Human-centric Computing and Information Sciences

RESEARCH

Open Access

Deep learning scheme for character prediction with position-free touch screen-based Braille input method

Sana Shokat[1], Rabia Riaz[1], Sanam Shahla Rizvi[2], Abdul Majid Abbasi[1], Adeel Ahmed Abbasi[1] and Se Jin Kwon[3*]

*Correspondence: sjkwon@kangwon.ac.kr
[3] Dept. of Computer Engineering, Kangwon National University, 346 Joongang-ro, 25913 Samcheok, Gangwon-do, South Korea
Full list of author information is available at the end of the article

Abstract

Smart devices are effective in helping people with impairments, overcome their disabilities, and improve their living standards. Braille is a popular method used for communication by visually impaired people. Touch screen smart devices can be used to take Braille input and instantaneously convert it into a natural language. Most of these schemes require location-specific input that is difficult for visually impaired users. In this study, a position-free accessible touchscreen-based Braille input algorithm is designed and implemented for visually impaired people. It aims to place the least burden on the user, who is only required to tap those dots that are needed for a specific character. The user has input English Braille Grade 1 data (a–z) using a newly designed application. A total dataset comprised of 1258 images was collected. The classification was performed using deep learning techniques, out of which 70%–30% was used for training and validation purposes. The proposed method was thoroughly evaluated on a dataset collected from visually impaired people using Deep Learning (DL) techniques. The results obtained from deep learning techniques are compared with classical machine learning techniques like Naïve Bayes (NB), Decision Trees (DT), SVM, and KNN. We divided the multi-class into two categories, i.e., Category-A (a–m) and Category-B (n–z). The performance was evaluated using Sensitivity, Specificity, Positive Predicted Value (PPV), Negative Predicted Value (NPV), False Positive Rate (FPV), Total Accuracy (TA), and Area under the Curve (AUC). GoogLeNet Model, followed by the Sequential model, SVM, DT, KNN, and NB achieved the highest performance. The results prove that the proposed Braille input method for touch screen devices is more effective and that the deep learning method can predict the user's input with high accuracy.

Keywords: Braille, Machine learning, Natural language processing, Deep learning, Touch screen, Convolutional neural network, Transfer learning, GoogLe net, Visually impaired

Introduction

The term "visually impaired" is used for people with no vision or non-recoverable low vision. According to the data, 89% of visually impaired people belong to less developed countries, and 55% of them are women [1]. Braille is a language specifically designed for the visually impaired and is composed of combinations of six-dot patterns. Symbols corresponding to sounds in natural language are represented by activating and deactivating these dots [2].

Initially, Braille was read and written on paper or slates with raised dots. The Perkin Brailler, a Braille type-writer, was introduced for visually impaired people in 1951 [3]. With the advent of technology, touch-screen devices have become available with applications such as talkback services, screen readers, navigators, smart magnifiers [4], and obstacle detection and avoidance system for visually impaired people [5] [6]. It allows people to communicate easily anytime all over the world [7]. These helped in improving conversational technology and natural language communication for educational purpose [8]. Most visually impaired people are hesitant to use smartphones because of accessibility and usability issues, such as difficulty in finding the location of keys on smartphones. Furthermore, whenever visually impaired people perform a task, they require feedback regarding its outcome [9].

Various studies have been conducted to convert Braille into a natural language. Applications such as BeMyEyes [10], BeSpecular [11], TapTapSee [12], KNFBReader [13], Color Teller [14], Text to Speech Synthesizer [15–17] have been specifically developed to improve the quality of life for people with visual impairments. Most of these take input from scanned Braille documents using computational tools. Scanning means the process consists of two steps; write and then scan the document. However, if the input is taken directly from touch screen devices, the conversion can be performed instantaneously [18]. Currently, only a few studies have investigated touchscreen-based Braille input, and most of these schemes require screen-location specific input, which is difficult for visually impaired users [19]. Besides, no studies have collected the Braille character dataset using Android devices.

Traditional machine learning models have been extensively used in numerous research domains. However, these techniques require in-depth knowledge of the domain and use pre-defined features for evaluation. Extensive manual feature extraction is also required in these models. Deep Learning (DL) techniques become popular because they learn features directly from the data without pre-defined features, and their performance is not dependent on the size of the dataset.

This study focuses on the design, implementation, and evaluation of a new touchscreen-based Braille input method. An accessible Braille input method that places little burden on the user is proposed for the visually impaired. Users are only required to tap the dots needed for the specific character anywhere on the screen. The user's input is predicted using a DL method, which automatically extracts features from saved images to identify the Braille character and translates them into the corresponding English character. To the best of our knowledge, this is the first study to use DL for Braille-to-natural language conversion by classifying Braille images collected from touch screen devices.

The main contributions of this study are as follows:

A position-free accessible touchscreen-based Braille input method for the visually impaired is proposed to put the least burden on the user.

- A Grade 1 Braille dataset was collected from visually impaired students from the Special Education School, Manak Payyan, Pakistan because there is no existing Braille dataset on which DL techniques can be applied.
- A DL model is proposed to classify Braille images and match them with specific English alphabetic characters.
- Evaluation of the performance of an actual dataset collected using Android devices from visually impaired people is presented.
- Comparison with traditional methods like NB, SVM, KNN, and DT is made.

In the rest of this paper, Sect. 2 briefly covers the background and related work. Section 3 materials and methods describe the new accessible Braille input method and its important features. Section 4 provides the experimental setting for evaluation and Sect. 5 presents the results in detail. The conclusions and future directions are given in Sect. 6.

Background and related work

Braille input mechanism

Several input methods have been designed for entering Braille using touch screens, e.g., Type In Braille [20], Edge Braille [21], VBraille [22], Perkinput [23], and Braille Easy [9]. Small screen sizes, difficulty in finding specific locations on the screen [24], the lack of physical keys, and a lack of adequate input are some factors that make the use of these applications impractical for visually challenged people. Accessibility is a major challenge faced by visually impaired people when using these software applications. Although many researchers have focused on making touch screens viable for the visually impaired, it is still a cumbersome process [25–27].

Braille dots are entered using three rows and two columns format in TypeIn Braille [20]. The user enters the data by tapping on the screen, but tapping with both fingers simultaneously for entering two consecutive dots is not viable for the visually impaired. In Braille Touch, visually impaired people have to use both hands and multiple fingers to enter characters using a fixed position [23]. A comparative study conducted by Subash specifies four large buttons on the screen and tapping gestures such as single tap, double-tap, and triple tap are used to enter Braille dots [28]. Similarly, Braille Easy uses single, double, and triple taps to enter dots, but memorizing the reference points is difficult [9].

From the literature, it is clear that the most important problem faced by the visually impaired when entering data on touch screen smart devices is finding the position of the dots on the screen. This problem was resolved by Shabnam, who designed Eyedroid [29]. Braille dots are entered by activating and deactivating dots anywhere on the screen, and visually impaired people do not need to find the fixed location of the dots. Edge Braille is another position-free Braille entry method that requires a continuous pattern to be drawn along the edges of the screen [21].

Navigation is also an important factor for the usability analysis of Braille input mechanisms. VBraille navigation is performed by swiping up and down, as well as from left to right [22]. Difficulty in memorizing the gestures for navigation purposes and switching among various writing modes are the major challenges faced by the users of TypeIn

Braille [20]. In SingleTap Braille, operations are performed by flipping left, right, up, and down. Learning letters requires a user to press "1" [30]. Improved SingleTap Braille uses different swipe gestures for inserting upper-case letters, lower-case letters, adding spaces, backspaces, and Grade 2 Braille [31]. Braille Tap introduced four different gestures to perform arithmetic operations: swiping from bottom to top for addition and subtraction, swiping from top to bottom for division and multiplication, swiping from left to right for clear operation, and swiping from right to left for backspace [32]. Braille Enter was introduced to remove the navigation issues of single-tap Braille. Braille Enter supports upper and lower-case letters, numbers, and special characters. The addition or deletion of spaces is also allowed [27]. Karmal et al. designed an application for blind, deaf, and dumb people that takes images, audio, and text through the customized keyboard as an input. Feedback to the user is provided in the form of voice and text [33] [34].

Some applications provided Tactile based object recognition [35] and audio feedback. Audio feedback poses privacy issues in some scenarios [36]. Voice feedback is provided to inform the user of the task's completion in [20, 25, 31, 32, 37]. On completion of the task, vibrio-tactile feedback is provided by [20] and [21]. An educational application named mBraille was designed for learning English and Bengali [38]. Braille-to-text conversion is also performed in other languages such as Hindi, Tamil, Urdu, Odia, Arabic, Chinese, Bengali, Telugu, Malayalam, and Kannada [18, 39–45].

Braille-to-natural language conversion using deep learning

DL techniques provide better results for image and pattern recognition compared to other machine learning techniques [46–48]. A deep learning method for Braille data recognition has been used by Li et al. [49]. In their scheme, data was collected in the form of Braille images taken with a digital camera from a Braille book. They designed Stacked Denoising Auto Encoder (SDAE) for reducing the feature extraction and dimension reduction problem in Braille character recognition. They achieved 92% accuracy when SDAE was used with the Softmax classifier. This encoder works better than traditional methods for automatic feature extraction from images. An improved Optical Braille recognition system was designed by Vishwanath et al. that increases the picture quality of the scanned images [50]. To improve the quality of scanned Braille images Vishwanath et al. presented an improved optical Braille recognition system.

