Hindawi
Mobile Information Systems
Volume 2021, Article ID 7211419, 16 pages
https://doi.org/10.1155/2021/7211419

Research Article

Detection of Touchscreen-Based Urdu Braille Characters Using Machine Learning Techniques

Sana Shokat,[1] Rabia Riaz,[1] Sanam Shahla Rizvi,[2] Inayat Khan,[3] and Anand Paul[4]

[1]*Department of Computer Science and IT, University of Azad Jammu and Kashmir, Muzaffarabad, CO 13100, Pakistan*
[2]*Raptor Interactive (Pty) Ltd., Eco Boulevard, Witch Hazel Ave, Centurion 0157, South Africa*
[3]*Department of Computer Science, University of Buner, Buner 19290, Pakistan*
[4]*The School of Computer Science and Engineering, Kyungpook National University, Daegu, Republic of Korea*

Correspondence should be addressed to Anand Paul; paul.editor@gmail.com

Received 9 September 2021; Revised 23 October 2021; Accepted 20 November 2021; Published 27 December 2021

Academic Editor: Zhongguo Yang

Revolution in technology is changing the way visually impaired people read and write Braille easily. Learning Braille in its native language can be more convenient for its users. This study proposes an improved backend processing algorithm for an earlier developed touchscreen-based Braille text entry application. This application is used to collect Urdu Braille data, which is then converted to Urdu text. Braille to text conversion has been done on Hindi, Arabic, Bangla, Chinese, English, and other languages. For this study, Urdu Braille Grade 1 data were collected with multiclass (39 characters of Urdu represented by class 1, Alif (ا), to class 39, Bri Yay (ے). Total ($N = 144$) cases for each class were collected. The dataset was collected from visually impaired students from The National Special Education School. Visually impaired users entered the Urdu Braille alphabets using touchscreen devices. The final dataset contained ($N = 5638$) cases. Reconstruction Independent Component Analysis (RICA)-based feature extraction model is created for Braille to Urdu text classification. The multiclass was categorized into three groups (13 each), i.e., category-1 (1–13), Alif-Zaal (ا - ذ), category-2 (14–26), Ray-Fay (ر - ف), and category-3 (27–39), Kaaf-Bri Yay (ق - ے), to give better vision and understanding. The performance was evaluated in terms of true positive rate, true negative rate, positive predictive value, negative predictive value, false positive rate, total accuracy, and area under the receiver operating curve. Among all the classifiers, support vector machine has achieved the highest performance with a 99.73% accuracy. For comparisons, robust machine learning techniques, such as support vector machine, decision tree, and *K*-nearest neighbors were used. Currently, this work has been done on only Grade 1 Urdu Braille. In the future, we plan to enhance this work using Grade 2 Urdu Braille with text and speech feedback on touchscreen-based android phones.

1. Introduction

Smart devices are the most powerful tool for improving people's living standards with visual disabilities [1]. Recent trends predict a drastic increase in smartphone users, with an expected increase of up to 9 billion by 2021 [2, 3]. There are various applications meant to assist visually impaired users, such as screen readers, sound and speech output devices, location finders, wearable devices for mobility, stereo vision-based systems, and virtual assistants [4–7]. Rapid growth in smartphone usage has changed the learning attitude of people [8]. People are increasingly turning to technology to explore new ideas. People learn by watching videos, tutorials, and online courses on their smart devices [9]. Braille is a commonly used language for visually impaired people. Louis Braille designed it in 1821. Braille is comprised of six dots in the form of two columns and three rows [10]. Visually impaired people write on sheets with the help of a stylus and read by gliding their fingers over the raised dots. It is difficult for visually impaired people to write Braille using these devices. Some interfaces convert textbooks into Braille books, but this facility is limited to specific languages; there is no procedure for converting Urdu text into Braille [11]. Previously, visually impaired people could only use their phones to make phone calls and send and receive text messages [12]. But now, people with visual

impairments can read Braille with the help of a different screen reader software like Apple's VoiceOver [13]. Different applications such as NavTap, Braille Play [14], Braille Tap [15], Braille Key, TypeIn Braille [16], Perkinput [17], Braille Easy [18], Eyedroid [19], and DRISHYAM [20] were developed to facilitate Braille text entry using smart devices. Although audio feedback was provided for user assistance in these applications, they also used numerous difficult gestures to memorize and took more time to perform a specific task. Due to usability issues, visually impaired users are unable to access such applications. Research is in progress for making applications that are less time-consuming and more usable. For Braille to Natural language conversion, image processing techniques were applied on scanned Braille sheets. Braille has been converted into Arabic [21], English [22], Bengali [23], Hindi [24], Tamil, maths [25–27], and Odia [28] using these techniques, respectively. Braille is converted into Urdu and Hindi using deterministic Turing machines [29] and image segmentation algorithms [30].

Previously, Braille was translated into other languages using scanned sheets as input. Their conversions are hectic due to extensive writing on those sheets by the users. Several touchscreen-based applications provide text-to-speech conversion methods that assist visually impaired people with reading and writing. Most of these applications burden the user [31, 32], such as memorizing so many gestures, finding the position of dots on the screen, and no editing options. A position-free Braille text entry method was proposed to address these problems. That application was designed to put the least burden on the users while entering the English Braille alphabet. Visually impaired users can enter Braille characters by clicking anywhere on the screen, subsequently saved in an image format. For character recognition, deep learning techniques with the GoogLeNet inception model achieved more than 95% accuracy [33]. As per our knowledge, there are very few studies for Urdu Braille data, and none of them takes user input directly from the touchscreen. So, there is a strong need for an application that takes run-time Braille data and converts it into natural languages. Currently, there is no mechanism available for Braille to Urdu conversion using touchscreen-based devices. So, in this study, the front-end interface proposed by Sana et al. was used to collect the Urdu Braille dataset. Braille input was saved in an image format in the previous version of this application. Here, with some backend processing algorithm improvements, Urdu Braille input is saved in numerical data. Machine learning techniques such as DT, SVM, and KNN with RICA-based feature extraction methods are used for Braille to Urdu conversion on the new Urdu Braille dataset, see Figure 1.

The main contributions of this research are as follows:

(a) Collection of Urdu Braille Grade 1 dataset using the application developed by Sana et al. from visually impaired students of the Special Education School, Manak Payyan, Pakistan [33]. There was no existing Urdu Braille dataset that was taken directly from touchscreen devices.

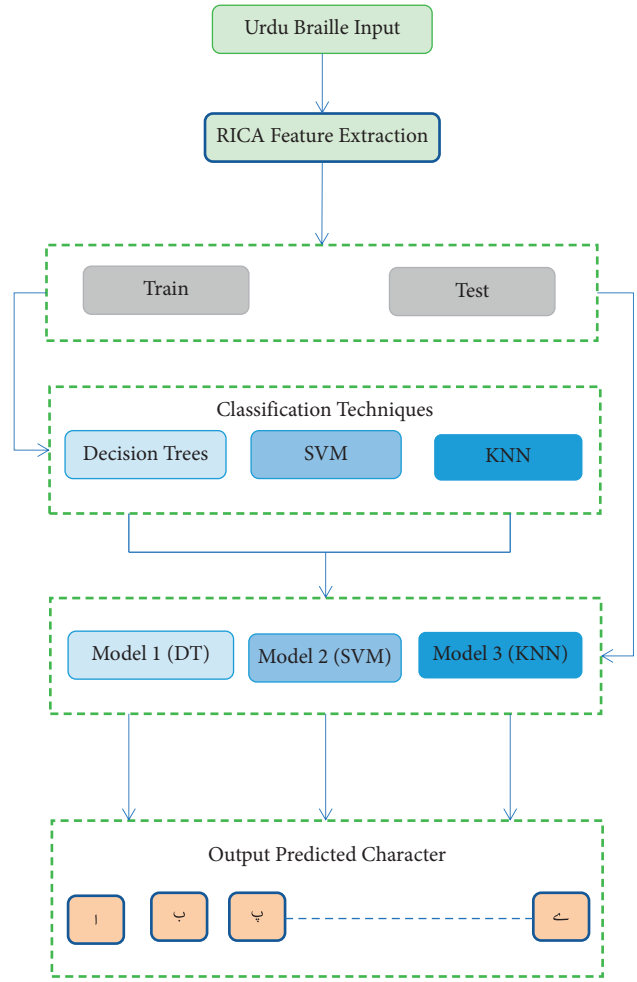

Figure 1: Schematic diagram for Braille to Urdu character prediction.

(b) Enhancement in the backend processing mechanism of Sana et al. has been proposed.

(c) Predication of Urdu character from Braille input is made with robust machine learning techniques, such as decision tree (DT), support vector machine (SVM), and *K*-nearest neighbor (KNN), using RICA-based feature extraction method.

(d) Evaluation of proposed mechanism on the collected Urdu dataset is made based on true positive rate (TPR), true negative rate (TNR), positive predicted value (PPV), negative predicted value (NPV), false positive rate (FPR), total accuracy (TA), and area under the curve (AUC).

(e) A comparative analysis with previous studies using scanned input-based data of different national and regional languages has been performed.

This study comprises the following sections: Section 2 provides materials and methods, information regarding the dataset, and its collection procedure. In Section 3, the results are presented in detail. Discussion is provided in Section 4, and finally, conclusions and future work are presented in Section 5.

2. Materials and Methods

2.1. Dataset. The front-end android-based application proposed was used to collect the Urdu dataset for this study [33]. On the android-based touchscreen, visually impaired users enter Urdu Braille characters. The dataset is collected from the National Special Education School Center (NSEC) "Manak Payyan." The age of the participants was between 12 and 19 years, and these students were either completely or partially blind. At this level, Urdu Grade 1 Braille data were collected, which included 39 distinct characters.

The final dataset comprises 5637 Urdu Braille characters collected using an android-based touchscreen device. Machine learning techniques are utilized to convert Braille to Urdu text using this dataset. All the ambiguous data were eliminated after checking the values against each alphabet manually.

