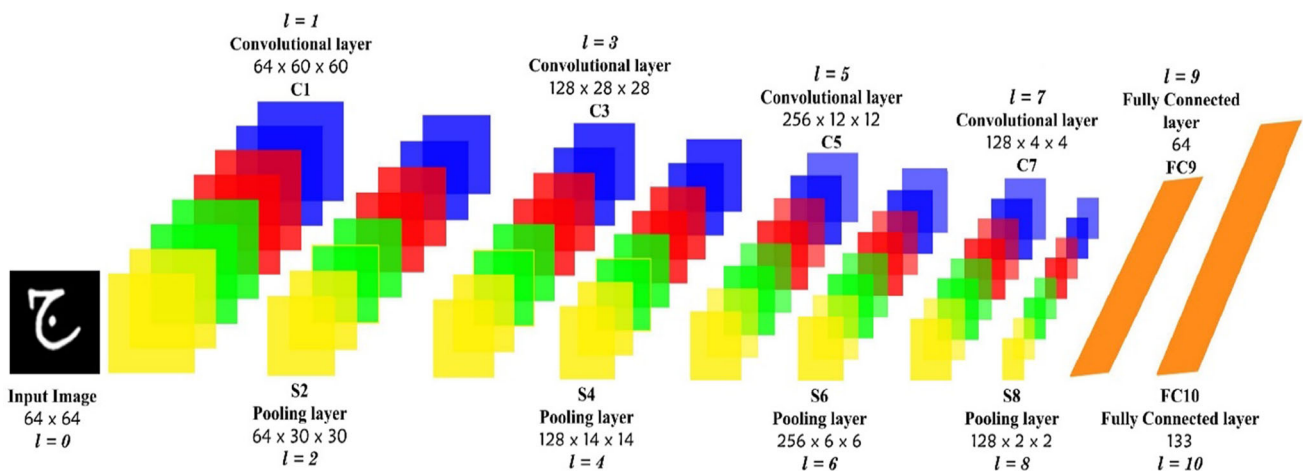


Fig. 24 The proposed CNN architecture for handwritten Urdu character recognition

receptive fields and hence extracting more abstract features for training. Each of the convolution layers is then followed by a max pooling layer with a pool size of 2 and zero paddings in order to reduce the spatial dimensions of output feature maps by a factor of 2, and eventually reduces the number of learnable parameters (less computation).

After performing various convolutional and down-sampling operations, the output features are flattened and squashed into dense layers to perform high-level reasoning. The proposed system is designed with two dense layers, which connect all the neurons of the previous layer to every single neuron it has by a learnable weight. The last fully connected layer contains 133 units of neurons equal to the number of classes to compute class scores (predicted probabilities) using the *softmax* activation function. Also, for every convolutional and fully connected layer output, *ReLU* activation function has been applied to increase the nonlinearity of the decision function and of the overall network without affecting the receptive fields of the convolution layer. *ReLU* is more plausible to biological neurons and makes the training significantly faster and improves the generalization ability of the deep neural network model.

3.5.2 CNN parameters

3.5.2.1 Learning rate The learning rate determines the speed at which the feedforward neural network is learning. It is denoted by a symbol α . A small value for α leads to slow convergence and a high value for α leads to divergence (i.e., the solution may not converge easily). An initial learning rate of 0.001 has been used in the proposed model.

3.5.2.2 Backpropagation optimizer The optimizer algorithm is responsible for adjusting the network weights in order to minimize the error (loss). We have used *Adam* Optimizer (*Adaptive moment estimation*) which is an effective variation of *stochastic gradient descent (SGD)*, requires less memory space, and is appropriate for large data with very noisy or sparse gradients. Unlike *SGD* which maintains a single learning rate for all weight updates, *Adam* optimizer computes individual adaptive learning rates for each weight of the neural network.

3.5.2.3 Mini-batch Dividing the training dataset into small subsets of instances (samples) is called a *batch*. The optimizer computes the gradient for each batch and updates the network parameters (e.g., weight etc.) to minimize the error. This process is called an *iteration* (i.e., one forward and backward pass). The number of iterations to process, evaluate the gradient, and update the parameters of all the batches of the training dataset is called an *epoch*. A model

may require many such epochs until it learns the examples of the dataset. In our model, we have used a batch size of 256 to train our model over 30 epochs.

3.5.2.4 Dropout Dropout is a regularization technique to deal with the problem of overfitting by ignoring randomly selected neurons during training and thus learn multiple internal representations of data. This means that their contribution to activations during the forward pass is temporarily removed and their corresponding weights are not updated during the backward pass. *Overfitting* refers to a situation where a model learns statistical regularities (i.e., memorizing noise) specific to the training dataset instead of learning the actual signal and therefore, performs less well on a subsequent new dataset. The dropout technique makes a neural network to better generalize and less likely overfit the training data. A dropout of 0.1 has been used in our model.