In a study carried out by Jha and Parvathi, Hindi and Odia text was converted into Braille using scanned documents. SVM robust machine learning technique was applied using the Histogram of Oriented Gradient (HOG) feature extraction method. Accuracies achieved for converting Hindi [51], and Odia [52] to Braille was 94.5% and 99% respectively. The same techniques were applied to convert English and Sinhala scanned documents into Braille. The results revealed that 99% and 80% accuracy was achieved for converting into Braille [53]. Another study carried out by Li. et al. using the same classifier as used in earlier studies. This time HAAR feature extraction method is used that showed a 10% classification error reduction for converting handwritten English sheets to Braille [54]. Using the KNN technique input taken from gesture-based touch screen English text was converted into Braille. The distance between two dots was calculated using Bayesian Touch Distance and 97.4% accuracy was attained [55]. Ahnaf et al.

designed a dynamic Braille recognition system using the YOLO algorithm that identifies an image captured from the camera and converts the information into Braille text. They achieved 87% accuracy for detecting an apple, 86% for recognizing a bottle, and 67% for recognizing a book [56].

From an extensive review of the current literature, it was found that the most common problems that the visually impaired face when using Braille on touch-screens are the use of both hands, multi-finger touch, multiple single-entry touch, gesture memorization, deficient feedback, privacy issues, and location-specific data entry.

Therefore, in this study, a position-free text entry method for Braille is presented that reduces the problems found in the methods discussed earlier. In the proposed method, the user is only required to tap the active dots for a specific character and is liberated from the need to enter unnecessary dots. This results in a reduction in training time as well as user memorization requirements. In the past, the researchers computed hand-crafted features using machine learning (non-deep learning) techniques, but their performance is based on the type and relevance of the features extracted. In this study, a new dataset is generated using this application and we have proposed the deep learning GoogleNet with a transfer learning approach. The deep learning methods being fine-tuned do not depend on the handcrafted features and their activation functions (ReLu) and convolution and pooling layers help for optimization and minimization of error to reach the goal. For further analysis classical machine learning like NB, DT, SVM, and KNN are applied.

Materials and methods

In this section, the DL model is used for text input character prediction. In the following subsections, the proposed Braille input interface use and character input methods are illustrated. Then, a DL-based classification model that predicts a user's input character is thoroughly discussed. The proposed framework for Braille to text character prediction from Braille input using the DL model is shown in Fig. 1.

Dataset collection

Presently, there is no existing digital dataset for Braille. For dataset collection, a new touch-screen interface for Android devices was developed. Braille dataset was collected using touch screen devices from the Special Education School, Manak Payyan, Pakistan. The average age of the participants in the study was 19 years. Initially, English letters (a–z) were collected as Braille input from 24 students who were partially or fully visually impaired. Each Braille input was saved as an image that is comprised of 64×64 pixels because this image size retains the important features of the image and reduces the computation time [57]. If any input image was greater than this size, then that image was resized and saved accordingly, as shown in Fig. 1. Inconsistencies from the data were removed manually by matching the patterns of the Braille dots on paper. Unmatched characters were removed from the dataset. There are 1284 images in the final dataset arranged in alphabetical order, as shown in Fig. 2. The training was performed using the 858 images of Braille Dataset, and validation was performed using 390 images. Choosing accurate, hyperparameter combinations such as optimizers, batch size, learning rate, accuracy and loss is the most challenging task when using CNN. If these

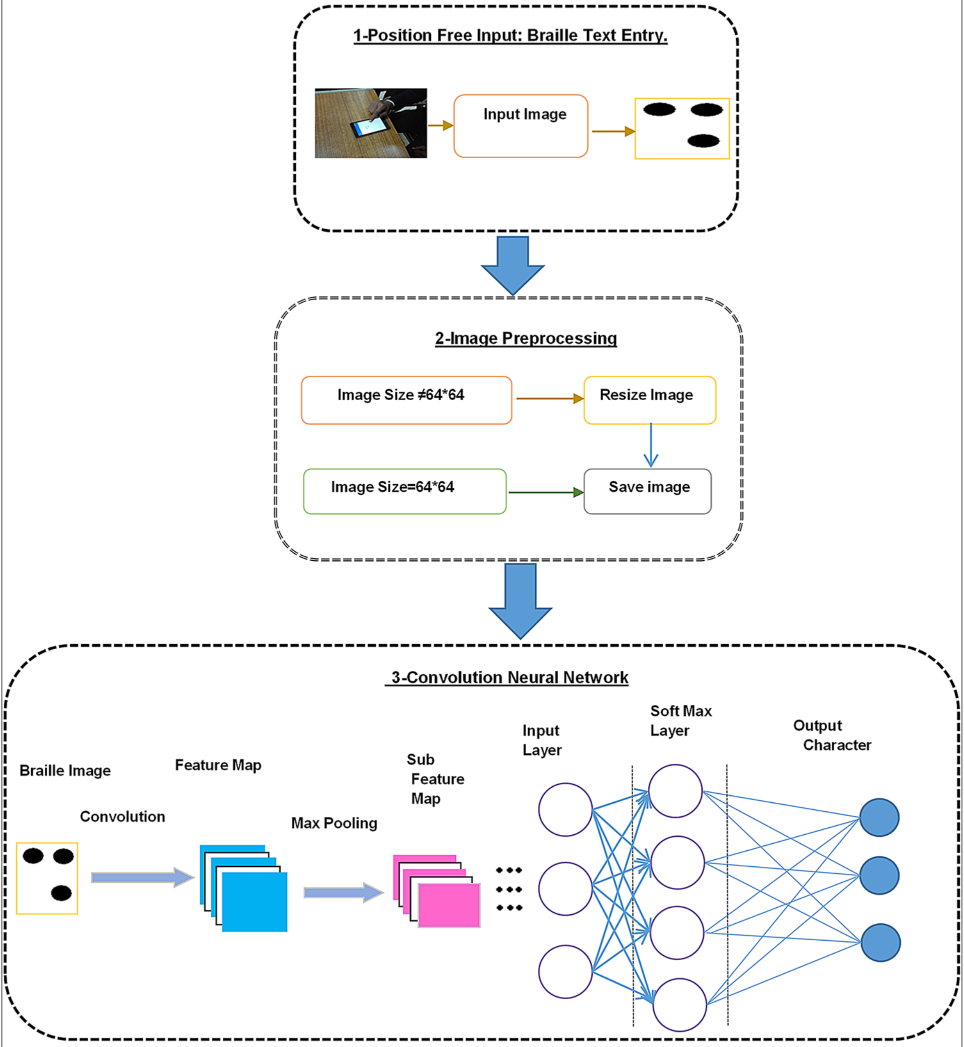


Fig. 1 Framework for text character prediction from Braille input using the DL model

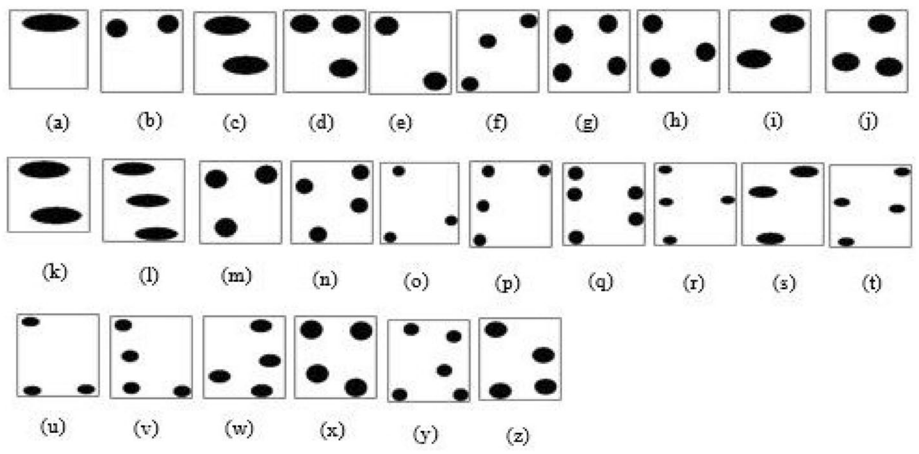


Fig. 2 Sample Braille input is taken from a participant using the proposed interface and stored in image format in the dataset

hyperparameters are not properly set, overfitting can occur in the model [58]. Therefore, we have used a batch size of 32 with 26 classes and 50 epochs. The risk of overfitting has been reduced by adding a dropout layer.

Braille text entry: interaction of visually impaired users with the proposed Braille input design

A new application is designed by keeping in mind the limitations of the earlier applications. This input method is better than previously designed techniques for the following reasons: Braille text entry places the minimum burden on visually impaired users. There are no fixed positions for entering data on the device, and the user can freely enter dots anywhere on the screen. Users only have to enter active dots, and there is no need to enter the deactivated dots. Users do not need to enter unnecessary dots, and there is no need for difficult gestures such as double taps and triple taps. Vibrio-tactile feedback is given. A touch screen interface for Braille input developed using the proposed method is depicted in Figs. 3 and 4.