2.2. Backend Processing Mechanism. An improved backend processing mechanism is proposed in this study. In work done earlier, the dataset consisting of images corresponding to each character was used.

In the current study using the position-free interface, values of "x" and "y" coordinating against each dot are stored in a database. Braille is composed of six-dot patterns, and each Braille character is represented by the activation and deactivation of these dots. For example, if a Braille character has two active dots, the proposed system will save the value of "x" and "y" coordinates of active dots, and the remaining four inactive dots will be assigned a "0" value. Initial data were stored in the form of a .txt file separated by a comma. To avoid ambiguity in the dataset, the researcher manually checked the dataset, commas were removed, and each extracted dot was stored in a .csv file. Figures 2(a) and 2(b) show a visually impaired user entering Dal "ڈ" and Toyn "ٹ" using a touchscreen-based Braille interface.

The algorithm for Braille input coordinate extraction is shown in Figure 3.

Since the only coordinate location of active dots is saved in the dataset for each character instead of the whole image, thus, this approach reduces the storage requirements. In previous schemes, the single image saved in the database was 4 to 8 KB in size. For multiple instances, these requirements get multiplied by the number of cases, whereas with the new approach, a text file containing 144 different samples of a single character took only 9 to 10 KB space.

2.2.1. Feature Extraction. Different feature extraction techniques were applied earlier to predict Braille to text conversion for other languages. Jha and Parvathi extracted HOG features using Braille to Hindi text conversion [24]. Similarly, Li et al. have used the traditional feature extraction method using KNN, Naïve Bayes, etc., for recognizing Braille characters [34]. Moreover, Li et al. used a histogram of oriented gradients (HOG) with SVM to convert Braille characters into English, Sinhala, and Odia [35, 36]. This study extracts RICA-based features using DT, SVM, and KNN classifiers for converting Braille to Urdu text. This feature extraction method extracts more features than the actual dimensions of the input dataset. This algorithm also has the ability for faster execution of preprocessing steps.

2.2.2. RICA Feature Extraction Method. Reconstruction independent component analysis (RICA) is not a supervised learning technique, so it does not utilize the class label information. RICA algorithm was introduced to address the limitations of the ICA algorithm. This technique delivered more promising results than ICA. A lot of algorithms have been presented in recent years to learn sparse features.

A sparse filter can differentiate many artificial and natural signals, and this feature plays a vital role in different machine learning techniques.

The unlabeled data are given as input

$$\left\{y^i\right\}_{i=1}^{n}, \quad y^i \in \mathbb{R}^m. \tag{1}$$

For calculating independent components, the problem of optimization of standard ICA [37] can be defined mathematically as

$$\begin{aligned} &\min_X \frac{1}{n}\sum_{i=1}^{n} h\left(Xy^i\right), \\ &\text{subject to } XX^U = I, \end{aligned} \tag{2}$$

where $h(.)$ represents a nonlinear penalty function, $X \in S^{Lxm}$ is a matrix, L represents the vectors count, and I defines the identity matrix. Additionally, $XX^U = I$ is employed to prevent the vectors in X from being degenerated. For this purpose, a smooth penalty function can be used, i.e., $h(.) = \log(\cosh(.))$ [27].

However, some constraints related to orthonormality block the standard independent component analysis from learning on an overcomplete basis. Consequently, the defect, as mentioned above, prevents ICA from scaling into high-dimensional data. Hence, for the replacement of orthonormality constraints in ICA, soft reconstruction cost is used in RICA. After this substitution, the following unconstrained problem can be used to represent RICA filtering:

$$\min_X \frac{\lambda}{n}\sum_{i=1}^{n}\left(\left\|X^U X y^i - y^i\right\|_2^2 + \sum_{i=1}^{n}\sum_{k=1}^{l} h\left(X_k y^i\right)\right). \tag{3}$$

In the above-stated equation, $\lambda > 0$ exhibits the tradeoff between sparsity and reconstruction error rates. After performing swapping orthonormality constraints with reconstruction cost, in this way, even on unwhitened data, RICA can learn sparse representations when X is overcomplete. However, penalty h can yield sparse representations and is not invariant [38]. Therefore, RICA [39] swapped it by an additional pooling penalty represented by L2, simultaneously promoting pooling features to cluster correlated features. Moreover, for feature learning, L2 pooling also encourages sparsity. L2 pooling [40, 41] represents a two-layered network; the 1st layer is $(.)^2$ with square nonlinearity, and in the 2nd layer, $\sqrt{(.)}$ square root nonlinearity,

(a) (b)

Figure 2: (a) A visually impaired student entering Urdu character Dal "د." (b) A visually impaired student entering Urdu character Toyn "ط".

ALGORITHM *Braille Input Coordinate Extraction*

1. **Step 1:** Initialize Variables; *minX*, *minY*, *maxX*, *maxY*, *x* & *y*
2. *minX=20*
3. *minY=20*
4. *maxX= scrnwidth – 20*
5. *maxY= scrnheight – 20*
6. **Step 2:** Draw Point
7. **If** (x_i >= (scrnwidth - 30))
8. x_i = *maxX;*
9. **If** (y_i >= (scrnheight - 30))
10. y_i = *maxY;*
11. **else**
12. **If** (x_i <= 20)
13. x_i = *minX;*
14. **If** (y_i <= 20)
15. y_i = *minY;*
16. Get (x_i, y_i) *//Draw circle at x_i and y_i with radius 15 pixels*
17. Repeat **Step 2** [7-16] for each "ActiveDot"
18. **Step 3**: Check swipe direction
19. **If** event: swipe left to right
20. SaveCoordinate (x_i, y_i) ; *//for each Active Dot*
21. SaveCoordinate (0,0); *//for each Inactive Dot*
22. **else** event: swipe right to left
23. ClearScreen();

Figure 3: Proposed algorithm for Braille input coordinate extraction.

$$h\left(Xy^i\right) = \sum_{k=1}^{L} \sqrt{\varepsilon + H_k \cdot \left(\left(Xy^i\right) \odot \left(Xy^i\right)\right)} \tag{4}$$

Pooling matrix $H \in P(L \times L)$ where H_k denotes a row of that pooling matrix set to constant weights, i.e., 1 for every element in matrix H, element-wise multiplication is defined by $\odot$, and $\varepsilon > 0$ is a small constant. RICA is a linear method that investigates the sparse representation only in the actual data space. RICA method is unable to use the association between class label information and training samples.

2.3. Classification. A process for categorizing classes according to the extracted features is known as classification. There are different machine learning techniques, such as supervised, unsupervised, reinforcement, ensemble, neural networks, and deep learning [42]. Machine learning approaches such as DT, KNN, and SVM based on RICA-based feature extraction methods are applied for character prediction. 70%–30% data are used for training and validation purpose [43].

2.3.1. Decision Tree. A DT is a machine learning technique that is used for prediction. DT is popular because it does not require too many computations [44]. DT classifiers have a tree-like structure that divides the dataset into several subsets. This classifier trains the model by applying simple decision rules on training data [45]. The model is then used to forecast the desired values, read the dataset, and categorize them into classes [8].

The following equations can be used to design DT algorithms mathematically.

$$\begin{aligned} \overline{X} &= \{X_1, X_2, X_3, \ldots, X_m\}^T, \\ X_i &= \{x_1, x_2, x_3, \ldots, x_{ij}, \ldots, x_{in}\}, \\ S &= \{S_1, S_2, \ldots, S_i, \ldots, S_m\}. \end{aligned} \tag{5}$$

In this study, train and test data are divided, with a 70%–30% ratio. Training data are used to build a model, and test data are used to check the model's validity. A multiclass approach is used to predict Braille to Urdu text using DT. The DT is tuned using the default parameters.

2.3.2. KNN. KNN is the most common and simple nonparametric technique used for regression and classification models in machine learning. The Euclidian distance formula [46] is used to calculate the distance between the samples.

$$EU_{a,b} = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}, \tag{6}$$

where a and b represent the number of samples.

$$a_i - b_i, \tag{7}$$

where a_i–b_i are the i^{th} dimension feature dimensions of samples, and n represents the total number of feature dimensions.

The number of nearest neighbors determines the output value while using KNN. If the value of $K = 1$, the object can be classified and assigned to the nearest neighbor of that single class [45].

Here, KNN is used to classify Braille to Urdu text. $K = 3$ is selected, distance metrics as Euclidean distance, and distance weight as equal weight.

2.3.3. SVM. For pattern and character recognition, SVM is the most well-known machine learning classification technique. SVM is a supervised machine learning technique used in biomedical image processing, computer vision, speech recognition, etc. [47]. SVM builds a hyperplane in high-dimensional spaces to obtain a better classification. If the achieved hyperplane has the highest functional margin, the classifier will give good performance [48]. The greater the margin, the lower the risk of generalized error. SVM finds the hyperplane that provides the most significant minimum distance for the training data. SVM can produce more generalized outcomes. SVM is a twofold classifier that converts data into a hyperplane that depends upon high-dimensional data.

Let us consider a hyperplane $x.\ w + b = 0$, where w is normal.

Linearly separable data are represented as follows:

$$\{x_i, y_i\},\ x_i \varepsilon R^N d,\ y_i \varepsilon \{-1, 1\}, \quad i = 1, 2, \ldots, N, \tag{8}$$

where y_i is the twofold class label.

When we achieve maximum margin by maximizing, the objective function value of
$E = \|w\|^2$ gives

$$\begin{aligned} x_i.w + b &\geq 1, \text{for} y_i = +1, \\ x_i.w + b &\leq 1, \text{for} y_i = -1. \end{aligned} \tag{9}$$

By removing the discrepancies from the above equations, now we have

$$(x_i.b + b) y_i \geq 1, \text{ for all } i. \tag{10}$$

If data cannot be linearly separated, then a slack variable Ξi is used to identify misclassifications.

