International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(5s) 2026 pp. 623-635
https://doi.org/10.70917/ijcisim-2026-2758
Article

# An Intelligent Technique for the Classification of Consonants and Vowels in Devnagari Script

**Shilpa Tyagi[1]* , Chiranjit Dutta[2], Manu Singh[3]**

[1]Department of Computer Applications, SRM Institute of Science and Technology,Delhi-NCR Campus, Ghaziabad, Uttar Pradesh, India
*shilpatyagi897@gmail.com
[2]Department of Computer Science & Engineering, SRM Institute of Science and Technology,Delhi-NCR Campus, Ghaziabad, Uttar Pradesh, India
[3]Department of Computer Science & Engineering, ABES Engineering College,Ghaziabad, Uttar Pradesh, India

**Abstract:** Convolution neural networks (CNN) and probabilistic neural networks (PNN) have taken considerable steps to recognize handwritten characters of consonants and vowels in the Devanagari script. The main objective of this work is to use the benefits of CNNS and PNNS to solve difficulties caused by the complex structure of the figures of Devanagari. CNN, known for its ability to extract spatial and hierarchical information, is particularly good for observing complex patterns and small variations between characters, which is ideal for this task. On the contrary, PNN uses probabilistic methodology depending on statistical metrics to effectively categorize characters, guaranteeing rapid convergence and high accuracy. The study examines the relative efficiency and usability of these approaches to recognize Devanagari scripts by merging them. A large collection of handwritten figures of Devanagari, including vowels and consonants, is used for the experiment. To improve the quality and variability of data, the data set has undergone extensive pre-process, which included enlargement and normalization. With a small medium square error (MSE) and good correlation coefficients, the CNN model showed exceptional capabilities of elements extraction, reached more than 98% of training accuracy and over 96% accuracy on validation and test data sets. Similarly, because the PNN model is non -neural, it did exceptionally well and gained the same accuracy with less training time. These findings show how well the two models of the complexity of the Devanagari script are suitable.
**Keyword:** CNN, Devanagari Script, Optical Character Recognition, PNN, Deep Learning.

## 1. INTRODUCTION

State governments in India use regional languages or scripts to document decisions, fulfill the minutes, notes and recommendations by higher expiry of their officials at lower levels and other information. The disputed documents must recognize the authorship of written decisions, notes, registration of meetings and recommendations. It is difficult for a human being to read and translate all these documents in English manually. Devanagari, a script used for languages such as Hindi, Marathi and Sanskrit, represents a unique challenge for processing text thanks to its complex structure, which includes vowels, consonants, mantras (diacritic marks) and ligatures [1]. Devanagari hand -written script recognition is a sophisticated computer program that reads and decodes a readable manually written input before doing it by a number of automated procedures for identifying and creating a script that is comparable. The process of script recognition consists of three steps: preprocessing, feature extraction, and classification (recognition). To enable the efficient extraction of distinctive features from source photos, preprocessing involves eliminating noise from the images. In the feature extraction step, different features are extracted. Recognition simply involves classifying the features that have been extracted using the features that are stored or available for those sets of characters or words.

Handwritten character identification is a demanding company due to natural differences in writing styles among the authors. The challenge of the character recognition will be extremely demanding due to variations of written styles, patterns, and sizes. It creates one of the most important aspects of image processing and images recognition is the extraction of the elements. Character characters are components that can be uniquely identified [2]. Finding a collection of the most striking and useful features for the classification process is the basic goal of extraction and choosing functions. Maintaining high accuracy of classification can also compress high -dimensional space of functions to low -dimensional functions.

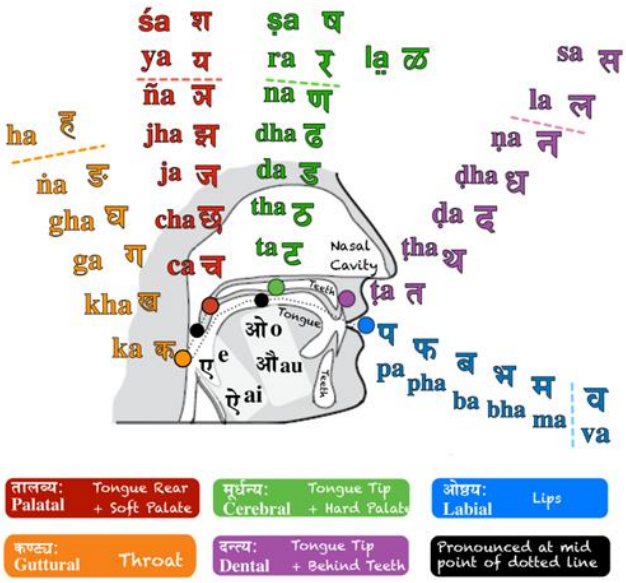



Fig. 1. Devnagri Alphabets[1]

| | |
|---|---|
| Vowels | अ आ इ ई उ ऊ ए ऐ ओ औ अं अः |
| Consonants | क ख ग घ ङ च छ ज झ ञ ट ठ ड ढ ण<br>त थ द ध न प फ ब भ म य र ल व ह श<br>स ष क्ष ज्ञ |