3.5.2.5 Loss function It is a method of determining how well specific algorithm models the given data. As a part of the optimizer, a loss function is used to repeatedly estimate the loss (error) for the current state of the model so that the network weights can be updated to reduce the loss in the next evaluation. For multi-class classification where each sample belongs to only one class, we have used *categorical cross-entropy* loss function in our model. It is also called as a softmax loss function as it is a combination of softmax activation function and cross-entropy loss function. The cross-entropy (CE) loss function after the softmax activation function is applied to each of the class score is defined as:

$$CE = - \sum_{i=1}^N t_i \log f(x_i)$$

where s_i is the CNN score and t_i is the ground truth for each class i . Since the labels in the multi-class classification are one-hot encoded, so only the positive class keeps its term in the loss. Therefore, there is only one element of the target vector which is not zero i.e., $t_i = t_p$. So, discarding the elements of the summation which are zero due to target labels, we can write cross-entropy for multi-class classification as:

$$CE = - \log \left(\frac{e^{x_p}}{\sum_{k=1}^N e^{x_k}} \right)$$

where x_p is the output CNN score for the positive class.

All the network hyper-parameters are often selected empirically and need a validation process to determine each of them ideally for the data at hand.

4 Experimental results

This section presents the results of experiments carried out to study the effectiveness of the proposed CNN model. We also investigated the improvement of performance with respect to different CNN hidden layers and compared the realized results with the latest recognition systems proposed for the Urdu script. As discussed earlier, the proposed CNN model was trained on a dataset of 74, 285 samples from the earlier mentioned values of CNN hyper-parameters over 10, 20, and 30 consecutive epochs in order to discover the best accuracy that can be obtained from our model. After applying all these runs incrementally, we found that the best accuracy of our model can be obtained at epoch 30. The curves for the accuracy and error rates (loss) over 30 epochs for training and cross-validation datasets are illustrated in Fig. 25. Classification rates after 30 epochs read at 99.60% and 98.83% on training and cross-validation datasets, respectively. Once the network was trained, the model was tested on the dataset comprising of 21, 223 samples and the test accuracy of 98.82% was achieved. In our study, we also performed experiments on different CNN layer models such as 2 layer CNN, 3 layer CNN and 4 layer CNN, however, the best recognition rates were observed with the CNN model having 4 hidden layers. Table 1 summarizes the accuracy and loss for training, validation, and test datasets on different layer CNN

models. It can be seen from Table 1, the more the hidden layers the better the recognition accuracy, however, with more hidden layers, the more complex the neural network and more computational time is needed to produce the results. For the purpose of this study, the recognition rate of 98.82% is adequate, and adding more layers would only result in increasing the complexity of the system.

To evaluate the overall performance of the proposed CNN model, various performance measures have been evaluated and summarized in Table 2.

In addition to various performance measures, Table 2 also shows the numbers of samples present and the number of samples that have been misclassified in each class. It is quite interesting to observe that most of the character classes (97 classes) have misclassified only 0 to 2 samples, 33 out of 133 classes have misclassified 3 to 6 samples and only 3 classes have misclassified 8, 9, and 13 samples each. The character ‘*Khai Final*’ tops the list with 13 misclassifications, this is followed by the character ‘*Seen Final*’ (س) with 9 misclassifications. The third position is occupied by the character ‘*Seen Initial*’ (س) with 8 misclassifications. The misclassification is primarily due to writer-dependent variations or due to the similarity in the way in which different isolated Urdu characters or different positional characters are written. Examples of such problem include the characters:

- *Ayn Isolated* (ع) and *Hi Final*

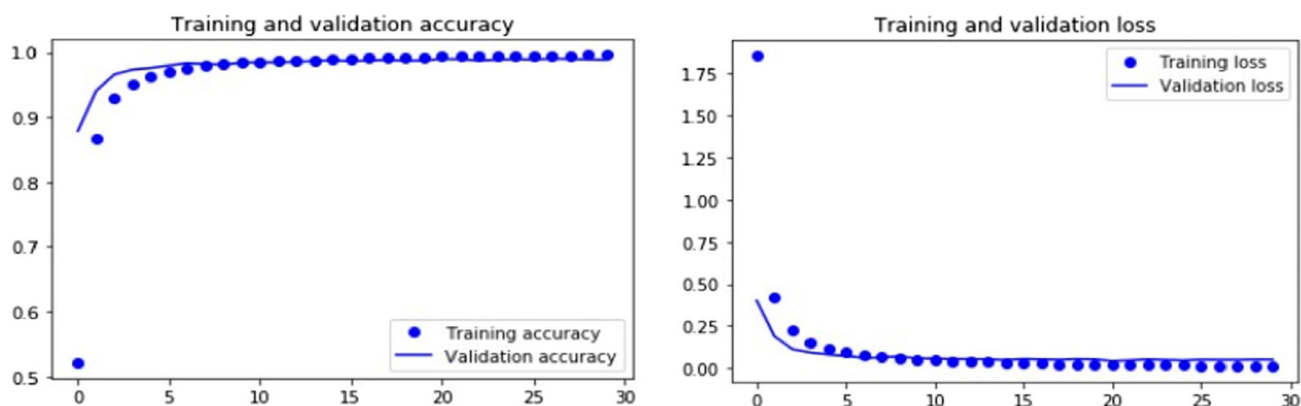


Fig. 25 Accuracy and loss curves for training and validation dataset over 30 epochs

Table 1 Accuracy and loss for different layer CNN models

Model		Proposed 4 layer CNN	3 layer CNN	2 layer CNN
Accuracy	Training	99.60%	99.43%	99.39%
	Validation	98.83%	98.67%	97.57%
	Test	98.82%	98.56%	97.05%
Loss	Training	0.0113	0.0183	0.0183
	Validation	0.0487	0.0567	0.1147
	Test	0.0504	0.0630	0.1324

Table 2 Performance measures of each class

Class	Samples	TP	FN	TPR/Recall	FPR/Fall Out	FNR/Miss Rate	Precision	F1 Score	ROC_AUC
1	299	298	1	0.99666	0.000096	0.00334	0.99333	0.99499	0.99828
2	143	142	1	0.99301	0.000047	0.00699	0.99301	0.99301	0.99648
3	153	153	0	1	0.000047	0	0.99351	0.99674	0.99998
4	149	148	1	0.99329	0.00019	0.00671	0.97368	0.98339	0.99655
5	150	146	4	0.97333	0	0.02667	1	0.98649	0.98667
6	142	141	1	0.99296	0	0.00704	1	0.99647	0.99648
7	150	150	0	1	0	0	1	1	1
8	150	150	0	1	0	0	1	1	1
9	152	150	2	0.98684	0.000142	0.01316	0.98039	0.98361	0.99335
10	150	150	0	1	0	0	1	1	1
11	149	149	0	1	0.000095	0	0.98675	0.99333	0.99995
12	150	147	3	0.98	0.000047	0.02	0.99324	0.98658	0.98998
13	150	147	3	0.98	0.000047	0.02	0.99324	0.98658	0.98998
14	143	141	2	0.98601	0.000047	0.01399	0.99296	0.98947	0.99298
15	150	150	0	1	0.000095	0	0.98684	0.99338	0.99995
16	150	146	4	0.97333	0	0.02667	1	0.98649	0.98667
17	150	147	3	0.98	0.000237	0.02	0.96711	0.97351	0.98988
18	142	139	3	0.97887	0	0.02113	1	0.98932	0.98944
19	150	150	0	1	0.000047	0	0.99338	0.99668	0.99998
20	148	146	2	0.98649	0.00019	0.01351	0.97333	0.97987	0.99315
21	148	148	0	1	0.000095	0	0.98667	0.99329	0.99995
22	143	143	0	1	0.000095	0	0.98621	0.99306	0.99995
23	150	150	0	1	0.000047	0	0.99338	0.99668	0.99998
24	150	150	0	1	0	0	1	1	1
25	150	150	0	1	0.000095	0	0.98684	0.99338	0.99995
26	149	148	1	0.99329	0	0.00671	1	0.99663	0.99664
27	149	148	1	0.99329	0	0.00671	1	0.99663	0.99664
28	147	146	1	0.9932	0.000047	0.0068	0.9932	0.9932	0.99657
29	146	144	2	0.9863	0	0.0137	1	0.9931	0.99315
30	149	149	0	1	0	0	1	1	1
31	151	151	0	1	0.000047	0	0.99342	0.9967	0.99998
32	148	146	2	0.98649	0.000047	0.01351	0.9932	0.98983	0.99322
33	150	148	2	0.98667	0.000047	0.01333	0.99329	0.98997	0.99331
34	146	140	6	0.9589	0.000237	0.0411	0.96552	0.9622	0.97933
35	148	148	0	1	0	0	1	1	1
36	150	148	2	0.98667	0.000047	0.01333	0.99329	0.98997	0.99331
37	148	144	4	0.97297	0	0.02703	1	0.9863	0.98649
38	148	139	9	0.93919	0.00019	0.06081	0.97203	0.95533	0.9695
39	152	151	1	0.99342	0.000095	0.00658	0.98693	0.99016	0.99666
40	145	145	0	1	0	0	1	1	1
41	150	148	2	0.98667	0.000047	0.01333	0.99329	0.98997	0.99331
42	148	144	4	0.97297	0.000047	0.02703	0.9931	0.98294	0.98646
43	146	142	4	0.9726	0.000095	0.0274	0.98611	0.97931	0.98625
44	142	139	3	0.97887	0.000047	0.02113	0.99286	0.98582	0.98941
45	199	199	0	1	0	0	1	1	1
46	139	139	0	1	0.00019	0	0.97203	0.98582	0.99991
47	184	182	2	0.98913	0.000428	0.01087	0.95288	0.97067	0.99435
48	123	122	1	0.99187	0	0.00813	1	0.99592	0.99593