Thus, in this scenario, an objective function is defined as

$$E = \frac{1}{2}\|w\|^2 + C \sum_i L(\Xi i), \tag{11}$$

subject to

$$(x_i.b + b) y_i \geq 1 - \xi_i, \text{ for all } i. \tag{12}$$

Here, C and L represent hyperparameters and cost functions, respectively. Cost functions are used to detect outliers. The dual formulation with $L(\Xi i) = \Xi i$ is

$$\alpha = \max_\alpha \left(\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j \right), \tag{13}$$

subject to

$$\begin{aligned} 0 \leq \propto_i \leq C, \\ \sum_i \propto_i y_j = 0. \end{aligned} \tag{14}$$

Here,

$$\begin{aligned} \propto &= \{\propto_1, \propto_2, \propto_3, \ldots, \propto_i\}, \\ \omega_0 &= \sum_i \alpha_i x_i y_i. \end{aligned} \tag{15}$$

Kernel trick is used to handle nonlinearly separable data [49]. The nonlinear mapping function from the input space is transformed into a higher dimensional feature space. Polynomial, Gaussian, and radial-based functions are the most popular kernels.

SVM polynomial kernels:

$$K(x_i, y_i) = \exp\left(\frac{-1}{2} \frac{\|x_i - y_i\|^2}{\sigma^2} \right). \tag{16}$$

SVM Gaussian polynomial kernels:

$$K(x_i, y_i) = \exp\left(\frac{-1}{2} \frac{\|x_i - y_i\|^2}{\sigma^2} \right). \tag{17}$$

SVM fine Gaussian kernels:

$$K(x_i, y_i) = \exp\left(\frac{-1}{2} \frac{\|x_i - y_i\| \|x_i - y_i\|}{\sigma^2} \right). \tag{18}$$

Dual formation of a nonlinear case is shown as

$$\alpha^* = \max_\alpha \left(\sum_i \alpha_i + \sum_{i,j} \alpha_i \alpha_j y_i y_j, K(x_i.y_j) \right), \tag{19}$$

subject to

$$\begin{aligned} 0 \leq \propto_i \leq C, \\ \sum_i \propto_i y_j = 0. \end{aligned} \tag{20}$$

Grid search is the famous evaluation metric used for SVM evaluation. Optimal parameters are carefully selected by setting the grid range and step size. Only one parameter, "c," a soft margin constant, is used in a linear kernel, whereas the SVM Gaussian kernel and SVM fine Gaussian kernel contain two training parameters, cost "c" and sigma, which can be used to control the nonlinearity of the degree. RICA feature extraction method was employed in the study with 70% data for training and 30% for testing. In this study, a polynomial kernel with default parameters is used.

2.4. Performance Evaluation Metrics. To predict Urdu Braille characters, true positive rate (TPR), true negative rate (TNR), positive predicted value (PPV), negative predicted

value (NPV), false positive rate (FPR), and total accuracy (TA) are used to evaluate Urdu Braille character prediction.

2.4.1. True Positive Rate. TPR is also called sensitivity or recall. TPR indicates how many correct alphabets are classified as true. Mathematically, it is written as

$$\frac{\text{recall}}{\text{sensitivity}} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{21}$$

2.4.2. True Negative Rate. TNR is also called specificity. This metric defines the number of negative predicted values that are correctly identified. It can be expressed mathematically as

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{22}$$

2.4.3. Positive Predictive Value. A test predicted positive results when it is true positive. Mathematically, it can be presented as

$$\text{precision}\,(\text{PPV}) = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{23}$$

2.4.4. Negative Predictive Value. NPV shows a test result in negative prediction, and the subject also has a negative value. Mathematical representation is given as follows:

$$\text{negative predictive value}\,(\text{NPV}) = \frac{\text{TN}}{\text{TN} + \text{FN}}. \tag{24}$$

2.4.5. Total Accuracy. Total accuracy is defined by adding all true positives and all true negatives and dividing it by all true negatives, false positives, true positives, and false negatives.

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TN} + \text{FP} + \text{TP} + \text{FN}}. \tag{25}$$

2.4.6. ROC. It measures the proportions of all true positives and all true negatives. They are calculated by plotting the ROC curve against the true positive and false positive values. TPR is plotted along the x-axis, whereas FPR is plotted along the y-axis. The area under the curve (AUC) value lays between 0 and 1. A value >0.5 shows separation. A value greater than 0.5 indicates separation. In this study, Braille Urdu characters, which are predicted true when they belong to the true class, have values 1 or approaching 1.

3. Results

Urdu Braille characters are predicted from a newly collected dataset from touchscreen devices. The performance is computed using RICA feature extraction methods and machine learning algorithms such as DT, KNN, and SVM. TPR, TNR, PPV, NPV, TA, FPR, and AUC were the performance metrics employed in the evaluation.

Figures 4(a)–4(c) show AUC values using DT for category-1 (class 1–class 13), Alif-Zaal (ا - ذ), category-2 (class 14–class 26), Ray-Fay (ر - ف), and category-3 (class 27–class 39), Qaaf-Bri Yay (ق - ے), respectively. By extracting RICA features, the highest performance attained from category-1 (class 1–class 13), Alif-Zaal (ا - ذ), is Braille class 6 with 99.9% TA and with 0.9979 AUC value, as shown in Figure 4(a). They were followed by other classes such as Alif, Bay, Hay, and Khay (ا, ب, ح, خ) yielding accuracies of (99.90%), with AUC (0.9995, 0.9914, 0.9895), respectively. Other classes such as Chay and Zaal (چ، ذ) also achieved better accuracies of 99.85% and 97.76% with AUC (0.9884 and 0.9887, respectively). Other performance measures, such as TPR and TNR, yield performance (TPR >94%) and (TNR >98%). From category-2 (14–26), Ray-Fay (ر - ف) shows the highest performance for class 18, 21, 14, and 15, i.e., Seen, Zuuad, Ray, Rray, Fay, Toyn, and Ghaen (س, ض, ر, ڑ, ف, ط, غ) with accuracies of (99.95%, 99.95%, 99.85%, 99.85%, 99.80%, 99.64%, and 99.64%), with AUC (0.9899, 0.9997, 0.9992, 0.9495, 0.9615, 0.9495, and 0.9683, respectively) with TPR >90% and TNR >99%, as shown in Figure 4(b). For category-3 (27–39), Qaaf-Bri Yay (ق ے), maximum TA of 100% is achieved for class 35 (Gol Hy), 37 (Hamza), and 28 (Kaaf) (ه, ء, ك) with AUC value of (0.9553, 1, 0.9995). These results are followed by Gaaf, Meem, and Hy (گ, م, ھ) with TA of (99.75%, 99.85%, and 99.75%) (with AUC value of (0.0732, 0.8125, and 0.9553) TPR >94%, and TNR >99%, except for class 31, i.e., TPR = 62.50%, as shown in Figure4(c). Amongst all, the highest separation (AUC = 1) was seen for the Urdu Braille characters Gol Hy and Hamza (ه, ء). AUC value of other Urdu Braille characters such as Alif, Bay, Pay, and Tay (ا, ب, پ, and ت) are above 99% indicating good classification. Detailed results are presented in Table 1.

The maximum accuracy obtained by using KNN for category (1–13), Alif-Zaal (ا - ذ), is for class 2, 4, 5, and 13, i.e., Bay, Tay, Ttay, and Zaal (ب, ت, ٹ, ذ) with TA (99.95%, 99.85%, 99.80%, and 99.75%) and AUC (0.9997, 0.9992, 0.9990, and 0.9896, respectively), as shown in Figure 5(a). For category-2 (14–26), (Ray-Fay) (ر - ف), highest TA of 100% is achieved for class 15, 18, and 23, followed by 26, 21, 14, and 19, i.e., Rray, Seen, and Zoyn (ظ, س, ڑ). Fay and Zuad (ض, ف) achieved a TA of 099.95%, while Ray and Sheen (ر, ش) had TAs of 99.90% and 99.75%, respectively, with AUC (1, 1, 1, 0.9884, 0.9922, 0.9995, and 0.9800) with TPR >96% and TNR >99%, as shown in Figure 5(b). Similarly, highest TA achieved for category-3 (27–39), (Qaaf-Bri Yeh) (ق - ے), is for class 35, 37, 31, 30, 27, and 28, i.e., Gol Hy, Hamza, Meem, Laam, Qaaf, and Kaaf (ه, ء, م, ل, ق, ك) (99.75%, 99.80%, 100%, 100%, 99.69%, and 99.69%), with AUC (1, 1, 0.7143, 0.9457, 0.9873, and 0.9578, respectively) along with TPR >42% for Meem (م), TPR = 100% for Gol Hy (ه), Hamza (ء), and TNR = 100% for Gol Hy, Hamza, and Meem (ه, ء, م), as shown in Figure 5(c). Overall findings indicate that Braille characters such as ڑ, س, ظ, ه, and ء got the highest AUC value of 1 that shows 100% separation among all classes.