Fig. 2. Devnagari Script

Figure 1 illustrates the classification and articulation points of Devanagari consonants and vowels based on phonetic principles. It maps the sounds of the Devanagari script to specific regions of the vocal apparatus, such as the tongue, palate, lips, and throat. The chart categorizes sounds into different groups based on their place and manner of articulation.

- Velar (Guttural): Sounds like "ka", "kha", "ga", and "gha" are produced by the rear part of the tongue contacting the soft palate.
- Palatal: Consonants such as "cha", "chha", "ja", and "jha" involve the tongue pressing against the hard palate.
- Retroflex (Cerebral): Sounds like "ṭa", "ṭha", "ḍa", and "ḍha" are articulated by curling the tongue tip back to touch the roof of the mouth.
- Dental: Consonants such as "ta", "tha", "da", and "dha" are formed by the tongue touching the back of the upper front teeth.
- Labial: Sounds like "pa", "pha", "ba", "bha", and "ma" are produced using the lips.

The figure also highlights nasal consonants like "ṅa", "ña", and "na", which involve airflow through the nasal cavity. Additionally, vowels are positioned based on the tongue's movement and resonance within the oral and nasal cavities, such as "a", "ā", "i", "ī", "u", "ū", "e", and "ai". The dotted lines represent unique articulation points for semi-vowels ("ya", "ra", "la", "va") and aspirated sounds, which are pronounced with a burst of air [3]. The color-coded regions correspond to specific articulatory zones: the throat (guttural), palate (palatal), dental ridge (dental), tongue tip (cerebral), and lips (labial). Each group of sounds is systematically organized and represents a scientific phonetic basis of Devanagari, which is in line with classical linguistic theory. This illustration emphasizes the organized and logical structure of Devanagari phonology by offering an anatomical and phonetic basis for

[1] https://www.sanskritacademy.org/consonants/

understanding. The vowels and consonants used in the Devnagari script are categorized in Figure 2. It can be separately or modify consonants when pairing with diacritic marks (Matras). It includes both short and long forms, such as अ, आ, इ, ई, उ, ऊ, ऋ, ए, ऐ, ओ, औ, and औ. Consonants are changed into syllables by these matras, as in क + ा = का (ka + aa = kaa). On the contrary, consonants are sounds that must be expressed using a vowel. In Devanagari there are 33 basic consonants that are arranged according to their location and style of articulation. These include gutturals (e.g., क, ख, ग, घ), palatals (e.g., च, छ, ज, झ), cerebrals (e.g., ट, ठ, ड, ढ), dentals (e.g., त, थ, द, ध), and labials (e.g., प, फ, ब, भ). Additionally, semi-vowels like य, र, ल, व and sibilants like श, ष, स complement the system. In Devanagari, words and complex sounds form a harmonious interaction of vowels and consonants [4]. The conferences offer a structure and their systematic classification reflects the scientific grounding of the script, while the vowels form the basis of phonetic expression.

When combined, they capture the depth and adaptability of the Devanagari script, ideal for expressing different phonemes of Indian languages. For text recognition systems, such as OCR and NLP tools, the Devnagari vowel and consonant classification are extremely important. The vowels and consonants in the Devanagari scripts are categorized using different methods. For better results, the data file must be processed before classification. This includes a letter size, noise elimination and a picture binarization. The key first step is the extraction of elements, which is the process of converting unprocessed data into numerical inputs for classifiers. The Devanagari script has vowels and consonants of different shapes. The character identification can be helped by structural features such as strokes, loops and curves, as well as shape descriptors such as aspect ratio and pixel density [5]. Zoning calculates the pixel density in each of the zones created by dividing the character of the character. This helps to collect spatial information that distinguishes vowels from consonants. The boundaries of the characters are represented by characteristics based on contours that are useful for the distinction of complex shapes. This method is particularly useful for vowels because they often have more curves and diacritic brands than consonants. The character image's edge directional information is captured by Histograms of Oriented Gradients (HOG), which also calculates the gradients of pixel intensity. This method works well for differentiating between the more angular shapes of consonants and the rounded shapes of vowels.