Table 2 (continued)

Class	Samples	TP	FN	TPR/Recall	FPR/Fall Out	FNR/Miss Rate	Precision	F1 Score	ROC_AUC
49	205	199	6	0.97073	0.000143	0.02927	0.98515	0.97789	0.98529
50	144	144	0	1	0	0	1	1	1
51	194	190	4	0.97938	0.000238	0.02062	0.97436	0.97686	0.98957
52	139	137	2	0.98561	0	0.01439	1	0.99275	0.99281
53	153	151	2	0.98693	0.000664	0.01307	0.91515	0.94969	0.99313
54	159	158	1	0.99371	0.00019	0.00629	0.97531	0.98442	0.99676
55	133	127	6	0.95489	0.000047	0.04511	0.99219	0.97318	0.97742
56	145	132	13	0.91034	0.000095	0.08966	0.98507	0.94624	0.95512
57	143	143	0	1	0.000237	0	0.96622	0.98282	0.99988
58	148	140	8	0.94595	0.000285	0.05405	0.9589	0.95238	0.97283
59	146	141	5	0.96575	0.00038	0.03425	0.94631	0.95593	0.98269
60	157	153	4	0.97452	0	0.02548	1	0.9871	0.98726
61	146	145	1	0.99315	0.000142	0.00685	0.97973	0.98639	0.9965
62	146	146	0	1	0.000047	0	0.9932	0.99659	0.99998
63	150	150	0	1	0.000095	0	0.98684	0.99338	0.99995
64	149	147	2	0.98658	0.000047	0.01342	0.99324	0.9899	0.99326
65	142	137	5	0.96479	0	0.03521	1	0.98208	0.98239
66	145	144	1	0.9931	0	0.0069	1	0.99654	0.99655
67	150	148	2	0.98667	0.000095	0.01333	0.98667	0.98667	0.99329
68	148	148	0	1	0.000142	0	0.98013	0.98997	0.99993
69	300	298	2	0.99333	0	0.00667	1	0.99666	0.99667
70	293	293	0	1	0.000096	0	0.99322	0.9966	0.99995
71	296	296	0	1	0	0	1	1	1
72	302	302	0	1	0	0	1	1	1
73	152	148	4	0.97368	0.000285	0.02632	0.96104	0.96732	0.9867
74	150	146	4	0.97333	0.000237	0.02667	0.96689	0.9701	0.98655
75	150	144	6	0.96	0.000095	0.04	0.9863	0.97297	0.97995
76	147	145	2	0.98639	0.000095	0.01361	0.98639	0.98639	0.99315
77	147	141	6	0.95918	0.000427	0.04082	0.94	0.94949	0.97938
78	148	146	2	0.98649	0.00019	0.01351	0.97333	0.97987	0.99315
79	150	148	2	0.98667	0.000095	0.01333	0.98667	0.98667	0.99329
80	150	149	1	0.99333	0.000047	0.00667	0.99333	0.99333	0.99664
81	150	150	0	1	0.000095	0	0.98684	0.99338	0.99995
82	154	151	3	0.98052	0.00019	0.01948	0.97419	0.97735	0.99016
83	145	145	0	1	0.000095	0	0.98639	0.99315	0.99995
84	146	144	2	0.9863	0.000142	0.0137	0.97959	0.98294	0.99308
85	150	150	0	1	0	0	1	1	1
86	149	147	2	0.98658	0.000095	0.01342	0.98658	0.98658	0.99324
87	149	145	4	0.97315	0.000047	0.02685	0.99315	0.98305	0.98655
88	150	148	2	0.98667	0.000047	0.01333	0.99329	0.98997	0.99331
89	149	149	0	1	0.000095	0	0.98675	0.99333	0.99995
90	151	146	5	0.96689	0	0.03311	1	0.98316	0.98344
91	143	143	0	1	0.000095	0	0.98621	0.99306	0.99995
92	149	148	1	0.99329	0	0.00671	1	0.99663	0.99664
93	150	148	2	0.98667	0	0.01333	1	0.99329	0.99333
94	149	147	2	0.98658	0.000095	0.01342	0.98658	0.98658	0.99324
95	147	143	4	0.97279	0	0.02721	1	0.98621	0.98639
96	150	150	0	1	0.000047	0	0.99338	0.99668	0.99998