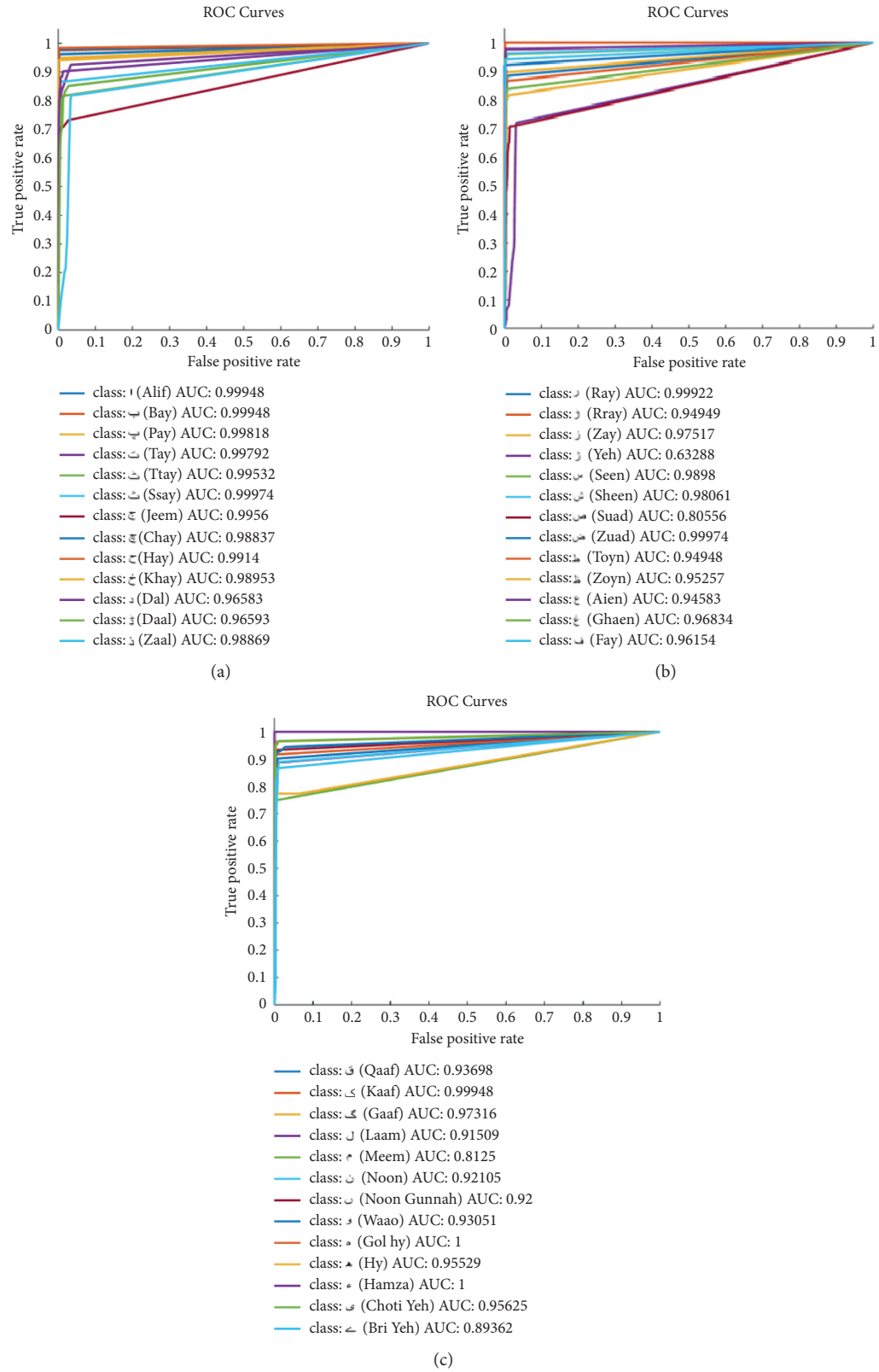

Figure 4: (a) ROC curve for category-1 (ا - ذ) using DT. (b) ROC curve for category-2 (ر - ف) using DT. (c) ROC curve for category-3 (ق - ے) using DT.

Table 1: Decision Tree classifier results for Grade 1 Urdu alphabets.

Serial #	Urdu characters	English equivalent	TPR (%)	TNR (%)	PPV (%)	NPV (%)	FPR (%)	Total accuracy (%)	AUC
1	ا	Alif	100.0	99.90	96.00	100.0	0.10	99.90	0.9995
2	ب	Bay	100.0	99.90	95.24	100.0	0.10	99.90	0.9995
3	پ	Pay	100.0	99.64	84.09	100.0	0.36	99.64	0.9982
4	ت	Tay	100.0	99.58	82.61	100.0	0.42	99.59	0.9979
5	ٹ	Ttay	100.0	99.06	67.86	100.0	0.94	99.08	0.9953
6	ث	Ssay	100.0	99.95	97.50	100.0	0.05	99.95	0.9997
7	ج	Jeem	100.0	99.12	63.04	100.0	0.88	99.13	0.9956
8	چ	Chay	97.78	99.90	95.65	99.95	0.10	99.85	0.9884
9	ح	Hay	98.33	99.95	98.33	99.95	0.05	99.90	0.9914
10	خ	Khay	97.96	99.95	97.96	99.95	0.05	99.90	0.9895
11	د	Dal	93.48	99.69	87.76	99.84	0.31	99.54	0.9658
12	ڈ	Daal	93.55	99.64	80.56	99.90	0.36	99.54	0.9659
13	ذ	Zaal	100.0	97.74	29.03	100.0	2.26	97.76	0.9887
14	ر	Ray	100.0	99.84	94.00	100.0	0.16	99.85	0.9992
15	ڑ	Rray	90.00	99.90	81.82	99.95	0.10	99.85	0.9495
16	ز	Zay	95.56	99.48	81.13	99.90	0.52	99.39	0.9752
17	ژ	Yeh	26.79	99.79	78.95	97.89	0.21	97.71	0.6329
18	س	Seen	97.96	100.0	100.0	99.95	0.00	99.95	0.9898
19	ش	Sheen	96.23	99.90	96.23	99.90	0.10	99.80	0.9806
20	ص	Suad	61.11	100.0	100.0	98.91	0.00	98.93	0.8056
21	ض	Zuad	100.0	99.95	98.33	100.0	0.05	99.95	0.9997
22	ط	Toyn	90.00	99.90	95.74	99.74	0.10	99.64	0.9495
23	ظ	Zoyn	90.57	99.95	97.96	99.74	0.05	99.69	0.9526
24	ع	Aien	89.58	99.58	84.31	99.74	0.42	99.34	0.9458
25	غ	Ghaen	93.88	99.79	92.00	99.84	0.21	99.64	0.9683
26	ف	Fay	92.31	100.0	100.0	99.79	0.00	99.80	0.9615
27	ق	Qaaf	87.50	99.90	95.45	99.69	0.10	99.59	0.9370
28	ك	Kaaf	100.0	99.90	96.49	100.0	0.10	99.90	0.9995
29	گ	Gaaf	94.74	99.90	96.43	99.84	0.10	99.75	0.9732
30	ل	Laam	83.02	100.0	100.0	99.53	0.00	99.54	0.9151
31	م	Meem	62.50	100.0	100.0	99.85	0.00	99.85	0.8125
32	ن	Noon	84.21	100.0	100.0	99.69	0.00	99.69	0.9211
33	ں	Noon Gunnah	84.00	100.0	100.0	99.58	0.00	99.59	0.9200
34	و	Waoo	86.21	99.90	96.15	99.58	0.10	99.49	0.9305
35	ہ	Gol Ha	100.0	100.0	100.0	100.0	0.00	100.0	1.0000
36	ھ	Hy	91.11	99.95	97.62	99.79	0.05	99.75	0.9553
37	ء	Hamza	100.0	100.0	100.0	100.0	0.00	100.0	1.0000
38	ی	Choti Yeh	91.30	99.95	98.82	99.57	0.05	99.54	0.9563
39	ے	Bri Yeh	78.72	100.0	100.0	99.48	0.00	99.49	0.8936

We also achieved above 0.99 AUC for several URDU Braille characters such as پ ,ب ,ا, and ت, etc., by using KNN on the extracted feature set. KNN also exhibits promising results when using RICA features to extract Braille characters from a touchscreen device. Table 2 shows detailed DT results for all Grade-1 Urdu Braille characters.

Furthermore, SVM was used to measure the performance. Category (1–13), Alif-Zaal (ا - ذ), shows the highest accuracies for class 10, 6, 9, 8, 3, 4, and 7, i.e., Jeem, Tay, Pay, Chy, Hay, Ssay, and Khay (خ,ث,ح,چ,پ,ت,ج) are with TA (100%, 99.95%, 99.95%, 99.90%, 99.60%, 99.60%, and 99.80%), with AUC (1, 0.9997, 0.9997, 0.9995, 0.9995, 0.9995, and 0.9990, respectively) with TPR = 100% and TNR >99%, as shown in Figure 6(a). For category-2 (14–26), (Ray-Fay) (ر - ف), highest performance was achieved for class 26, 15, 23, 25, 14, 19, and 21, i.e., Ghaen, Ray, Fay, Rray, Zoyn, Sheen, and Zuad (غ, ر, ف, ڑ, ظ, ش, ض) with TA achieved (100%, 99.95%, 99.90%, 99.90%, 99.85%, 99.85%, and 99.85%) and AUC (1, 0.9444, 0.9828, 0.9905, 0.9992, 0.9900, and 0.9776, respectively), as shown in Figure 6(b). Similarly, for category-3, (Qaaf-Bri Yeh) (ق - ے), highest TA is achieved for class 34, 35, 36.37, 38, 28, 30, and 33, i.e., Waao, Hy, Gol Ha, Hamza, Choti Yeh (و, ھ, ہ, ء, ی) all achieved highest TA of (100%), with AUC (1) with TPR = 100% and TNR = 100%.

However, Kaaf, Laam, and Noon Gunnah (ك, ل, ں) obtained 99.95% TA; AUC for Kaaf and Laam is 0.9997 and 0.9889 for Noon Gunnah with TPR 97% and TNR >99%, as shown in Figure 6(c). SVM outperforms other classifiers, as mentioned above, in terms of AUC value. Braille characters such as ء, ھ, ہ, و, ف, خ, and ی got the highest AUC value of 1. SVM shows 99% separation for ا, ب, پ, ت, ٹ, ث, چ, ذ, ر, ز, ق, ك, and ل. Figures 7(a)–7(c) illustrate overall performance on all Urdu alphabets from Alif-Bri Yay by measuring PR, TNR, PPV, NPV, TNR, TA, and AUC.