Input algorithm designed for Braille image extraction

The proposed input method is illustrated in Fig. 5. In the proposed method, the user has to input the Braille dots to enter a specific character on the screen. Swipe right gestures are used to save the character and swipe left used if the user wants to clear the screen because of a mistake. At the current stage of development, the user has to enter data

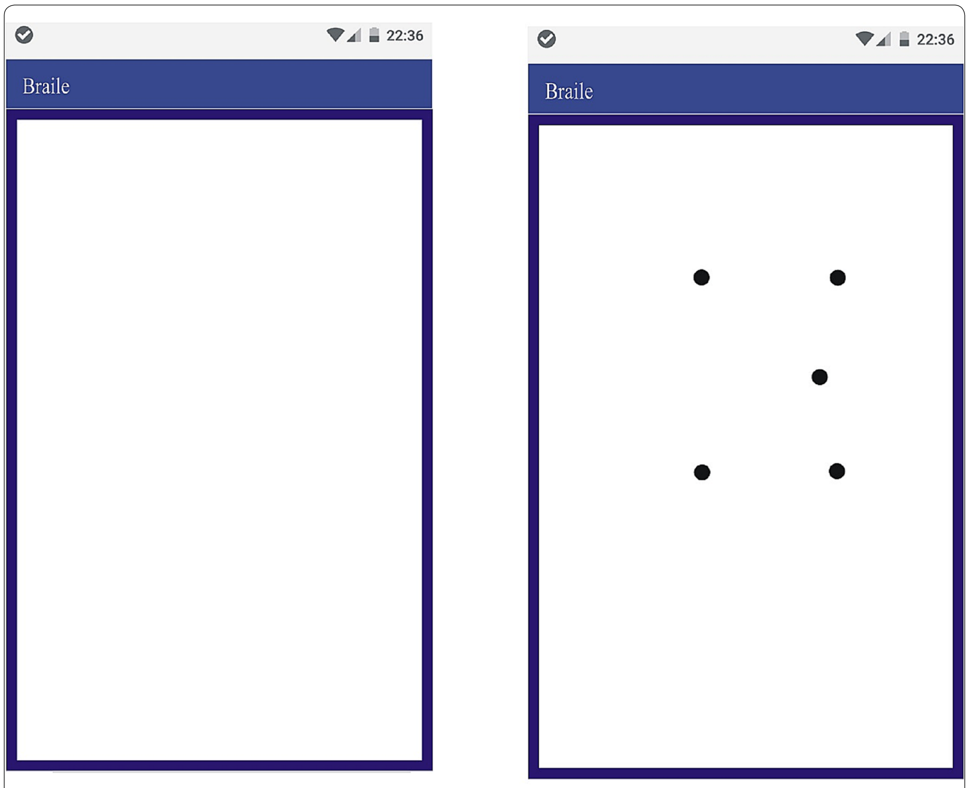


Fig. 3 Proposed Braille interface

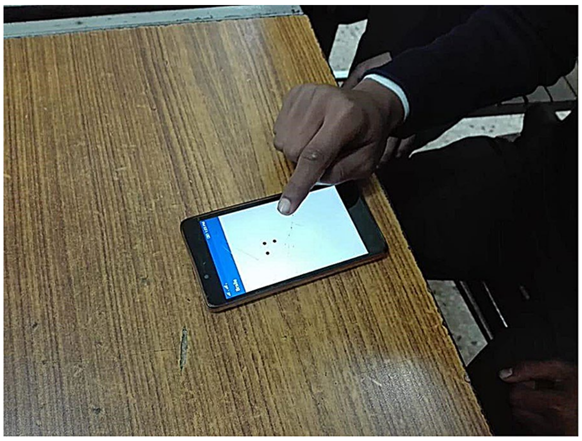


Fig. 4 A participant using the proposed Braille interfaces

character by character. This mechanism is extremely easy for visually impaired users to memorize and use. Braille character recognition from the input image is performed using a DL model that has been trained on the collected Braille dataset. Thus, there is little burden on visually impaired users when using this application.

Classification techniques

Different DL methods can be used for analyzing the Braille dataset. CNN is the most suitable method for working with image data because it consists of several hundred layers [59].

In this research, two CNN techniques, the sequential model, and transfer learning were trained using the collected dataset.

Sequential method

In this method, no pre-trained model is available, so the model needs to be trained from scratch. This method is more challenging than the others are, but it provides better accuracy. The sequential model consists of the following layers:

Convolution layer

This layer uses a linear function. The layer consists of input and kernels (filters). The dimensions of the input and kernel should be the same; the rest of the parameters can change. Kernels are a number of activation filters that are generated from the training image data. These activations are applied to the input image to generate the output. The output consists of the same number of parameters as the input. A special dimensional parameter called the stride is also used to downsample the image.

Non-linear Activation Layer

Non-linear activation functions are most commonly used among most recent neural networks. This layer allows the model to map network input and output, which is required for learning from the dataset. The most commonly used activation

ALGORITHM *Braille Input Image 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.  Get (X, Y)
    //Draw circle at X and Y with radius 15 pixels
8.      If (X>= (scrnwidth - 30))
9.  X= maxX;
10. If (Y>= (scrnheight - 30))
11. Y=maxY;
12. Else
13.     If (X<= 20)
14. X=minX;
15.     If (maxY<= 20)
16. Y= minY;
17. drawCircle (X, Y);
18. Step 3: Check swipe direction
19.      If event:swipe left to right
20. Check image size
21. If (ImageSize>64×64)
22. Resize()
23. SaveImage()
24.     Else
25. Save Image ()
26.      Else event: swipe right to left
27. Re-initialize all variables
28. X.clear();
29. Y.clear();
30. Step 4: Pass Image to Deep Learning Model
```

Fig. 5 Algorithm for Braille image extraction from the proposed input interface

functions are the sigmoid, hyperbolic tangent, Softmax, rectified linear unit (ReLU), Leaky-ReLU, and SoftPlus. This layer introduces non-linearity into the network. For applying it on an image, negative values are replaced by zero, and the highest positive values are used.

Sigmoid function: This is a non-linear activation function; it prevents jumps in output values. Output values lie in the range of (0)–(1). It can be calculated using Eq. 1.

Where "h" is the sigmoid function and "i" is input data. E is the mathematical constant, and its value is equal to 2.718 approx.

Hyperbolic tangent: This function can easily model neutral, extremely positive, or highly negative input values. Its output values lie between (−1) – (1). It measures non-linear activations with the help of the function given in Eq. 2.

Where "f" is the hyperbolic tangent function, "b" is the input data and "e" is the exponential constant.

Softmax: This function normally works at the output layer where input is converted into multiple classes. This function works for finalizing the output using a probability distribution, as shown in Eq. 3.

Where Sigma (σ) is the SoftMax function "S" is the input "j" is the number of inputs, and e is the exponential constant.

ReLU: This function is the most widely used non-linear function. It replaces all the negative values with a zero in a pixel feature map. This layer increases the non-linear properties of a model. It enables quick convergence of a network. Neural networks are then able to learn more complex functions using Eq. 4.

Where "f" is the ReLu activation function and "a" is the input value. If this value is less than 0, It will place a 0, and if this value is greater than 0, it places value that lies in a.

$$h(i) = \sigma(i) = \frac{1}{1+e^{-i}} \tag{1}$$

$$f(b) = \tanh(\mathrm{b}) = \frac{2}{1+\mathrm{e}^{-2\mathrm{b}}} - 1 \tag{2}$$

$$\sigma\left(S_j\right) = \frac{e^{xj}}{\sum_i e^{Sj}} \tag{3}$$

$$f(a) = max(0, a) \tag{4}$$

Pooling layer

This layer also uses a non-linear activation function, and it is used for downsampling. The pooling layer reduces the dimensionality of each feature map by extracting the highest values that contain the most important features. This layer reduces the computational complexity of the network.

Fully connected layer

The Fully Connected Layer is the final output layer. Every node in this layer is connected to another layer as in an ordinary neural network. After processing from this layer, a bunch of output data is generated. The top three outputs are selected using a probability distribution algorithm such as Softmax or a SVM. The fully connected layer is very load driven, and it makes a network load bound.

$$\delta(y)_j = \frac{e^{yj}}{\sum_{m=1}^{m} eym} \tag{5}$$

where delta (δ) shows the fully connected layer function. "Y" is the input data, and "j" is the number of input data, and e is the exponent function.

Transfer learning

Transfer learning is used to transfer base knowledge to any relevant target [60]. Transfer learning achieves accuracy by applying previously gained information to a new task. After the innovation of transfer learning in CNN, a considerable improvement in accuracy was shown for small samples of images [61, 62]. Transfer learning can improve learning using three different methods: (1) without applying any further learning and only transferring the already extracted knowledge, (2) by reducing the time required to train the model for a task from scratch using transferred knowledge, and (3) by improving the final performance level [63]. The GoogLeNet architecture of transfer learning is used in this study.

GoogLeNet (Inception model)

GoogLeNet is a CNN-based architecture that has won the "ImageNet Large Scale Visual Recognition Challenge" [64]. It achieved a substantial improvement over ZFNet, which was the winner in 2013 [65]. In another study, Wei et al. proposed a CNN based 3D object classification model and achieved an accuracy of up to 93.3% [66]. The network architecture of GoogLeNet differs from other models because a convolution layer of 1×1 kernels with ReLU activation is used in the middle of the model. The major advantage of using 1×1 convolutions is the reduction of computational complexity. Instead of using fully connected layers, global average pooling is used. Different sizes of convolution kernels are applied to the input and all the outputs are concatenated using an Inception module [67]. In the proposed technique, along with the position-free text entry method, we reduced the tapping time of the user. The input character is recognized by the application using DL. Thus, it reduces the burden on the user by shifting it to the application.

Naïve Bayes algorithm

NB is a popular text classification technique that is based on Bayes theorem for calculating probability [68]. This simple technique is most suitable for multi-class problems. This algorithm assumes that the presence of one feature does not affect the presence of any other feature.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{6}$$

The posterior probability is calculated using:

- Where P(c|x) is the posterior probability of the class,
- P(x|c) is the likelihood of the class,
- P(c) is the prior probability of the class,
- Moreover, P(x) is the predictor prior to probability.

The following steps are needed to perform the classification using NB: Initially, the dataset is converted into a frequency table, by finding the probabilities create a like hood table and after this NB equation is used to calculate the posterior probability of each class.

Support vector machines

SVM is a generalized classifier, and it can be used for different domains like character recognition, image recognition, etc. This technique even works better where there is a small sample of data for training [68]. Kernel trick is used to handle non-linear separable data [69]. The non-linear 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.

The decision function for an input "i" is given below:

$$D(i) = V.i - x \tag{7}$$

where "V" is the vector to the normal plane and "i" is the displacement relative to the origin.

Decision Trees

DT is also one of the famous machine learning models used. It has a tree-like structure, and it is easy to understand even for non-expert users. Data is checked, and common attributes are extracted from all the classes [68]. These attributes are further divided into branches until it meets the required criteria.

Mathematically it can be represented as follows:

$$(S, Z) = (S_1, S_2, S_3, S_4 \ldots\ldots\ldots S_n, Z) \tag{8}$$

where "S" is a vector that is composed of different features used for classification and "Z" is the target variable.

K Nearest Neighbors

KNN is a machine-learning algorithm that can be used for both classification and regression-based problems. This algorithm calculates the distance between each object, and then the nearest K training objects are delineated [70]. At last, K objects that belong to the same category are classified.

Performance measures

Prediction assessment using a confusion matrix

Confusion Matrix is a way of representing the performance of the classification algorithm with deep insight into the number of observations of each class. It shows the standard output for binary or multi-class classification problems [71]. It plots instances of output classes' verses predicted classes. Performance metrics that are used to analyze the performance of these classification techniques based on confusion matrix are True Positive Rate (TPR), True Negative Rate (TNR), Positive predictive value (PPV), Negative predictive value (NPV), False Positive Rate (FPR), False Negative Rate (FNR), False Discovery Rate (FDR) and Total Accuracy (TA) as shown in Table 1. TNR is calculated as the true negative that measures the proportion of actual negative results that are correctly identified. We have correctly classified TPR True Positive and True Negative samples. PPV shows the number of positive results that are actually true. NPV measures how likely it is that a negative result is actually predicted as negative. FPR is how often the model depicts a false result as true. Accuracy is the measure of the proportion of True Positives (TP) and True Negatives (TN). These models are also compared with NB, KNN, SVM, and DT algorithms.

$$Recall/Sensitivity(TPR) = \frac{TP}{TP + FN} \tag{9}$$

$$Specificity(TNR) = \frac{TN}{TN + FP} \tag{10}$$

$$PositivePredictiveValue(PPV) = \frac{TP}{TP + FP} \tag{11}$$

$$NegativePredictiveValue(NPV) = \frac{TN}{TN + FN} \tag{12}$$

Table 1 Comparison of DL and classical machine learning techniques

Classification techniques used	TPR	TNR	PPV	NPV	TA	FPR	FNR	FDR
Naïve Bayes	NaN	96.64%	NaN	99.38%	96.38%	3.36%	NaN	NaN
Decision Trees	NaN	98.82%	NaN	98.23%	97.20%	1.18%	NaN	NaN
KNN	NaN	98.60%	NaN	98.11%	97.04%	1.40%	NaN	NaN
SVM	68.67%	99.16%	76.75%	98.72%	83.00%	0.84%	31.33%	23.25%
Sequential Model	90.76%	99.63%	91.07%	99.2%	92.21%	0.36%	9.20%	8.90%
GoogLeNet Model	95.89%	99.83%	96.61%	99.83%	95.8%	0.16%	4.10%	3.39%

$$FalsePositiveRate(FPR) = 1 - Specificity \tag{13}$$

$$TotalAccuracy(TA) = \frac{TP + TN}{TN + FP + TP + FN} \tag{14}$$

The other well-known performance measures for multi-class classification problems are Receiver Operating Characteristics (ROC) and AUC. ROC is one of the most appropriate assessment techniques for evaluating the performance of any classification model. It measures the performance of any classification problem using different threshold settings [72]. This indicates how much a model can distinguish between different classes. ROC accuracy is better than the total accuracy as total accuracy is only measured at a specific cut point, whereas ROC accuracy is measured on all cut points.

Experimental setup

In the model proposed, the user inputs Braille dots using touch screens, as shown in Fig. 4. The input is saved as an image. This image is processed to map the Braille dots to the corresponding Braille character. The algorithm for predicting English character is shown in Fig. 6.

A sequential model, which consists of a linear stack of layers, was built, trained, and validated using the Braille dataset. Each layer is considered an object. Variables are initialized for the training and validation of the model, which feeds data to the next layer. An input image of 64 × 64 is passed to the convolution layer. This layer works as