Table 2 (continued)

Class	Samples	TP	FN	TPR/Recall	FPR/Fall Out	FNR/Miss Rate	Precision	F1 Score	ROC_AUC
97	150	150	0	1	0	0	1	1	1
98	148	146	2	0.98649	0.000047	0.01351	0.9932	0.98983	0.99322
99	150	150	0	1	0.000047	0	0.99338	0.99668	0.99998
100	150	150	0	1	0	0	1	1	1
101	149	148	1	0.99329	0.000095	0.00671	0.98667	0.98997	0.9966
102	148	147	1	0.99324	0.000047	0.00676	0.99324	0.99324	0.9966
103	147	141	6	0.95918	0	0.04082	1	0.97917	0.97959
104	147	145	2	0.98639	0.000047	0.01361	0.99315	0.98976	0.99317
105	154	154	0	1	0.000047	0	0.99355	0.99676	0.99998
106	150	149	1	0.99333	0.00019	0.00667	0.97386	0.9835	0.99657
107	150	149	1	0.99333	0.000285	0.00667	0.96129	0.97705	0.99652
108	150	148	2	0.98667	0.000095	0.01333	0.98667	0.98667	0.99329
109	150	150	0	1	0.000047	0	0.99338	0.99668	0.99998
110	147	146	1	0.9932	0.00019	0.0068	0.97333	0.98316	0.9965
111	150	147	3	0.98	0.000095	0.02	0.98658	0.98328	0.98995
112	147	142	5	0.96599	0	0.03401	1	0.9827	0.98299
113	149	149	0	1	0	0	1	1	1
114	146	146	0	1	0.000095	0	0.98649	0.9932	0.99995
115	283	277	6	0.9788	0.000143	0.0212	0.98929	0.98401	0.98933
116	300	295	5	0.98333	0.000143	0.01667	0.98993	0.98662	0.99159
117	149	144	5	0.96644	0.000332	0.03356	0.95364	0.96	0.98306
118	298	296	2	0.99329	0.000143	0.00671	0.98997	0.99162	0.99657
119	149	149	0	1	0.000142	0	0.98026	0.99003	0.99993
120	151	150	1	0.99338	0.000237	0.00662	0.96774	0.98039	0.99657
121	300	299	1	0.99667	0	0.00333	1	0.99833	0.99833
122	149	149	0	1	0.000047	0	0.99333	0.99666	0.99998
123	149	149	0	1	0.00019	0	0.97386	0.98675	0.99991
124	148	145	3	0.97973	0.000047	0.02027	0.99315	0.98639	0.98984
125	148	148	0	1	0.000047	0	0.99329	0.99663	0.99998
126	150	147	3	0.98	0	0.02	1	0.9899	0.99
127	149	148	1	0.99329	0.000047	0.00671	0.99329	0.99329	0.99662
128	145	145	0	1	0.000095	0	0.98639	0.99315	0.99995
129	148	148	0	1	0.00019	0	0.97368	0.98667	0.99991
130	149	148	1	0.99329	0	0.00671	1	0.99663	0.99664
131	150	148	2	0.98667	0.000047	0.01333	0.99329	0.98997	0.99331
132	150	149	1	0.99333	0	0.00667	1	0.99666	0.99667
133	150	150	0	1	0	0	1	1	1
Total/Macro average	21,223	20,972	251	0.98779	0.000089	0.01221	0.98797	0.98780	0.99385

- *Gain Isolated* (غ) and *Khai Final*
- *Seen Initial* (س) and *Seen Medial* (سد)
- *Seen Isolated* (س) and *Seen Final* (س)
- *Sheen Initial* (ش) and *Sheen Medial* (شد)
- *Sheen Isolated* (ش) and *Sheen Final* (ش)

Also, some characters have the same shape and differ by the number and position of *nuqtas* (dots) below or above

the primary character. In this case, misclassification occurs due to the loss of the basic part of characters such as *dots*. Examples of such problem include the characters:

- *Beh* (ب) and *Peh* (پ)
- *Teh* (ت) and *Say* (ث)
- *Jeem* (ج), *Cheem* (چ), *Hi* (ح) and *Khai* (خ)
- *Daal* (د) and *Zaal* (ذ)