Results of Shokat et al. [33] reported that Naive Bayes, DT, KNN, SVM, sequential model, and GoogLeNet inception model have TA of (96.38%), (97.20%),

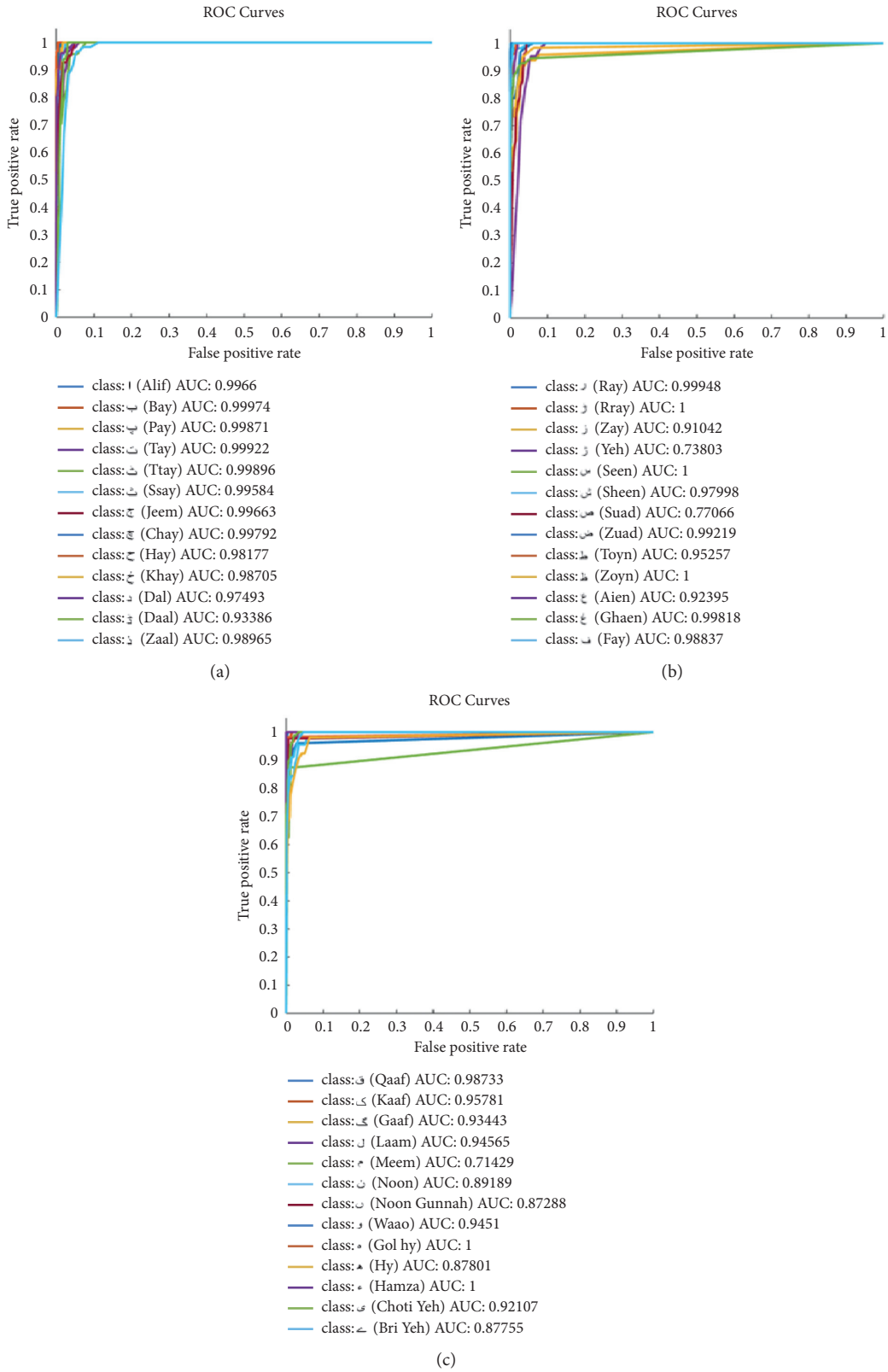

Figure 5: ROC to predict Urdu Braille characters. (a) ROC curve for category-1 (ا - ذ) using KNN. (b) ROC curve for category-2 (ر - ف) using KNN. (c) ROC curve for category-3 (ق - ے) using KNN.

Table 2: KNN classifier results for Grade 1 Urdu alphabets.

Serial #	Urdu characters	English equivalent	TPR (%)	TNR (%)	PPV (%)	NPV (%)	FPR (%)	Total accuracy (%)	AUC
1	ا	Alif	100.0	99.32	80.00	100.0	0.68	99.34	0.9966
2	ب	Bay	100.0	99.95	97.50	100.0	0.05	99.95	0.9997
3	پ	Pay	100.0	99.74	86.11	100.0	0.26	99.75	0.9987
4	ت	Tay	100.0	99.84	93.02	100.0	0.16	99.85	0.9992
5	ٹ	Ttay	100.0	99.79	88.89	100.0	0.21	99.80	0.9990
6	ث	Ssay	100.0	99.17	71.43	100.0	0.83	99.18	0.9958
7	ج	Jeem	100.0	99.33	71.11	100.0	0.67	99.34	0.9966
8	چ	Chay	100.0	99.58	84.62	100.0	0.42	99.59	0.9979
9	ح	Hay	96.72	99.63	89.39	99.89	0.37	99.54	0.9818
10	خ	Khay	97.62	99.79	91.11	99.95	0.21	99.75	0.9871
11	د	Dal	95.45	99.53	82.35	99.90	0.47	99.44	0.9749
12	ڈ	Daal	87.50	99.27	71.43	99.74	0.73	99.03	0.9339
13	ذ	Zaal	100.0	97.93	43.66	100.0	2.07	97.96	0.9896
14	ر	Ray	100.0	99.90	96.00	100.0	0.10	99.90	0.9995
15	ڑ	Rray	100.0	100.0	100.0	100.0	0.00	100.0	1.0000
16	ز	Zay	83.33	98.75	59.32	99.63	1.25	98.42	0.9104
17	ژ	Yeh	48.08	99.53	73.53	98.60	0.47	98.17	0.7380
18	س	Seen	100.0	100.0	100.0	100.0	0.00	100.0	1.0000
19	ش	Sheen	96.15	99.84	94.34	99.90	0.16	99.75	0.9800
20	ص	Suad	54.24	99.89	94.12	98.60	0.11	98.52	0.7707
21	ض	Zuad	98.44	100.0	100.0	99.95	0.00	99.95	0.9922
22	ط	Toyn	90.57	99.95	97.96	99.74	0.05	99.69	0.9526
23	ظ	Zoyn	100.0	100.0	100.0	100.0	0.00	100.0	1.0000
24	ع	Aien	85.42	99.37	77.36	99.63	0.63	99.03	0.9240
25	غ	Ghaen	100.0	99.64	86.00	100.0	0.36	99.64	0.9982
26	ف	Fay	97.67	100.0	100.0	99.95	0.00	99.95	0.9884
27	ق	Qaaf	97.73	99.74	89.58	99.95	0.26	99.69	0.9873
28	ك	Kaaf	91.67	99.90	95.65	99.79	0.10	99.69	0.9578
29	گ	Gaaf	86.89	100.0	100.0	99.58	0.00	99.59	0.9344
30	ل	Laam	89.13	100.0	100.0	99.74	0.00	99.75	0.9457
31	م	Meem	42.86	100.0	100.0	99.80	0.00	99.80	0.7143
32	ن	Noon	78.38	100.0	100.0	99.59	0.00	99.59	0.8919
33	ں	Noon Gunnah	74.58	100.0	100.0	99.22	0.00	99.24	0.8729
34	و	Waoo	89.83	100.0	100.0	99.69	0.00	99.69	0.9492
35	ہ	Gol Ha	100.0	100.0	100.0	100.0	0.00	100.0	1.00
36	ھ	Hy	75.76	99.84	89.29	99.59	0.16	99.44	0.8780
37	ء	Hamza	100.0	100.0	100.0	100.0	0.00	100.0	1.0000
38	ی	Choti Yeh	84.38	99.84	96.43	99.20	0.16	99.08	0.9211
39	ے	Bri Yeh	75.51	100.0	100.0	99.38	0.00	99.39	0.8776

(97.04%), (83.00%), (92.21%), and (95.8%), respectively. For Naive Bayes, DT, and KNN, TPR and PPV showed no significant value. Better results were seen with SVM, sequential model, and GoogLeNet inception model. The maximum performance with the lowest error rate was achieved by SVM with RICA-based feature extraction method with TPR (93.96%), TNR (99.85%), PPV (94.51%), NPV (99.87%), TA (99.73%), and FPR (0.14%). After SVM, the results using DT showed the second best performance. The obtained results show TPR (90.98%), TNR (99.78%), PPV (92.21%), NPV (99.78%), TA (99.57%), and FPR (0.21%). At last, performance achieved using KNN is TPR (90.2%), TNR (99.72%), PPV (89.75%), NPV (99.77%), TA (99.5%), and FPR (0.28%). Table 3 compares the findings of Sana et al. with the current study [33]. More promising results have been achieved with an improved backend processing system for dataset collection; we have achieved more promising results in total accuracy and space used.