```
ALGORITHM English Character Prediction
Input: Braille Dataset
Output: Predicted English Character
1.  Repeat Step 2 to 16 for all images
2.  For each image
3.      If size> 64×64
4.              Reduce Size = 64× 64
5.      Else
6.      For (i=1; i++; i<=10)
7.      {
8.              Ifi>6;
9.                      j=6;
10.             Else
11.                     j= i;
12.     Step 1 Conv $2^j-1×1$
13.     Step 2 Conv Activation ×ReLU $2^j-1×1$
14.     Step 3 MaxPool = $1×1$
15.     Step 4 DropOut =0.25
16.     }
```

Fig. 6 Algorithm for English character prediction

a "flashlight" layer that illuminates the major portions of the image. The "flashlight" is the filter, and the region it illuminates is the receptive field. A filter is also an array of numbers. These numbers are weights at a particular layer, which acts as a feature identifier. Because the features convolve with the input, their values are multiplied by the input image pixels. This method is called pyramid-wise multiplication. The results of multiplication from each region are then summed for all parts of the image. CNN learns the values for its filters during training. Then, the feature information is passed to the next layer, which is the activation layer. This layer has a non-linear activation function. The pooling layer reduces the dimensionality of each feature map but retains the most important features, reducing the computational complexity of this network.

These three layers are grouped together and then repeated nine times. The features of the first layer are more abstract than those of the other layers. The output feature map is input to the next layer, and the filter in each layer learns more abstract features. The dropout layer drops a random set of activations in that layer by setting them to zero as data flows through it. Minimize function is used to measure the difference between the targeted and the expected output. To minimize the loss function, the derivative of the loss with respect to the weights in each layer is calculated.

$$\sigma'_j = \frac{\partial j}{\partial s_j^i} \tag{15}$$

The derivation is performed starting from the last layer to compute the directions in which network up-gradation is required. The loss is propagated backward for each layer, and the value of each filter is updated so that they change in the direction of the gradient to minimize the loss. The learning process is intrigued by using the compile method. Adam optimization can be used to maximize accuracy in a smaller number of epochs, but it increases the training time because of the increase in the number of images [57, 73]. It individually calculates the learning rate for the first and second moments of the gradient weight of the neural network. The end fit function is used for training and validation. Input and output neurons of DNN architecture are shown in Fig. 7.

Results and discussion

The classification performance was evaluated using Deep Learning techniques like Sequential Model and GoogLeNet Inception Model. For making a comparative analysis, classical machine learning techniques like NB, DT, SVM, and KNN are also applied. Features were extracted using the traditional feature extraction method. The performance was evaluated using the following performance metrics: TPR, TNR, PPV, TA, FPR, FNR, FDR, and AUC.

In this study, we have analyzed the classification of English Braille alphabets. Figures 8 and 9 show the results in the form of the ROC curve that is plotted with the TPR on the Y-axis and the FPR on the X-axis. The AUC represents a portion of a square unit, and its value lies between 0 and 1. An AUC greater than 0.5 indicates the separation of the classes [74]. Figure 8 shows the performance of the Sequential Model for Category- A (a-n). The highest separation is achieved for all the classes from a-n. Similarly, for Category-B (n-z) we have obtained the highest separation for all the classes except for class

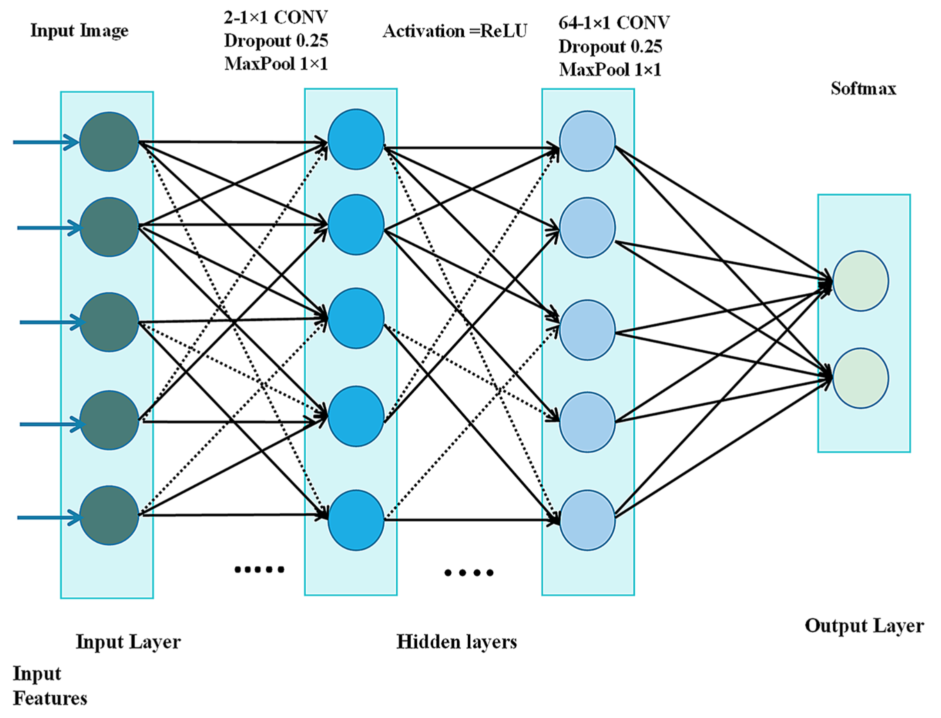


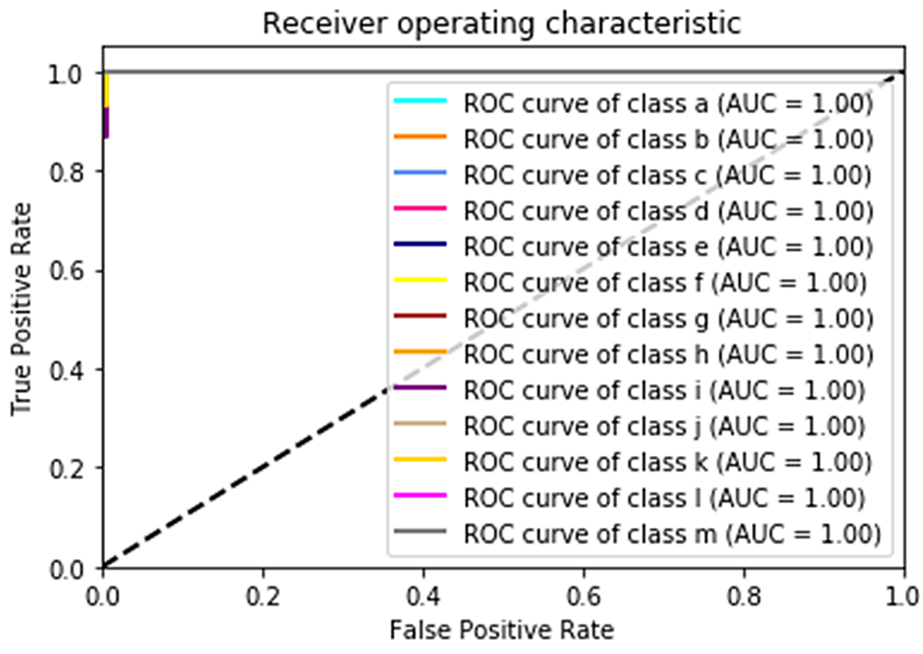
Fig. 7 Graphical representation of Input and Output neurons of DNN

"o," i.e. AUC (0.98) and "x," i.e. AUC (0.97). The total accuracy achieved for Sequential Model is 92.21%. The model has an average micro and macro ROC separation of 100%, as shown in Fig. 10a.

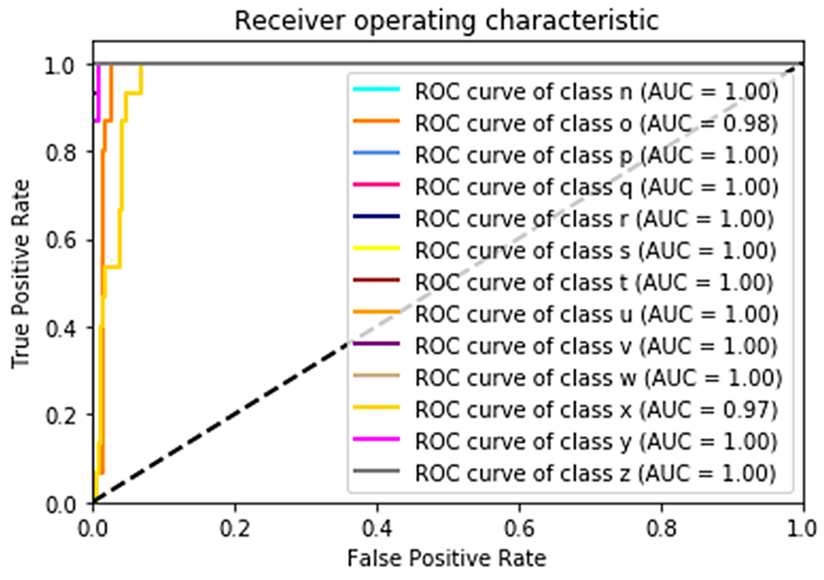
Figure 9 shows the performance of the GoogLeNet Inspection Model. The results show that the highest separation is achieved for all the cases of Category-A (a–m). For Category-B (n–z) highest separation is achieved except for class o i.e. AUC (0.98) and x i.e. AUC (0.099). The total accuracy achieved for the GoogLeNet inspection model is 95.8%. It has also achieved an average separation of 100% in terms of micro and macro ROC, as shown in Fig. 10b.

Figure 11 demonstrates a comparison of various machine-learning techniques. The results obtained from the Naïve Bayes classifier show that the highest accuracy was achieved for the character "a" with TA = 100%, TPR = 100%, and TNR = 97%. Similarly, the Decision Tree classifier has shown that the highest accuracy is achieved for "v" with TA = 100%, TPR and TNR = 100% followed by "w" that achieved TA = 92% with TPR and TNR = 100%. KNN showed TA = 100% with TPR and TNR = 100% for "a"," b"," f"," q" and "w". Followed by "a", "c" and "d" with TA = 99% and TPR, and TNR = 100%. Moreover, SVM also showed TA = 100%, TPR and TNR = 100% for "a", "e", "i" and "r". Followed by "c" and "l" that achieved TA = 99% and TPR = 100%.