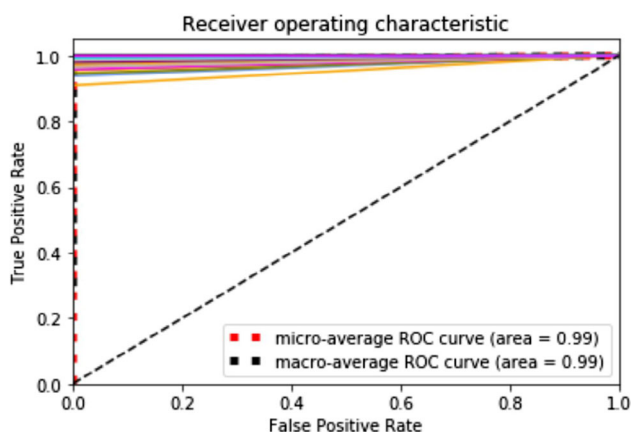


Fig. 26 ROC_AUC curve for 133 classes

- *Reh* (رہ) and *Zay* (زے) and *Cxay* (جے)
- *Seen* (سین) and *Sheen* (شین)
- *Suad* (صواد) and *Duad* (دواد)
- *Toi* (ٹوئی) and *Zoi* (زوئی)
- *Ayn* (این) and *Gain* (گین)
- *Noon* (نون) and *Noon-Ghunna* (نون گھننا) etc.

While some positional characters have coinciding geometrical shape similarity with some isolated characters

which can only be differentiated by the context in which they are written. Examples of such type of problem include characters:

- *Ttay Initial* (تے) and *Day Isolated* (ڈے)
- *Ttay Medial* (تے) and *Dhaal Isolated* (ڈھال)
- *Say Initial* (سے) and *Cxay Isolated* (جے)
- *Zay Isolated* (زے) and *Noon Initial* (نے)
- *Zaal Final* (زے) and *Noon Medial* (نے)

Also, some characters like ‘*Reh*’ and ‘*Daal*’ which can only be differentiated by writing their lower part, where the character ‘*Reh*’ is elongated smoothly than the character ‘*Daal*’ are also responsible for misclassifications. The above-mentioned characters are very confusing and cannot be distinguished easily as an individual character until we rely on the preceding characters of the same word or in general have an overall look of the whole word. Despite of such inherit complexities in Urdu script, the proposed model proved to be more effective in achieving a high accuracy rate of 98.82%, area under curve (AUC) rate of 99.39%, and low root-mean-square error rate (RMSE) of 4.24. Figure 26 shows the receivers operating characteristics (ROC) curve for all the 133 classes.

Table 3 Comparison with existing Urdu recognition systems

Authors (Year)	Technique used	Dataset, Size	Highest Recognition Rate (%)
Akram and Hussain [5]	HMMs	Dataset-1, 1446 Ligatures	98.37
Rizvi et al. [34]	Random Forest and Logistics Algorithm	129 Instances	98.40
Naz, Ahmed, et al. [28, 30]	Zoning features and 2-dimensional LSTM	UPTI, 10,000 Text Lines	93.39
K. Khan et al. [19]	Hu moments for feature extraction, Decision Tree J-48 for Classification	441 Characters	92.06
Din et al. [11]	HMMs	UPTI, 1525 Ligatures	92.26
Z. Ahmed et al. [3, 4]	Statistical Correlation and Pattern Matching	3500 Ligatures	97.40
Kadhm and Abdul [17]	DCT and HOG for feature extraction, SVM for Classification	2913 Words	96.32
Sobia T Javed et al. [16]	HMMs	1500 Ligatures	92.00
I. Ahmad et al. [1]	GBLSTM	UPTI, 10,063 Sentences	96.71
N. Javed et al. [14, 15]	CNN	18,000 Ligatures	95.00
Naz et al. [31]	CNN and RNN for Feature Extraction, MDLSTM for Classification	UPTI, 10,000 Text Lines	98.12
Naz, Umar, et al. [28, 30]	MDLSTM	10,000 Text Lines	96.40
Pal and Sarkar [32]	Topological, Contour and Water reservoir concept	NA	97.80
Proposed Technique	CNN	106,120 Characters	98.82

4.1 Comparison with existing systems

The recognition rate achieved by the proposed CNN algorithm reached up to 98.82% is the bestever result obtained by any technique proposed for handwritten Urdu as the best to our knowledge because of the complex nature of the Urdu language. If compared with existing Urdu character datasets, this study introduced a much diverse *Nasta'liq* Urdu dataset written by 750 Urdu writers than the existing ones. Also, the proposed dataset contains a large number of character classes (133) than the existing datasets where classes may range from 30 to 50 character classes and includes all the individual, positional as well as numeral characters. Furthermore, if compared on the basis of dataset samples, our dataset has 101,620 samples which is also higher than any other existing Urdu character datasets where samples may range from 441 to 41,228. Apart from the introduction of a new Urdu character dataset, this study also proposed a current state-of-the-art deep learning model (CNN) which is best known for its achievements in the field of image classification and computer vision. Therefore, this study will be very helpful for the development of future OCR systems of Urdu language. Table 3 demonstrates the comparison between the performance of the proposed OCR system with the existing systems evaluated on different datasets and including both analytical and holistic approaches.