4. Discussion

To improve the living quality for visually challenged persons, Braille must be converted to natural language. Braille to natural language conversion has been done in many studies. Most of the studies translated scanned Braille documents into standard English or the other way around. Jha and Parvathi conducted a study that used an SVM classifier trained by extracting features using the histogram of oriented gradient (HOG) feature extraction method to translate handwritten Odia and Hindi text into Braille. Converting Odia [36] and Hindi [25] using this classification technique, 99% and 94.5% accuracies were achieved, respectively, into Braille. Using the same technique, 99% and 80% accuracies were achieved for converting handwritten English and Sinhala documents into Braille text [35]. An SVM classifier trained on HAAR feature extraction methods is used to convert handwritten English sheets, with a classification error of less than 10% [50]. English to Braille conversion was

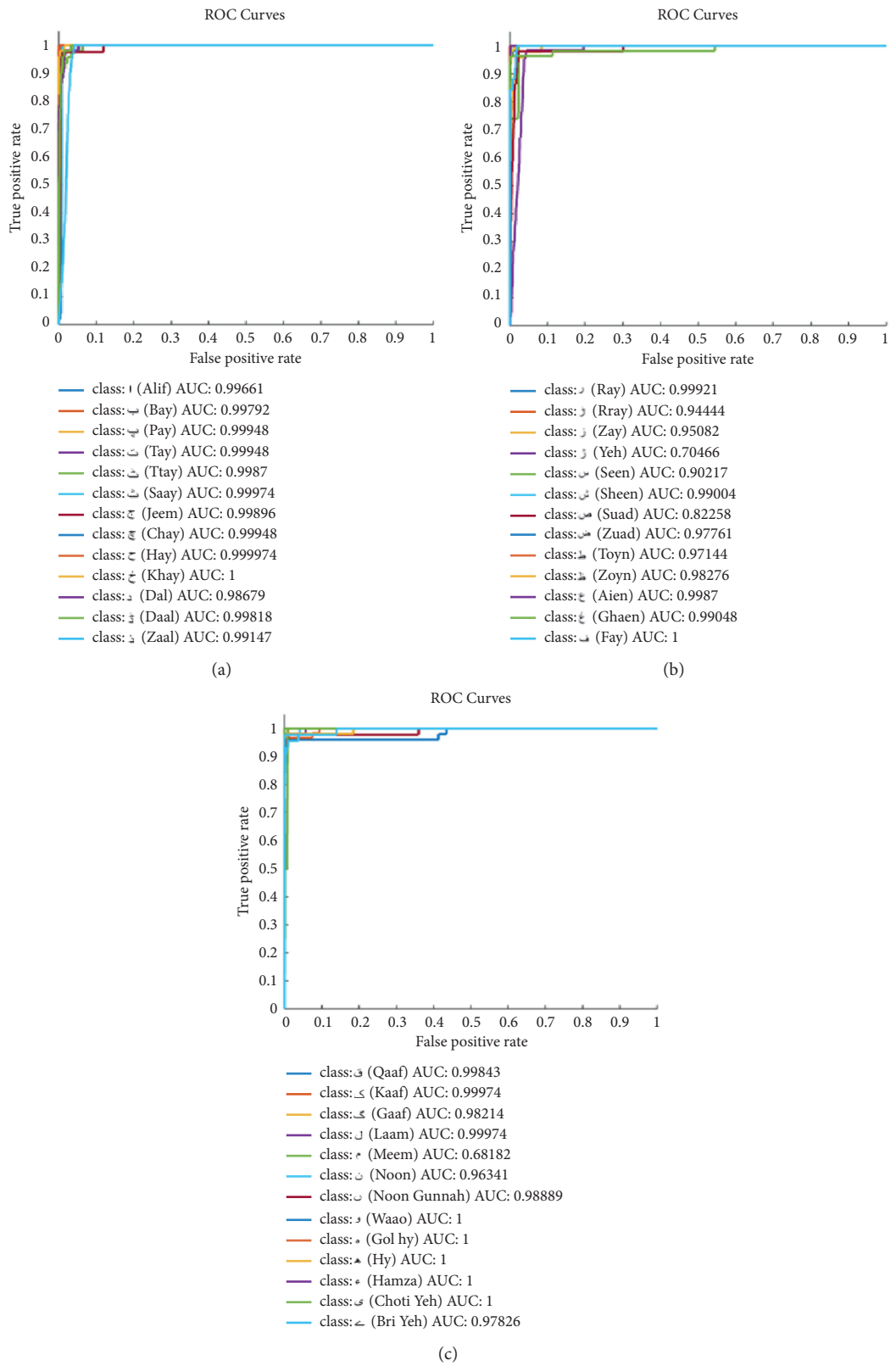

Figure 6: ROC to predict Urdu Braille characters. (a) ROC curve for category-1 (ا - ذ) using SVM. (b) ROC curve for category-2 (ر - ف) using SVM. (c) ROC curve for category-3 (ق - ے) using SVM.

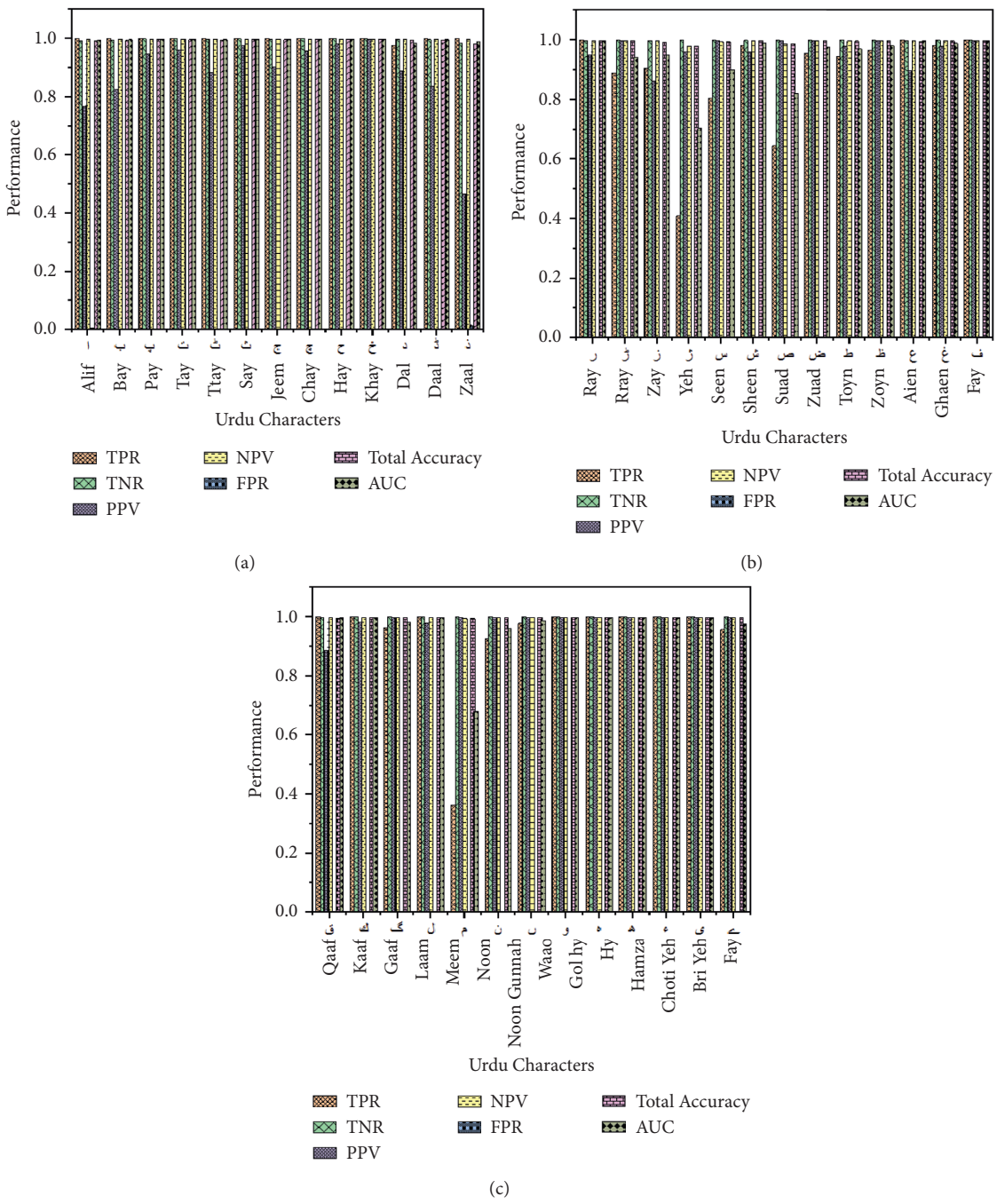

Figure 7: Performance metrics showing results obtained for the SVM classifier. (a) Performance metrics for category-1 (ا - ذ) using SVM. (b) Performance metrics for category-2 (ر - ف) using SVM. (c) Performance metrics for category-3 (ق - ے) using SVM.

taken place by taking input from a gesture-based touchscreen using KNN for classification. The distance between two dots was computed using Bayesian touch distance, which yielded a 97.4 percent accuracy [51]. Another study was carried out to recognize scanned Braille characters using KNN, Naïve Bayes, random forest, and SVM depicting that 63%, 53%, 65%, and 69.6% accuracies were achieved [34], as shown in Table 4.

For Braille to Urdu character recognition, RICA-based features are extracted, and robust machine learning algorithms such as DT, SVM, and KNN are used. A new Urdu Braille dataset was collected from visually impaired students using a newly built touchscreen-based free Braille input application. Main findings achieved for DT algorithm with category-1 (1–13), Alif-Zaal (ا - ذ), the highest detection performance was obtained with Braille class 6 (ث) with TA (99.95%)and AUC (0.9979).These results were followed by other classes such as (ا, ب, ح, خ) yielding accuracies of (99.90%), with AUC (0.9995, 0.9995, 0.9914, and 0.9895), respectively. Other performance indicators, such as TPR and

Table 3: Comparative analysis of Braille to natural language conversion using Naïve Bayes, DT, SVM, KNN, sequential model, and GoogLeNet model.

Classification techniques used	TPR (%)	TNR (%)	PPV (%)	NPV (%)	TA (%)	FPR (%)
Deep learning scheme for character prediction with position-free touchscreen-based Braille input method [33]						
Naïve Bayes	NaN	96.64	NaN	99.38	96.38	3.36
DT	NaN	98.82	NaN	98.23	97.20	1.18
KNN	NaN	98.60	NaN	98.11	97.04	1.40
SVM	68.67	99.16	76.75	98.72	83.00	0.84
Sequential model	90.76	99.63	91.07	99.2	92.21	0.36
GoogLeNet model	95.89	99.83	96.61	99.83	95.8	0.16
Detection of Urdu Braille characters based on reconstruction independent component analysis (RICA) features using robust machine learning techniques						
DT	90.98	99.78	91.21	99.78	99.57	0.22
KNN	90.2	99.72	89.75	99.77	99.5	0.28
SVM	93.96	99.85	94.51	99.87	99.73	0.15

Table 4: Comparative analysis with previous literatures.