We have also analyzed our results using the confusion matrix for the GoogLeNet Inception model. In Confusion Matrix, Truly predicted classes are shown on diagonal entries of an n × n matrix. Figure 12 shows a confusion matrix of a 26 class problem for Braille to English alphabet conversion using GoogLeNet architecture. There is a



(a) ROC Sequential Model for Category A(a-m)

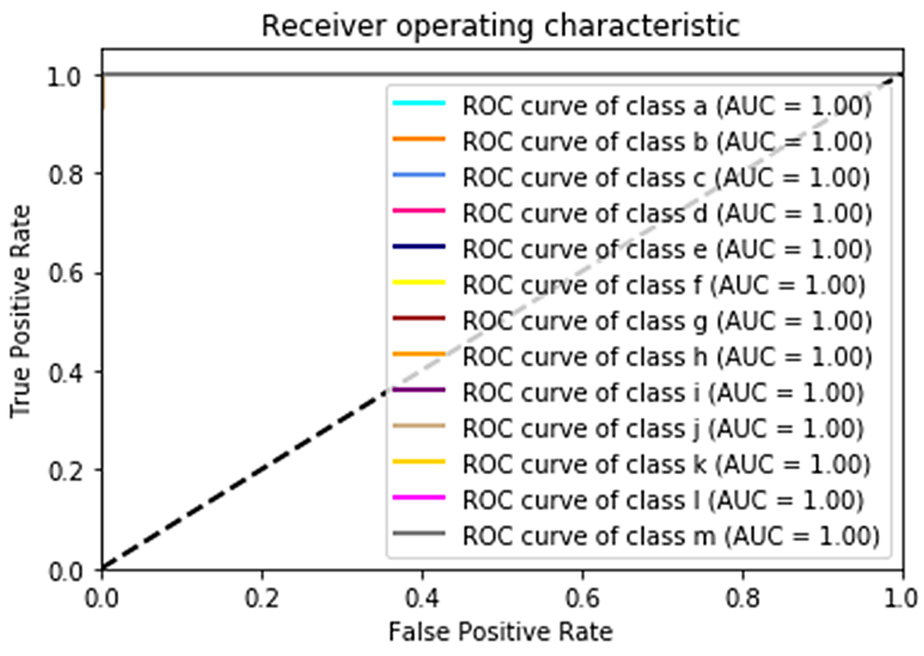


(b) ROC Sequential Model for Category-B (n-z)

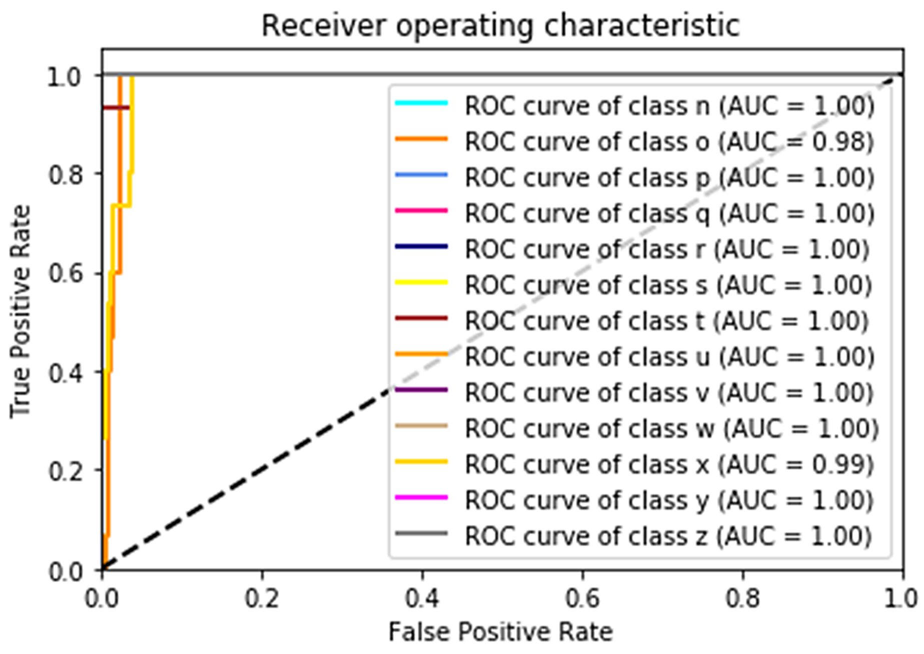
Fig. 8 ROC for the sequential model for **a** classes a–m and **b** classes n–z

possibility of confusion among several classes. As in this case, class l is misclassified as class b, class w is predicted as class d, j is predicted as t, and o is predicted as x.

We have also compared the results acquired from Deep learning techniques with Classical machine learning techniques. Table 1 shows a comparison of DL with classical machine learning techniques.



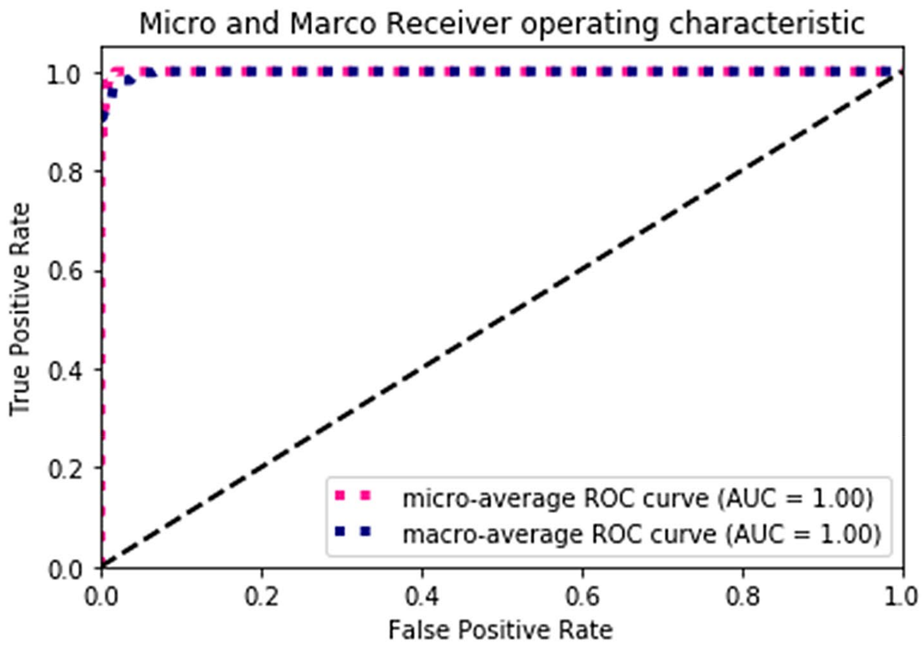
(a) ROC GoogLeNet for Category-A(a-m)



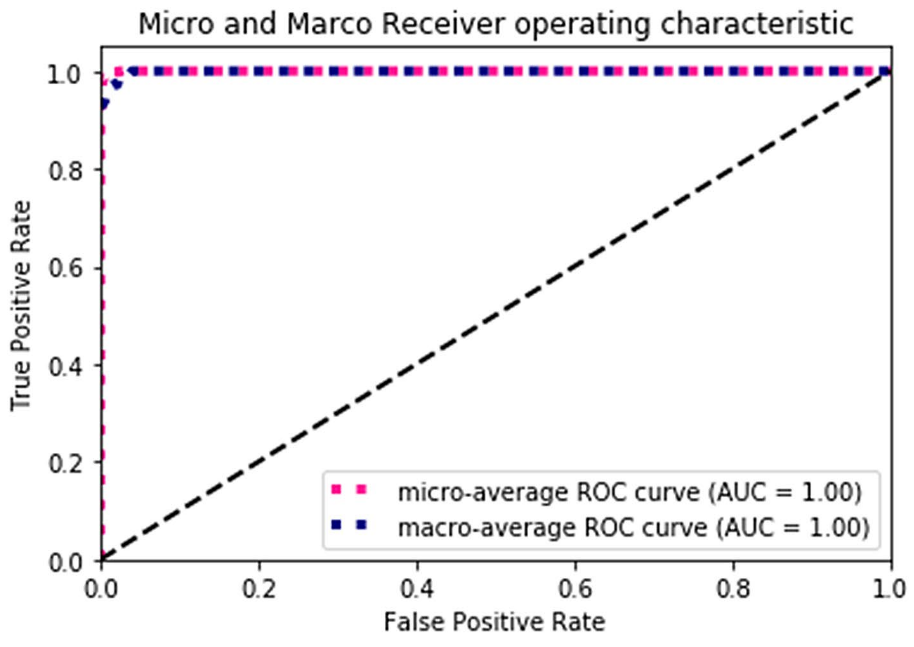
(b) ROC GoogLeNet for Category-B (n-z)

Fig. 9 ROC for GoogLeNet for **a** classes a–m and **b** classes n–z

All the comparisons are made using similar performance measures such as TPR, TNR, PPV, NPV, TA, FPR, FNR, and FDR. Among all these classifiers, the performance of the GoogLeNet Inspection Model was the best. It has achieved TA = 95.8% along with TPR = 95.89%, TNR = 99.83% and FDR = 3.39% only. The second-highest performance was achieved by Sequential Model with TA = 92.21%, TPR = 90.76%, TNR = 99.63%,



(a) Macro and Micro ROC Sequential Model



(b) Macro and Micro ROC GoogLeNet

Fig. 10 Macro and micro ROCs **a** for the sequential model and **b** GoogLeNet

and FDR = 8.90%. The third highest performance was achieved by SVM with TA = 83%, TPR = 68.67%, and TNR = 99.16% and FDR = 23.25%. Rest of the three classifiers NB, DT, and KNN obtained better TA = 96.38%, 97.20%, and 97.04% respectively than others

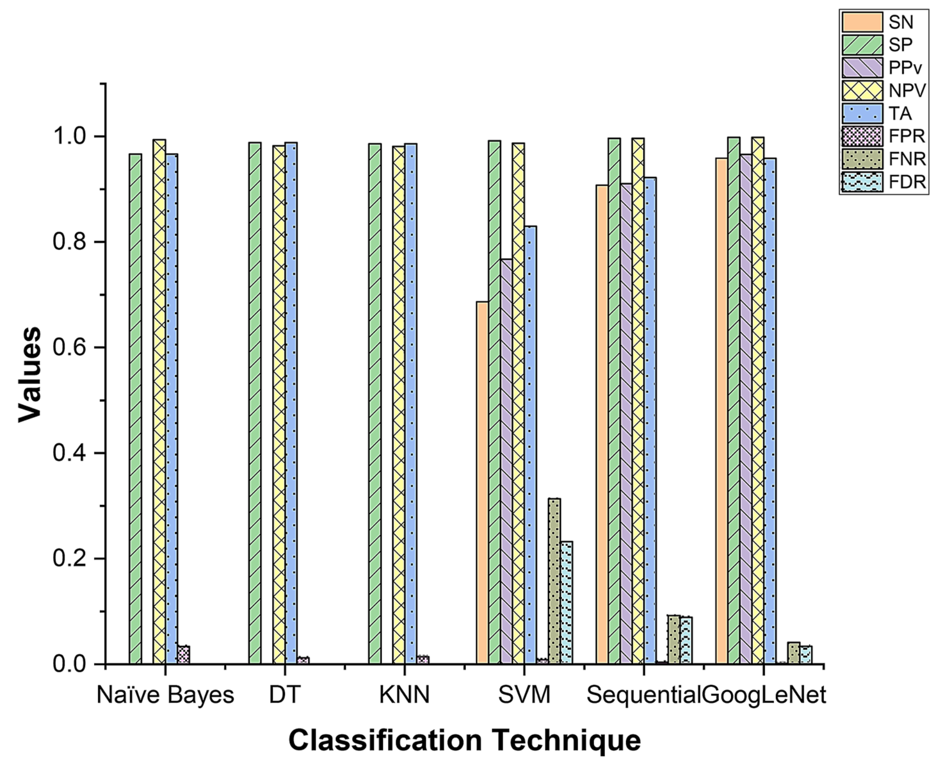
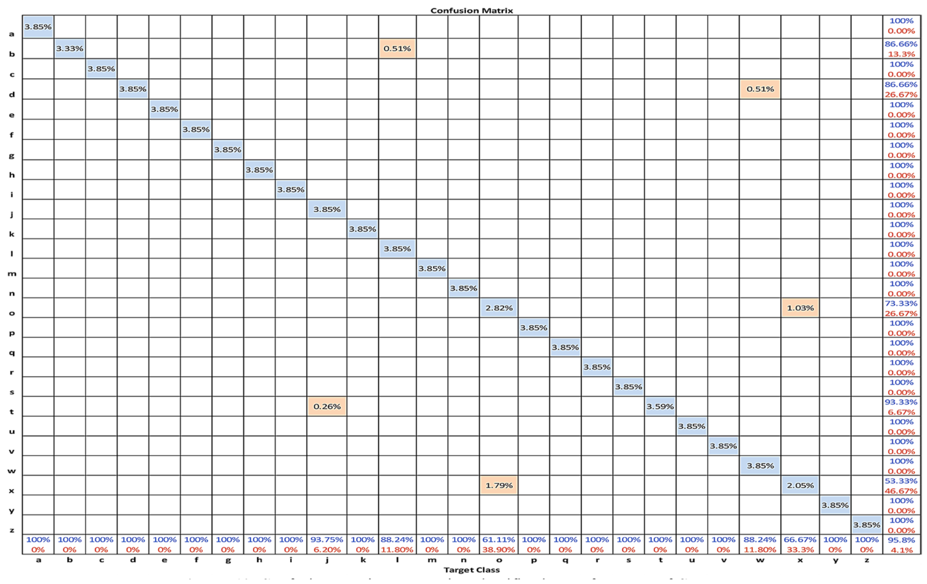


Fig. 11 Comparison with classical machine learning techniques



Confusion Matrix

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	
a	3.85%																										100% 0.00%
b		3.33%										0.51%															86.66% 13.3%
c			3.85%																								100% 0.00%
d				3.85%																			0.51%				86.66% 26.67%
e					3.85%																						100% 0.00%
f						3.85%																					100% 0.00%
g							3.85%																				100% 0.00%
h								3.85%																			100% 0.00%
i									3.85%																		100% 0.00%
j										3.85%																	100% 0.00%
k											3.85%																100% 0.00%
l												3.85%															100% 0.00%
m													3.85%														100% 0.00%
n														3.85%													100% 0.00%
o															2.82%									1.03%			73.33% 26.67%
p																3.85%											100% 0.00%
q																	3.85%										100% 0.00%
r																		3.85%									100% 0.00%
s																			3.85%								100% 0.00%
t										0.26%										3.59%							93.33% 6.67%
u																					3.85%						100% 0.00%
v																						3.85%					100% 0.00%
w																							3.85%				100% 0.00%
x															1.79%									2.05%			53.33% 46.67%
y																									3.85%		100% 0.00%
z																										3.85%	100% 0.00%
	100% 0%	100% 0%	100% 0%	100% 0%	100% 0%	100% 0%	100% 0%	100% 0%	100% 0%	93.75% 6.20%	100% 0%	88.24% 11.80%	100% 0%	100% 0%	61.11% 38.90%	100% 0%	100% 0%	100% 0%	100% 0%	100% 0%	100% 0%	100% 0%	88.24% 11.80%	66.67% 33.3%	100% 0%	100% 0%	95.8% 4.1%

Target Class

Fig. 12 Confusion Matrix representing the classification performance of GoogLeNet

but with no TPR. Therefore, we cannot consider these performances for better classification. Thus, TA alone cannot indicate the efficiency of a classifier. In addition to a high TA, an efficient classifier must have higher values of TPR, TNR, PPV, NPV, and smaller values of FPR, FNR, and FDR. Therefore, GoogLeNet shows better performance than the rest of the techniques, all in terms of TPR, TNR, PPV, NPV, TA, FPR, FNR, and with reduced FDR 3.39%.

Conclusions and future work

Deep learning (DL) has been widely used for pattern recognition. To our knowledge, this is the first study to use this technique for Braille-to-natural language conversion. In this research, we have designed an innovative touchscreen-based Braille input method that provides maximum ease-of-use for visually impaired people. A Braille dataset was then collected using a purpose-built Android-based smartphone application. In this study, only level 1 Braille data for English alphabets were used, and English characters corresponding to Braille input were recognized using DL techniques. Performance evaluation indicates that GoogLeNet Inception Model has the highest accuracy of 95.8% with 95.89% specificity (TNR) and 99.83% sensitivity (TPR). It also has a reduced error rate of 3.39%.

In the future, we plan to include an error correction feature so that it can be more beneficial to the visually impaired community. DL techniques will also be used for accurate classification of level 2 Braille datasets for English word recognition. Currently, this application only supports English alphabet characters but could be extended to support other regional languages such as Urdu and Pothohari. These techniques will also be applied for Braille-to-Urdu conversion and numerical characters for level 1 and level 2 dataset. Other DL techniques can be applied to improve the accuracy and efficiency of the designed interface. Several other related educational games could also be developed using this position-free input interface.