5 Conclusion and future recommendations

In this paper, we proposed a new and comprehensive handwritten dataset for *Nasta'liq* Urdu script named Handwritten Urdu Character dataset (HUCD). The data have been gathered from 750 individuals of Kashmir valley covering isolated and positional characters as well as numerals. The main motive for preparing the HUCD offline dataset is to make it publicly available to the Urdu language research community for free of cost. Since no publicly available benchmark dataset is available for the *Nasta'liq* Urdu script, we also proposed a novel deep learning model based on CNN for the recognition of handwritten Urdu characters and numerals. The proposed analytical model achieved a state-of-the-art recognition accuracy of 98.82% with a root-mean-square error rate of 4.24. It was observed that the deeper the network the better the recognition result, however, with more hidden layers the complexity of the model also increases. It was also observed the maximum number of misclassifications occurs due to the similarity in geometrical shapes of various isolated and positional characters as well as the characters that were badly written or have some basic parts

missing. The purpose of using the deep learning model was to take full advantage of CNN, which has achieved breakthroughs in many fields of image recognition and does not need explicit feature engineering and extraction. In our further study, we intend to extend the model to recognize Urdu ligatures or words including all the diacritics (*Aerabs*) which play an important role in the pronunciation of Urdu words. The proposed recognition model can also be applied for recognition of other Urdu like scripts like *Kashmiri*, *Persian*, *Arabic*, *Turkish*, etc. as well as other Urdu writing styles like *Naskh*, *Kofi*, *Riqa*, *Aswad*, *Taleeq*, *Batool*, *Baghdadi*, *Jabeen*, etc. by preparing an appropriate dataset. Special characters can also be included to increase dataset range.

Funding No funding was received.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

- Ahmad I, Wang X, Hao Mao Y, Liu G, Ahmad H, Ullah R (2018) Ligature based Urdu Nastaleeq sentence recognition using gated bidirectional long short term memory. *Clust Comput* 21(1):703–714
- Ahmad Z, Orakzai JK, Shamsher I, & Adnan A (2007) Urdu Nastaleeq optical character recognition. *Paper presented at the Proceedings of world academy of science, engineering and technology*, pp. 2380–2383
- Ahmed SB, Naz S, Swati S, Razzak I, Umar AI, Khan AA (2017) UCOM offline dataset-an Urdu handwritten dataset generation. *Int Arab J Inf Technol (IAJIT)* 14(2):239–245
- Ahmed Z, Iqbal K, Mehmood I, Ayub MA (2017). Ligature analysis-based Urdu OCR framework. *Paper presented at the 2017 International Conference on Frontiers of Information Technology (FIT)*, pp. 87–92
- Akram QUA, Hussain S (2019) Improving Urdu recognition using character-based artistic features of nastalique calligraphy. *IEEE Access* 7:8495–8507
- Al-Rashaideh H (2006) Preprocessing phase for Arabic word handwritten recognition. *Inf Process (Russian)* 6(1):11–19
- Alom MZ, Taha TM, Yakopcic C, Westberg S, Sidike P, Nasrin MS, Asari VK (2019) A state-of-the-art survey on deep learning theory and architectures. *Electronics* 8(3):292
- Arica N, Yarman-Vural FT (2001) An overview of character recognition focused on off-line handwriting. *IEEE Trans Syst Man Cybern Part C Appl Rev* 31(2):216–233
- Daud A, Khan W, Che D (2017) Urdu language processing: a survey. *Artif Intell Rev* 47(3):279–311
- Din IU, Malik Z, Siddiqi I, Khalid S (2016) Line and ligature segmentation in printed Urdu document images. *J Appl Environ Biol Sci* 6(3):114–120
- Din IU, Siddiqi I, Khalid S, Azam T (2017) Segmentation-free optical character recognition for printed Urdu text. *EURASIP J Image Video Process* 2017(1):62