Supported languages	Techniques used	Feature extraction/ algorithms/others	Accuracy (%)	References
Hindi	SVM	HOG features	94.5	[24]
Chinese	Image segmentation techniques	Image segmentation techniques	79.63	[52]
Chinese	DNN	DNN	90.47	[53]
Simple Braille recognition	MLP	SDAE	86	[34]
	RBF		80	
	SoftMax		92	
	MLP	Traditional feature extraction	64	
	RBF		55	
	SoftMax		65	
	KNN		63	
	Naïve Bayes		53	
	Random forest		65	
	SVM		69.6	
Tamil	Nil	Image segmentation technique	99.2	[28]
Hindi			98.8	
Sinhala	SVM	HOG feature extraction method	80	[35]
Arabic	Newly designed Braille letter recognition and transcription scheme for Braille to Arabic text conversion	Image segmentation	99	[21]
Odia	SVM	HOG features	99	[36]
Korean	CNN model	------	99.6	[54]
Bangla	Deep neural network	VGG-16	94.62	[55]
		ResNet-50	93.58	
		DenseNet-121	94.08	
Grade 1 Urdu Braille	Decision tree	RICA feature extraction method	99.57	Newly proposed coordinate-based model
	KNN		99.50	
	SVM		99.73	

TNR, show that performance (TPR >94%) and (TNR >98%) are achieved. Similarly, category-2 (14–26), Ray to Fay (ر - ف), for DT shows the best results for class 18, 21, 14, and 15, i.e., Seen, Zuuad, Ray, and Rray (س, ض, ر, ڑ) with accuracies of 99.95%, 99.95%, 99.85%. and 99.85%, with AUC (0.9899, 0.9997, 0.9992, and 0.9495 respectively) with TPR = 100% and TNR >99%. For category-3 (27–39), Qaaf-Bri Yeh (ے - ق), highest TA is achieved for class 35, 37, and 28, i.e., Gol Hy and Hamza (ہ, ء) with (100%) TA. TA of Kaaf is 99.90%, with AUC (1, 1, and 0.9995), respectively, with TPR >94% and TNR >99%. By employing KNN, the highest TA achieved for category (1–13), Alif-Zaal (ا - ذ), is for class 13, 2, and 4, i.e., Bay and Tay (99.85) (ب, ت% and 99.85%), with AUC (0.9997 and 0.9992, respectively). For category-2 (14–26), (Ry-Fy) (ف - ر), highest TA is achieved for class 15, 18, 23, 26, and 21, i.e., Rray, Seen, and Zoyn (ڑ, س, ظ) TA = 100% followed by Fay and Zuad (ف, ض) with TPR (99.95%) with AUC (1, 1, 1, 0.9884, and 0.9922) with TPR = 100% and TNR >99%. Similarly, for category-3 (27–39) (Qaaf-Bri Yeh) (ے - ق), highest TA was achieved for class 35, 37, and 31, i.e., Gol Ha, Hamza, and Meem (ہ, ء, م) TA (100%, 100%, and 99.80%), with AUC (1, 1, and 0.7143)

with TPR >42% for Meem (م) and TPR = 100% for Gol Hy (ہ) and Hamza (ء) and TNR = 100% for Hay, Hamza, and Meem (ء,ہ م,). Moreover, performance evaluation for SVM category (1–13), Alif-Zaal (ا - ذ), shows highest accuracies for class 10, 6, 9, and 8, i.e., Khay, Ssay, Hay, and Chy (خ, ث, ح, چ) with TA (100%, 99.95%, 99.95%, and 99.90%), with AUC (1, 0.9997, 0.9997, and 0.9995, respectively) with TPR = 100% and TNR >99%. For category-2 (14–26) (Ray-Fay) (ر - ف), highest performance was achieved for class 26, 15, 23, and 25, i.e., Fay, Rray, Zoyn, and Ghaen (ف ڑ, غ, ظ) with TA achieved (100%, 99.95%, 99.90%, and 99.90%) with AUC (1, 0.9444, 0.9828, and 0.9905). Similarly, for category-3 (Qaaf-Bri Yeh) (ق - ے), the highest TA is achieved for class 34, 35, 36.37, and 38, i.e., Waao, Hy, Gol Hay, Hamza, and Choti Yeh (ی, ء, ہ, و) all achieved the highest TA of (100%), with AUC (1) with TPR = 100% and TNR = 100%.

5. Conclusions and Future Work

Braille is a growing means of communication for people with visual impairments. More than 150 million people continue to use Braille around the world for several reasons. Literacy is one of the powerful cases that show Braille's importance for learning how to read and write. With the advent of technology, Braille is more accessible to visually impaired users. Various studies have been carried out to convert Braille into a natural language. However, most studies have used handwritten scanned sheets to translate Braille into natural languages such as Arabic, Hindi, Tamil, Odia, Chinese, Korean, Bangla, English, and Gujarati, and vice versa [21, 28, 36, 52–57]. As per our knowledge for Urdu Braille recognition, no such study has been found so far. In this research, the Urdu Braille dataset is collected using a touchscreen-based android application [33]. This application is easy to use and less burdensome for visually impaired people. Robust machine learning techniques include SVM with polynomial kernels, DT with default parameters, and KNN with $K = 3$ for Urdu Braille character recognition. The dataset was divided into 70%–30% for training and validation. Performance metrics used to evaluate these classifiers' performance are PPV, NPV, FPR, FNR, FPR, TA, and AUC. The RICA-based feature extraction method is used for Urdu Braille character recognition. The highest performance using DT classifier shows that class 35, Gol Hy (ہ), and class 37, Hamza (ء), achieved a maximum separation AUC (1) with TA, TPR, TNR = 100%, and FPR = 0%. Similarly, the highest performance using KNN shows that class 15, 18, 23, 35, and 37, i.e., Rray, Seen, Zoyn, Gol Hy, and Hamza (ڑ, س, ظ, ہ, ء) attained the highest separation AUC (1) with TPR, TNR, TA = 100%, and FPR = 0%. Furthermore, using SVM, the highest performance was achieved by class 10, 26, 34, 35, 36, 37, and 38, i.e., Khay, Fay, Waao, Hy, Gol Hy, Hamza, and Choti Yeh (خ, ف, و, ح, ہ, ء, ی) with highest separation AUC (1) with TPR, TNR, TA = 100%, and FPR = 0%. SVM has the highest TA of all classifiers with TA (99.73%), TPR (93.96%), TNR (99.85%), PPV (94.51%), NPV (99.87%), and FPR (0.14%).

RICA features are extracted, and robust machine learning techniques are used to recognize Urdu Braille characters. We intend to expand this dataset in the future to include other languages such as English and maths with advanced Braille levels. The performance will then be evaluated using convolutional neural network (CNN) and transfer learning techniques such as GoogleNet inception model and LSTM. By implementing these models, performance of the system will further improve. Along with this, we will be more focused on providing error detection and voice feedback services for visually impaired users.

Data Availability

This dataset is not publicly available. But it could be provided on request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korean government under reference number (2020R1A2C1012196).

References

[1] Y. Lee and J. Lee, "A checklist for assessing blind users' usability of educational smartphone applications," *Universal Access in the Information Society*, vol. 18, no. 2, pp. 343–360, 2019.

[2] Tech. rep, "Ericsson Mobility Report," 2019. https://www.ericsson.com/en/mobility-report/reports/november-2019.

[3] S. O'Dea, "Smartphone users 2020 | Statista," Smartphone users worldwide 2016-2021. 2020. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide-2020.

[4] W. Grusseneyer and E. Folmer, "Accessible touchscreen technology for people with visual impairments: a survey," *ACM Trans. Access. Comput.*vol. 9, no. 2, pp. 1–31, 2017.

[5] S. Akram, A. Mahmood, I. Ullah et al., "Construction and analysis of a novel wearable assistive device for a visually impaired person," *Applied Bionics and Biomechanics*, vol. 2020, Article ID 6153128, 11 pages, 2020.

[6] A. A. Díaz-toro, S. E. Campaña-bastidas, and E. F. Caicedo-bravo, "Vision-based system for assisting blind people to wander unknown environments in a safe way," *Journal of Sensors*, vol. 2021, Article ID 6685686, 18 pages, 2021.

[7] V. Iyer, K. Shah, S. Sheth, and K. Devadkar, "Virtual assistant for the visually impaired," in *Proceedings of the 2020 5th International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, India, December 2020.

[8] K. Taylor and L. Silver, *Smartphone Ownership Is Growing Rapidly Around the World, but Not Always Equally*, Pew Research Center, Washington, DC, USA, 2019, https://Www.Pewresearch.Org/Global/2019/02/05/Smartphone-Ownership-Is-Growing-Rapidly-Around-the-World-But-Not-Always-Equally/.

[9] S. Shokat, R. Riaz, S. S. Rizvi, K. Khan, F. Riaz, and S. J. Kwon, "Analysis and evaluation of braille to text conversion methods," *Mobile Information Systems*, vol. 2020, Article ID 3461651, 13 pages, 2020.

[10] J. Rantala, R. Raisamo, J. Lylykangas et al., "Methods for presenting braille characters on a mobile device with a touchscreen and tactile feedback," *IEEE Transactions on Haptics*, vol. 2, no. 1, pp. 28–39, 2009.

[11] R. K. Virinchi, "Design and analysis of a system to read braille with remote cloud architecture," in *Proceedings of the IEEE 17th India Council International Conference (INDICON)*, New Delhi, India, September 2020.

[12] B. Leporini, M. C. Buzzi, and M. Buzzi, "Interacting with mobile devices via VoiceOver," in *Proceedings of the 24th Australian Computer-Human Interaction Conference on-OzCHI '12*, Melbourne, Australia, May 2012.

[13] M. Li, M. Fan, and K. N. Truong, "BrailleSketch," in *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility*, Baltimore, ML, USA, June 2017.