Classical machine learning techniques can be used for classification after features have been retrieved. These methods, which differentiate vowels from consonants, have been effectively used for categorization purposes. Because Support Vector Machines (SVMs) can handle high-dimensional data, they are frequently utilized in text recognition applications. SVMs are appropriate for differentiating Devanagari characters based on extracted characteristics since they can efficiently classify nonlinear data by utilizing a kernel function [6]. K-new neighbors (K-Nn) classify entry according to the majority vote of their closest neighbors. He integrates successfully with the reduction of dimensions for categorizing vowels and consonants. Algorithms such as random forests and decision -making trees classify hand -written letters Devanagari by creating a model of decision rules from functions.

The classification of vowels and consonants in the Script of Devanagari is one of many text classification challenges that have been transformed by neural networks and deep learning. In the tasks of the image -based classification, the CNN showed remarkable efficiency. CNNS can automatically extract hierarchical functions from unprocessed image data. CNN can distinguish complex structures such as vowels, consonants, and ligatures in connection with the Devanagari script by combining low -level data, such as the edges and strokes in the early layers. Techniques like rotation, scaling, and translation can be employed to artificially increase the dataset, which improves the model's generalization to unknown data because deep network training necessitates vast volumes of data. Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNNs) are helpful methods for handling sequential data [7]. When it comes to differentiating specific vowels and consonants in related script forms, LSTM's ability to recall prior characters enables it to take context into account. Transfer learning may be improved for Devanagari vowel and consonant classification and used to develop for general picture classification problems. When there is a shortage of labeled data for training, transfer learning is useful. Improved performance may result from combining deep learning techniques with conventional feature-based methods. For example, CNNs or SVMs can be used in conjunction with hand-crafted features like zoning or HOG to increase classification accuracy, especially when training data is few. The classification of vowels and consonants in Devanagari script still faces several challenges, including the handling of ligatures, diacritics, and variable character spacing [8]. There are research gaps which can be treated as the scope for the future.

Despite the growing interest in Devanagari script recognition, there is a scarcity of publicly available, diverse, and large-scale datasets that include handwritten vowels, consonants, and numerals. Devanagari script is inherently complex, with characters often involving intricate strokes and overlapping patterns. Existing DL models struggle to differentiate between visually similar characters or address challenges posed by distorted or noisy handwritten inputs. Few studies address the practical deployment of classification systems for real-world applications, such as OCR tools for educational platforms, government documentation, or multilingual systems. Issues like scalability, latency, and robustness in diverse operating environments have not been sufficiently tackled. The multilingual context in not in focused and highlighted. Research often isolates Devanagari from other Indian scripts, neglecting the potential for cross-lingual OCR systems that can handle multiple scripts. Many existing approaches focus on achieving high accuracy without considering computational constraints. To address these issues, we have contribution for the classification purpose which includes:

- To design and implement deep learning models, specifically CNN and PNN, for the precise classification of handwritten vowels and consonants in the Devanagari script.
- The aim is to achieve high accuracy in recognizing and distinguishing complex character patterns in handwritten inputs.
- To evaluate and compare the performance of CNN and PNN models in terms of classification accuracy, MSE, computational efficiency, and generalization ability.
- To contribute to the development of robust OCR systems for the Devanagari script by creating a framework that efficiently handles handwritten character recognition.

The remaining sections of the article are organized in the following manner such as, Section II focused on the literature review utilized for the classification of vowel and consonant in the Devnagari script. Section III explores the dataset with its description and exploratory data analysis. Section IV defines the proposed mechanism utilized to build the system more effective. Section V discuss about the result section and comparative analysis of the proposed system with existing system. Section VI concludes the article with some future scope key points.

## 2. LITERATURE REVIEW

Many Indian languages, including Sanskrit, Hindi, Marathi and Nepal, benefit from the rich structure of the composition provided by the content of Devanagari. Research on Devanagari's character recognition has begun several decades ago, and since then some OCR algorithms have appeared to recognize Devanagari characters [9]. The area of handwritten analysis and document recognition (HDAR), which analyzes and recognizes manually written data, was born from ongoing research to increase the ability to read machines. Automated mailing, postcode reading, bank inspection data, and institutional records are some of the often used HDAR system applications [9]. Performance rate deteriorates if each step of the character recognition system cannot contribute effectively to the recognition process. Handwriting differences, poor image quality, varied character recognition methods, categorization issues, and other variables make handwritten character recognition an extremely difficult undertaking. Recent identification techniques are effective at identifying high-quality characters, but they struggle to identify characters that are overlapped, similar in shape, slanted, or skewed, which lowers the recognition rate. For printed and handwritten characters, Sharma et al. [10] introduced various preliminary processing (extermination, binarization, dilution and removal and removal), extraction of elements (vertical stripes, intersections, shirorekha touch, the number of endpoints and slopes of the end curves) and classification techniques. Their accuracy of printed characters was 93.33%, while their accuracy on handwritten characters was 72.72%. Scholars are also drawn to the historical records that museums have preserved. Because ancient writings sometimes contain overlapping characters, a thin algorithm may severely distort them. Additionally, if the characters are half or touching one another, the recognition rate—which mostly depends on character segmentation—is lowered.