Abbreviations

AUC: Area under the Curve; CNN: Convolutional Neural Network; DL: Deep Learning; DT: Decision Tree; FDR: False Discovery Rate; FN: False Negative; FNR: False Negative Rate; FP: False Positive; FPR: False Positive Rate; GPU's: Graphics Processing Unit; KNN: K-Nearest Neighbor; NB: Naïve Bayes; NPV: Negative Predicted Value; PPV: Positive Predicted Value; ReLU: Rectified Linear Unit; RNN: Recurrent Neural Network; ROC: Receiver Operating Characteristics; SDAE: Stacked Denoising Auto Encoder; SVM: Support Vector Machine; Sn: Sensitivity; Sp: Specificity; TNR: True Negative Rate; TPR: True Positive Rate; ZFNet: Zeiler Furgus Net.

Acknowledgements

Not applicable.

Authors' contributions

SS and RR design the main algorithms. Meanwhile, they completed the revisions. SSR and AMA designed the experimental environment. AAA analyzes previous related works. SJK analyzed the experimental results. All authors read and approved the final manuscript.

Funding

This work was supported by Basic Science Research through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A3B04031440).

Availability of data and materials

The authors designed and implemented a new interface for entering Braille characters. There were no previously existing data of this type. All data used in this study were collected from the Special Education School, Mannak Payyan, Pakistan. This data is not publicly available. The corresponding author can provide this data on request.

Competing interests

The authors declare that there are no competing interests regarding the publication of this paper.

Author details

[1] Department of Computer Science and Information Technology, University of Azad Jammu and Kashmir, Muzaffarabad 13100, Pakistan. [2] Reptor Interactive (Pty) Ltd, Eco Boulvard. Witch Hazel Ave, Centurion 0157, South Africa. [3] Dept. of Computer Engineering, Kangwon National University, 346 Joongang-ro, 25913 Samcheok, Gangwon-do, South Korea.

Received: 15 November 2019 Accepted: 4 September 2020
Published online: 19 September 2020

References

1. Latest Global Blindness and VI prevalence figures published in Lancet Vision Atlas. https://atlas.iapb.org/news/latest-global-blindness-vi-prevalence-figures-published-lancet//. Accessed 8 Feb 2020
2. Rantala J, Raisamo R, Lylykangas J, Surakka V, Raisamo J, Salminen K, Pakkanen T, Hippula A (2009) Methods for presenting Braille characters on a mobile device with a touch-screen and tactile feedback. IEEE Trans Haptics 2(1):28–39
3. Grussenmeyer W, Folmer E (2017) Accessible touchscreen technology for people with visual impairments : a survey. ACM Trans Access Comput 9(2):1–31
4. Rodrigues A, Santos A, Montague K, and Guerreiro T (2017) Improving Smartphone Accessibility with Personalizable Static Overlays. In: Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility- ASSETS. Baltimore, MD, USA, ACM, p. 37–41, October 2017.
5. Jafri R, Khan MM (2018) User-centered design of a depth data based obstacle detection and avoidance system for the visually impaired. Human-centric Comput Inform Sci 8(1):1–14
6. Cao D, Chen Z, Gao L (2020) An improved object detection algorithm based on multi-scaled and deformable convolutional neural networks. Human-centric Computing and Information Sciences, Springer Open 10(14):1–22
7. Alqarni MA, Chauhdary SH, Malik MN, Ehatisham-ul-Haq M, Azam MA (2020) Identifying smartphone users based on how they interact with their phones. Human-centric Comput Inform Sci 10(1):1–14
8. Catania F (2020) Conversational Technology and Natural Language Visualization for Children's Learning. In Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, Honolulu, USA, ACM, p. 1–7. April 2020
9. Šepić B, Ghanem A, Vogel S (2015) Braille easy: one-handed Braille keyboard for smartphones. Stud Health Technol Inform 217:1030–1035
10. Be My Eyes-See the world together. https://www.bemyeyes.com/. Accessed 20 Feb 2020
11. BeSpecular . Available at: https://www.bespecular.com/. Accessed 12 Mar 2020
12. World D, TapTapSee Camera App for Visually Impaired _ Disabled World. https://www.disabled-world.com/assistivedevices/apps/taptapsee.php. Accessed 15 Feb 2020
13. Google, KNFB Reader App features the best OCR. https://www.knfbreader.com/. Accessed 10 Feb 2020
14. Kacorri H, Kitani KM, Bigham JP, and Asakawa C (2017) People with Visual Impairment Training Personal Object Recognizers : Feasibility and Challenges. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver Colorado, USA, ACM, p. 5839–5849, May 2017
15. Panda SP, Nayak NK, Rai CS (2020) A survey on speech synthesis techniques in Indian languages. Multim Syst 26:453–478
16. Matoušek J, Krňoul Z, Campr M, Zajíc Z, Hanzlíček Z, Grůber M, Kocurová M (2020) Speech and web-based technology to enhance education for pupils with visual impairment. J Multimodal User Interf 14:219–230
17. Verma P, Singh R, Singh AK (2013) A framework to integrate speech based interface for blind web users on the websites of public interest. Human-Centric Comput Inform Sci 3(1):1–21
18. Stella J, Valsan KS (2018) Text to Braille conversion: a survey. Int J Manag Appl Sci 4(1):15–18