12. Farooq F, Govindaraju V, Perrone M (2005) Pre-processing methods for handwritten Arabic documents. *Paper presented at the Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. pp. 1–5
13. Jain M, Mathew M, & Jawahar C (2017) Unconstrained ocr for urdu using deep cnn-rnn hybrid networks. *Paper presented at the 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*
14. Javed N, Shabbir S, Siddiqi I, Khurshid K (2017) Classification of Urdu ligatures using convolutional neural networks-a novel approach. *Paper presented at the 2017 International Conference on Frontiers of Information Technology (FIT)*. pp. 93–97
15. Javed ST, Fasihi MM, Khan A, Ashraf U (2017) Background and punch-hole noise removal from handwritten urdu text. *Paper presented at the 2017 International Multi-topic Conference (INMIC)*. pp. 1–6
16. Javed ST, Hussain S, Maqbool A, Asloob S, Jamil S, Moin H (2010) Segmentation free nastalique urdu ocr. *World Academy Sci Eng Technol* 46:456–461
17. Kadhm MS, Abdul APDAK (2015) Handwriting word recognition based on SVM classifier. *Int J Adv Comput Sci Appl* 1:64–68
18. Kaushal DS, Khan Y, Varma DS (2014) Handwritten Urdu character recognition using Zernike MI's feature extraction and support vector machine classifier. *Int J Res* 1(7):1084–1089
19. Khan K, Khan RU, Alkhalifah A, Ahmad N (2015). Urdu text classification using decision trees. *Paper presented at the 2015 12th International Conference on High-capacity Optical Networks and Enabling/Emerging Technologies (HONET)*. pp. 56–59
20. Khan NH, Adnan A, Basar S (2018) Urdu ligature recognition using multi-level agglomerative hierarchical clustering. *Clust Comput* 21(1):503–514
21. Khan SN, Khan K, Khan A, Khan AU, Ullah B (2018) Urdu word segmentation using machine learning approaches. *Int J Adv Comput Sci Appl* 9(6):193–200
22. Kumar G, Bhatia PK, Banger I (2013) Analytical review of preprocessing techniques for offline handwritten character recognition. *Int J Adv Eng Sci* 3(3):14–22
23. Latif G, Alghazo J, Alzubaidi L, Naseer MM, Alghazo Y (2018) Deep Convolutional Neural Network for Recognition of Unified Multi-Language Handwritten Numerals. *Paper presented at the 2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR)*, 90–95
24. Mahmood A (2013) Arabic and Urdu text segmentation challenges and techniques. *Int J Comput Sci Technol* 4:32–34
25. Muaz A (2010) Urdu optical character recognition system MS thesis. National University of Computer and Emerging Sciences, Lahore Pakistan
26. Nautiyal CT, Singh S, Rana US (2017) Noisy Character Recognition. *Global J Pure Appl Math* 13(6):1875–1892
27. Naz S, Ahmed S, Ahmad R, Razza M (2015) Arabic script based digit recognition systems. *Paper presented at the International Conference on Recent Advances in Computer Systems*, pp. 67–73
28. Naz S, Ahmed SB, Ahmad R, Razzak MI (2016) Zoning features and 2DLSTM for Urdu text-line recognition. *Procedia Comput Sci* 96:16–22
29. Naz S, Hayat K, Razzak MI, Anwar MW, Madani SA, Khan SU (2014) The optical character recognition of Urdu-like cursive scripts. *Pattern Recogn* 47(3):1229–1248
30. Naz S, Umar AI, Ahmad R, Ahmed SB, Shirazi SH, Siddiqi I, Razzak MI (2016) Offline cursive Urdu-Nastaliq script recognition using multidimensional recurrent neural networks. *Neurocomputing* 177:228–241
31. Naz S, Umar AI, Ahmad R, Siddiqi I, Ahmed SB, Razzak MI, Shafait F (2017) Urdu Nastaliq recognition using convolutional–recursive deep learning. *Neurocomputing* 243:80–87
32. Pal U, Sarkar A (2003) Recognition of printed Urdu script. *Paper presented at the Seventh International Conference on Document Analysis and Recognition*, 2003, pp. 1–5
33. Raza AA, Habib A, Ashraf J, Javed M (2017) A review on Urdu language parsing. *Int J Adv Comput Sci Appl* 8(4):93–97
34. Rizvi SS, Sagheer A, Adnan K, Muhammad A (2019) Optical character recognition system for Nastalique Urdu-like script languages using supervised learning. *Int J Pattern Recogn Artif Intell* 33:1953004
35. Sardar S, Wahab, A (2010) Optical character recognition system for Urdu. *Paper presented at the 2010 International Conference on Information and Emerging Technologies*, pp. 1–5
36. Sattar SA (2009) A Technique for the Design and Implementation of an OCR for Printed Nastalique Text. NED University of Engineering and Technology Karachi, Sindh Pakistan
37. Shafait F, Keysers D, Breuel T (2008) Efficient implementation of local adaptive thresholding techniques using integral images. *Electron Imaging Int Soc Optics Photonics* 6815:681510–681510
38. Shrestha A, Mahmood A (2019) Review of deep learning algorithms and architectures. *IEEE Access* 7:53040–53065
39. Singh D, Khan MA, Bansal A, Bansal N (2015) An application of SVM in character recognition with chain code. *Paper presented at the 2015 Communication, Control and Intelligent Systems (CCIS)*. pp. 1–5
40. Yamashita R, Nishio M, Do RKG, Togashi K (2018) Convolutional neural networks: an overview and application in radiology. *Insights Imaging* 9(4):611–629

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.