[14] T. Guerreiro, P. Lagoá, P. Santana, D. Gonçalves, and J. Jorge, "NavTap and BrailleTap: non-visual texting interfaces," in *Proceedings of the Rehabilitation Engineering and Assistive Technology Society of North America Conference (Resna)*, Baltimore, ML, USA, January 2008.

[15] J. Oliveira, T. Guerreiro, H. Nicolau, J. Jorge, and D. Gonçalves, "BrailleType: unleashing braille over touch screen mobile phones," in *Proceedings of the IFIP Conference on Human-Computer Interaction*, Lisbon, Portugal, December 2011.

[16] N. S. Subash, S. Nambiar, and V. Kumar, "BrailleKey: an alternative Braille text input system: comparative study of an innovative simplified text input system for the visually impaired," in *Proceedings of the 4th International Conference on Intelligent Human Computer Interaction: Advancing Technology for Humanity, IHCI 2012*, Kharagpur, India, June 2012.

[17] S. Azenkot, J. O. Wobbrock, S. Prasain, and R. E. Ladner, "Input finger detection for nonvisual touch screen text entry in Perkinput," in *Proceedings of the Graphics Interface*, Toronto Ontario Canada, May 2012.

[18] B. Šepić, A. Ghanem, and S. Vogel, "BrailleEasy: one-handed braille keyboard for smartphones," *Studies in Health Technology and Informatics*, vol. 217, pp. 1030–1035, 2015.

[19] M. Shabnam and S. Govindarajan, "Gesture recognition algorithm: braille-coded gesture patterns for touch screens: eyedroid," *Indian J. Sci. Technol.*vol. 9, no. 33, pp. 1–17, 2016.

[20] H. Gautam and P. Gaur, "DRISHYAM_ real-time text to braille conversion and realisation," in *Proceedings of the IEEE 17th India Council International Conference (INDICON)*, New Delhi, India, May 2020.

[21] S. D. Al-Shamma and S. Fathi, "Arabic Braille Recognition and transcription into text and voice," in *Proceedings of the 2010 5th Cairo International Biomedical Engineering Conference*, Cairo, Egypt, June 2010.

[22] P. R. Kumar, "Braille language converter for visually impaired people," *Int. J. Intellect. Adv. Res. Eng. Comput.*vol. 6, no. 2, pp. 2229–2232, 2018.

[23] L. Nahar, A. Jaafar, E. Ahamed, and A. B. M. A. Kaish, "Design of a braille learning application for visually impaired students in Bangladesh," *Assistive Technology*, vol. 27, no. 3, pp. 172–182, 2015.

[24] V. Jha and K. Parvathi, "Braille transliteration of Hindi handwritten texts using machine learning for character recognition," *Int. J. Sci. Technol. Research*, vol. 8, no. 10, pp. 1188–1193, 2019.

[25] A. Bier and Z. Sroczyński, "Rule based intelligent system verbalizing mathematical notation," *Multimedia Tools and Applications*, vol. 78, no. 19, pp. 28089–28110, 2019.

[26] M. Maćkowski, P. Brzoza, M. Żabka, and D. Spinczyk, "Multimedia platform for mathematics' interactive learning accessible to blind people," *Multimedia Tools and Applications*, vol. 77, no. 5, pp. 6191–6208, 2018.

[27] J. Yook, K. Kim, B. C. Son, and S. Park, "A translating program usability analysis of alternative multimedia mathematics materials for the blind," *Multimed. Tools Appl.*-Springer, Berlin, Germany, 2020.

[28] S. Padmavathi, K. S. S. Manojna, and S. S. Reddy, "Conversion of braille to text in English, Hindi and Tamil languages," *International Journal of Computer Science, Engineering and Applications*, vol. 3, no. 3, pp. 19–32, 2013.

[29] M. A. Fahiem, "A deterministic turing machine for context sensitive translation of braille codes to Urdu text," in *Combinatorial Image Analysis*Springer, Berlin Heidelberg, 2008.

[30] U. Beg, K. Parvathi, and V. Jha, "Text translation of scanned Hindi document to braille via image processing," *Indian Journal of Science and Technology*, vol. 10, no. 33, pp. 1–8, 2017.

[31] G. Kouroupetroglou, A. Pino, and P. Riga, "A methodological approach for designing and developing web-based inventories of mobile Assistive Technology applications," *Multimedia Tools and Applications*, vol. 76, no. 4, pp. 5347–5366, 2017.

[32] S. Priyadarsini, P. Ajit, K. Nayak, and S. Champati, "A survey on speech synthesis techniques in Indian languages," *Multimedia Systems*, vol. 26, pp. 453–468, 2020.

[33] S. Shokat, R. Riaz, S. S. Rizvi, A. M. Abbasi, and S. J. Kwon, "Deep learning scheme for charcater prediction with position-free touch screen-based Braille input method," *Human Centric Computing and Information Sciences*, vol. 10, no. 1, pp. 1–24, 2020.

[34] T. Li, X. Zeng, and S. Xu, "A Deep learning method for braille recognition," in *Proceedings of the 2014 International Conference on Computational Intelligence and Communication Networks*, Bhopal, India, November 2014.

[35] W. K. I. L. I. Perera and T. D. S. H. Wanniarachchi, "Optical braille recognition based on Histogram of oriented gradient features and support-vector machine," *International Journal of Engineering Science*, vol. 8, no. 10, pp. 19192–19195, 2018.

[36] V. Jha and K. Parvathi, "Machine learning based braille transliteration of Odia language," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 5, pp. 1866–1871, 2020.

[37] Y. Yanhui Xiao, Z. Zhenfeng Zhu, Y. Yao Zhao, Y. Yunchao Wei, and S. Shikui Wei, "Kernel reconstruction ICA for sparse representation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 6, pp. 1222–1232, Jun. 2015.

[38] A. Hyvärinen, J. Hurri, and P. O. Hoyer, *Natural Image Statistics*, Springer London, London, UK, 2009.

[39] Q. Le, A. Karpenko, J. Ngiam, and A. Y. Ng, "ICA with reconstruction cost for efficient overcomplete feature learning," in *Advances in Neural Information Processing Systems*Springer, Berlin, Germany, 2011.

[40] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Haifa, Israel, September 2010.

[41] Y. LeCun, ""Learning invariant feature hierarchies," in learning invariant feature hierarchies," in *Proceedings of the European Conference on Computer Vision*, Berlin, Heidelberg, June 2012.

[42] M. Hu, W. Li, K. Yan, Z. Ji, and H. Hu, "Modern machine learning techniques for univariate tunnel settlement

forecasting: a comparative study," *Mathematical Problems in Engineering*, vol. 2019, Article ID 7057612, 12 pages, 2019.

[43] I. Rasheed, V. Gupta, H. Banka, and C. Kumar, "Urdu Text Classification: a comparative study using machine learning techniques," in *Proceedings of the 2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, Coimbatore, India, May 2018.

[44] T. Shaikhina, D. Lowe, S. Daga, D. Briggs, R. Higgins, and N. Khovanova, "Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation," *Biomedical Signal Processing and Control*, vol. 52, pp. 456–462, 2019.

[45] L. Hussain, I. A. Awan, W. Aziz et al., "Detecting congestive heart failure by extracting multimodal features and employing machine learning techniques," *BioMed Research International*, vol. 2020, Article ID 4281243, 19 pages, 2020.

[46] Q. Ning, Z. Ma, and X. Zhao, "dForml(KNN)-PseAAC: detecting formylation sites from protein sequences using K-nearest neighbor algorithm via Chou's 5-step rule and pseudo components," *Journal of Theoretical Biology*, vol. 470, pp. 43–49, 2019.

[47] A. Gammerman, V. Vovk, H. Boström, and L. Carlsson, "Conformal and probabilistic prediction with applications: editorial," *Machine Learning*, vol. 108, no. 3, pp. 379-380, 2019.

[48] M. Achirul Nanda, K. Boro Seminar, D. Nandika, and A. Maddu, "A comparison study of kernel functions in the support vector machine and its application for termite detection," *Information*, vol. 9, no. 1, p. 5, 2018.

[49] A. Tharwat, "Parameter investigation of support vector machine classifier with kernel functions," *Knowledge and Information Systems*, vol. 61, no. 3, pp. 1269–1302, 2019.

[50] J. Li, X. Yan, and D. Zhang, "Optical braille recognition with haar wavelet features and support-vector machine," in *Proceedings of the International Conference on Computer, Mechatronics, Control and Electronic Engineering*, Changchun, China, December 2010.

[51] H. Udapola and S. R. Liyanage, "Braille messenger: adaptive learning based non-visual touch screen input for the blind community using braille," in *Proceedings of the International Conference on Innovations in Info-Business and Technology, 2017*, Ozo, Colombo, Sri Lanka, October 2018.

[52] X. Wang, Y. Yang, H. Liu, and Y. Qian, "Chinese-braille translation based on braille corpus," *International Journal of Advanced Pervasive and Ubiquitous Computing*, vol. 8, no. 2, pp. 56–63, 2016.

[53] X. Wang, J. Zhong, J. Cai, H. Liu, and Y. Qian, "CBConv: service for automatic conversion of Chinese characters into braille with high accuracy," in *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility*, New York, NY, USA, April 2019.

[54] S. Lee, S. Jung, and H. Song, "CNN-based drug recognition and braille embosser system for the blind," *Journal of Computing Science and Engineering*, vol. 12, no. 4, pp. 149–156, 2018.

[55] T. R. Abir, T. S. Bin Ahmed, M. T. Rahman, and S. Jafreen, *Handwritten Bangla Character Recognition to Braille Pattern Conversion Using Image Processing and Machine Learning*, BRAC University, Dhaka, Bangladesh, 2018.

[56] M. Gadag and V. Udayashankara, "Efficient approach for English braille to text conversion," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 5, no. 4, pp. 3343–3348, 2016.

[57] H. Parekh, S. Shah, F. Patel, and H. Vyas, "Gujarati braille text recognition: a review," *International Journal of Computer Science & Communication*, vol. 7, no. 1, pp. 19–24, 2019.