19. Frey B, Southern C, and Romero M (2011) Braille Touch : Mobile Texting for the Visually Impaired. In: International Conference on Universal Access in Human-Computer Interaction, Springer, Berlin, Heidelberg, p. 19–25, July 2011
20. Mascetti S, Bernareggi C, and Belotti M (2011) TypeInBraille : a Braille-based typing application for touch-screen devices. In: The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility, Dundee Scotland, UK, p. 295–296, October 2011
21. Mattheiss E, Regal G, Schrammel J, Garschall M, Tscheligi M (2015) EdgeBraille: Braille-based text input for touch devices. Journal of Assistive Technologies 9(3):147–158
22. Jayant C, Acuario C, Johnson W, Hollier J, and Ladner R (2010) VBraille : Haptic Braille Perception using a Touch-screen and Vibration on Mobile Phones. In: Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS. Orlando, Florida, USA: ACM, p. 295–296, October 2010
23. Azenkot S (2014) Eyes-Free Input on Mobile Devices. Dissertation, University of Washington
24. McNaughton J, Crick T, Hatch A (2017) Determining device position through minimal user input. Human-centric Comput Inform Sci 7(1):1–37
25. Gidh VY, Latey SM, Roy A, Shah K, Ingle S (2013) Braille Calculator. Int J Eng Comput Sci 2(2):1–3
26. Siqueira J, De-Melo-Nunes FAA, Silva CRG, De-Oliveira-Berretta L, Ferreira CBR, Félix IM, and Luna MM (2016) Braille Écran: A Braille Approach to Text Entry on Smartphones. In: IEEE 40th Annual Computer Software and Applications Conference, IEEE, p. 608–609, June 2016.
27. Alnfiai M, Sampalli S (2017) BrailleEnter: A Touch Screen Braille Text Entry Method for the Blind. Procedia Comput Sci 109:257–264
28. Subash NS, Nambiar S, and Kumar V (2012) Braille Key: An alternative Braille text input system: Comparative study of an innovative simplified text input system for the visually impaired. In: 4th International Conference on Intelligent Human Computer Interaction: Advancing Technology for Humanity, (IHCI), p. 4–7, December 2012.
29. Shabnam M, Govindarajan S (2016) Gesture recognition algorithm: Braille-coded gesture patterns for touch screens: eyedroid. Indian J Sci Technol 9(33):1–9
30. Alnfiai M, Sampalli S (2016) SingleTap Braille: developing a text entry method based on Braille patterns using a single tap. Procedia Comput Sci 94:248–255

31. Alnfiai M, Sampalli S (2017) Improved SingleTap Braille : Developing a single tap text entry method based on Grade 1 and 2 Braille encoding. J Ubiquit Syst Perv Netw 9(1):23–31
32. Alnfiai M, Sampalli S (2019) Braille Tap : Developing a Calculator Based on Braille Using Tap Gestures. Universal Access in Human-Computer Interaction. Springer, Designing Novel Interactions, Vancouver, Canada, pp 213–223
33. Leporini B, Buzzi MC, and Buzzi M (2012) Interacting with mobile devices via VoiceOver: usability and accessibility issues. In Proceedings of the 24th Australian Computer-Human Interaction Conference, Melbourne, Australia, ACM, pp. 339–348, 2012
34. Karmel A, Sharma A, Garg D (2019) IoT based assistive device for deaf, dumb and blind people. Procedia Comput Sci 165:259–269
35. Boruah A, Kakoty NM, Ali T (2018) Object recognition based on surface detection-a review. Procedia Comput Sci 133:63–74
36. Guerreiro T, Lagoá P, Santana P, Gonçalves D, and Jorge J (2008) NavTap and BrailleTap: Non-Visual Texting Interfaces. In: Rehabilitation Engineering and Assistive Technology Society of North America Conference (Resna), USA, p. 1–10
37. Bier A, Sroczyński Z (2019) Rule based intelligent system verbalizing mathematical notation. Multimedia Tools and Applications, Springer 78(19):28089–28110
38. Nahar L, Jaafar A, Ahamed E, Kaish ABMA (2015) Design of a Braille learning application for visually impaired students in Bangladesh. Assis Technol 27(3):172–182
39. Iqbal MZ, Shahid S, and Naseem M (2017) Interactive Urdu Braille Learning System for Parents of Visually Impaired Students. In: Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility, Baltimore: ACM, p. 327–328, October 2017
40. Parvathi K, Samal BM, and Das JK (2015) Odia Braille : Text Transcription via Image Processing. In: International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), USA: IEEE, p. 138–143, Feb 2015
41. Al-Shamma SD, and Fathi S (2010) Arabic Braille recognition and transcription into text and voice. In: 5th Cairo International Biomedical Engineering Conference, (CIBEC). Cairo, Egypt: IEEE, p. 227–231, Dec 2010
42. Devi GG, and Sathyanarayanan G (2018) Braille Document Recognition in Southern Indian Languages–A Review. In 2018 Fourth International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), IEEE, pp. 1–4. Feb 2018
43. Nasib AU, Kabir H, Ahmed R, and Uddin J (2018) A real time speech to text conversion technique for bengali language. In: International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2). Rajshahi, Bangladesh, IEEE, pp. 1–4. Sept 2018
44. Rasheed I, Gupta V, Banka H, and Kumar C (2018) Urdu Text Classification: A comparative study using machine learning techniques. In Thirteenth International Conference on Digital Information Management (ICDIM). Berlin, Germany, IEEE, pp. 274–278. Sept 2018
45. Wang X, Zhong J, Cai J, Liu H. and Qian Y (2019) CBConv: Service for Automatic Conversion of Chinese Characters into Braille with High Accuracy. In: The 21st International ACM SIGACCESS Conference on Computers and Accessibility. Pittsburgh, USA, ACM, pp. 566–568, Oct 2019
46. Bengio Y, Lamblin P, Popovici D, and Larochelle H (2006) Greedy Layer-Wise Training of Deep Networks. In: NIPS'06: In: Proceedings of the 19th International Conference on Neural Information Processing Systems, British Columbia, Canada, ACM, p. 153–160, 2006
47. Jaswal D, Sowmya V, Soman KP (2014) Image Classification Using Convolutional Neural Networks. International Journal of Scientific and Engineering Research 5(6):1661–1668
48. Gao X, Zhang J, and Wei Z (2018) Deep Learning for Sequence Pattern Recognition. In: 15th IEEE International Conference on Networking, Sensing and Control (ICNSC), Zhuhai, China: IEEE, p. 1–16, Mar 2018
49. Li T, Zeng X, and Xu S (2014) A deep learning method for Braille recognition. 6th International Conference on Computational Intelligence and Communication Networks, (CICN) 2014, p. 1092–1095, Nov 2014
50. Murthy VV, and Hanumanthappa M (2018) Improving Optical Braille Recognition in Pre-processing Stage. In: International Conference on Soft-computing and Network Security (ICSNS), Coimbatore, India, IEEE, pp. 1–3, Feb 2018
51. Jha V, Parvathi K (2019) Braille Transliteration of hindi handwritten texts using machine learning for character recognition. Int J Sci Technol Res 8(10):1188–1193
52. Jha V, Parvathi K (2020) Machine learning based Braille transliteration of odia language. Int J Innov Technol Explor Eng 5:1866–1871
53. Perera TDSH, Wanniarachchi WKILI (2018) Optical Braille recognition based on histogram of oriented gradient features and support-vector machine. Int J Eng Sci 8(10):19192–19195
54. Li J, Yan X, and Zhang D (2010) Optical Braille Recognition with Haar Wavelet Features and Support-Vector Machine. In: International Conference on Computer, Mechatronics, Control and Electronic Engineering. Changchun, China: IEEE, p. 64–67, Aug 2010
55. Udapola UBHS, and Liyanage SR (2017) Braille Messenger : Adaptive Learning Based Non- Visual Touch Screen Input for the Blind Community Using Braille. In: International Conference on Innovations in Info-business and Technology, Ozo, Colombo, Sri Lanka, p. 1–11, Nov 2017
56. Choudhury AA, Saha R, Shoumo SZH, Tulon SR, Uddin J, and Rahman MK (2018) An Efficient Way to Represent Braille using YOLO Algorithm. In: Joint 7th International Conference on Informatics, Electronics and Vision (ICIEV) and 2nd International Conference on Imaging, Vision and Pattern Recognition (icIVPR), IEEE, pp. 379–383, 2018
57. Balasuriya BK, Lokuhettiarachchi NP, Ranasinghe ARMDN, Shiwantha KDC, and Jayawardena C (2017) Learning Platform for Visually Impaired Children through Artificial Intelligence and Computer Vision. In: 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA) Learning, Colombo, Sri Lanka: IEEE, p. 1–7, Dec 2017
58. Zhang J, Wei Z, Chen J (2018) Subject Section A distance-based approach for testing the mediation effect of the human microbiome. Bioinformatics 34(11):1875–1883
59. Pan SJ (2010) Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359

60. Simonyan K, and Zisserman A (2015) Very Deep Convolutional Networks For Lrage-Scale Image Recognition. In: Conf. on Learning Representations (ICLR), San Diego, CA, USA, p. 1–14, 2015
61. Torrey L, Shavlik J (2010) Transfer Learning. Handbook of research on machine learning applications and trends: algorithms, methods, and techniques, vol 2. IGI Global, Hershey, USA, pp 242–264
62. Bengio Y, and Haffner P (1998) Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, New York City, USA, p. 2278–2324, June 1998
63. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, and Rabinovich A (2015) Going Deeper with Convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Boston, Massachusetts, USA, IEEE, p. 1–9, June 2015.
64. Zeiler MD, and Fergus R (2014) Visualizing and Understanding Convolutional Networks. In: 3th European conference on computer vision, Zurich, Switzerland, Springer, p. 818–833, September 2014.
65. Network of Networks - Encyclopedia. https://www.encyclopedia.com/computing/news-wires-white-papers-and-books/network-networks/. Accessed 5 Feb 2020.
66. Song W, Zhang L, Tian Y, Fong S, Liu J, Gozho A (2020) CNN-based 3D object classification using Hough space of LiDAR point clouds. Human-centric Comput Inform Sci 10(1):1–14
67. Kingma DP, and Ba J (2015) ADAM: A Method For Stochastic Optimization.In: Conf. on Learning Representations (ICLR). San Diego, CA, USA, p. 1–15, May 2015
68. Pranckevičius T, Marcinkevičius V (2017) Comparison of Naïve Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. Baltic J Modern Computing 5(2):221–232
69. Tharwat A (2019) Parameter investigation of support vector machine classifier with kernel functions. Knowl Inform Syst 61(3):1269–1302
70. Song W, Zhang L, Tian Y, Fong S, Liu J, and Gozho A (2014) KNN Algorithm with Data-Driven k Value KNN Algorithm with Data-Driven k Value. In: International Conference on Advanced Data Mining and Applications, Guilin, China, Springer, p. 499–512, December 2014
71. Choi JY, Yoo TK, Seo JG, Kwak J, Um TT, Rim TH (2017) Multi-categorical deep learning neural network to classify retinal images: a pilot study employing small database. PLoS ONE 12(11):1–16
72. Salim M (2018) Deep Neural Network Models for Image Classification and Regression. Dissertation, University of Trento
73. Zhang S, Bao Y, Zhou P, Jiang H, and Dai L (2014) Improving Deep Neural Networks For LVCSR Using Dropout And Shrinking Structure. In: IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP), IEEE, p. 6849–6853, May 2014
74. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958

Publisher's Note