In order to study these touching and half characters, Narang et al. [11] employed 5484 samples of a dataset of pre-segmented basic Devanagari characters from old manuscripts. Utilizing zigzag DCT features, they classified using decision tree, Naïve Bayes, and support vector machine (SVM) classifiers in conjunction with bagging and adaboost (adaptive boosting) to enhance recognition results. They experimented with various DCT feature lengths in conjunction with classifiers for recognition purposes. Modifiers and conjuncts are not recognized by their work. They created a brand-new method for separating lines, words, and characters from old records. They focused on the character's vertical and horizontal projection, average line height, and related elements using the zonewise projection profile approach. The image was separated into vertical zones of a set size. The horizontal projection profile (HPP) was computed for every vertical zone. Zonewise separation lines were taken into consideration for each row based on the threshold for each HPP. They were able to segment lines with 97% accuracy, characters with 98.5% accuracy, and touching and overlapping components with 96% and 100% accuracy, respectively. However, if the width of an isolated character is wide and the average line height of the character is not calculated appropriately, the algorithm's accuracy deteriorates.

Characters has two primary characteristics: statistical and structural. Character shape and overall structure are structural aspects, whereas centroid, intersection, open endpoints, and horizontal and vertical peak content are statistical features. Kumar et al. [12] with an accuracy rate of 92.18%, 84.67%and 86.79%. They have gained features based on intersections, open end points, horizontal peak range, land -use planning and diagonal elements. KNN (to the nearest neighbor), SVM and MLP classifiers were used for classification. They also mentioned a few issues, such as similar-shaped characters with the same meaning, choosing the right feature extraction techniques and extracting the right features, low-quality photos, handwriting in cursive, improper scanning, and binarization that reduces the text recognition rate. The study on classifying written Hindi words into the appropriate document categories was given by Puri et al. [13]. The model looks for and saves significant words, then creates word combinations with various modifiers to classify them. The frequency of occurrence for these words was determined and the paper was classed.

To recognize hand -written characters Imped Et al. [14] They examined a number of uniform and inconsistent (cut -based cuts based on shape and hierarchical) techniques. Bansal and Sinha [15] have created access

to segmentation and disintegration of composed characters. In order to extract structural attributes as elements, the composite character was divided into its basic symbols or individual basic characters. First, words were segmented and statistical data, such as the height and width of each separate word, was used to determine whether the character field contained composite characters. If this were the case, these compound characters were divided into separate characters. Using this method, they were able to segment characters with 85% accuracy. By using LSTM and two -way long -term memory (bltm) [16], it deals with an online manuscript of italic and uninhabited hand -written recognition for Devanagari and Bengal script. The middle zone is used for segmentation and extraction of elements after the word was divided into the upper, middle, and lower zones. Utilizing a simple stroke-based class labeling method, the RNN classifier achieved 99.50% accuracy. The authors discovered after completing the literature review that there has been relatively little research on handwritten Devanagari characters, with the majority of the data coming from office records and the postal service. Additionally, the data was produced solely for individual characters, asking participants to type their characters in a pre-defined box that provides each character a nearly uniform size. Because the characters are small and taken over from laptops, with many variants and for the same character, the authors of this work have built a new Devanagari character database from a very young age group that is difficult to work on. The authors designed a system of Devanagari character recognition for these characters, which combines several aspects and was evaluated with several classifiers.

## 3. DATASET DESCRIPTION

Ancient Brahmi is a source of phonetic script known as Devanagari. It serves as a basis for many Indian languages. The Hinde script of Devanagari is spoken of more than 342 million people around the world, and according to 2022 data, it is third in 45 languages. The Devanagari script contains about 11 vowels, 33 consonants and 10 numbers. The Devanagari script is written from left to right and lacks capital and lowercase letters. The Devnagari Handwritten Character dataset (DHCD) [17] consists of 44,000 pictures of handwritten numerals, vowels, and consonants in the Devanagari script. The size of the dataset ranges from 300 to 500 MB, contingent on the image file type and compression.

The dataset was created especially for CAPTCHA recognition and generating activities. In order to replicate handwriting variances, the dataset may contain enhanced characters (such as rotated, scaled, or translated) for CAPTCHA reasons. Three main types of characters from the Devanagari script are included in the dataset:

Vowels (Swar): Separate phonemes that can be used alone or in combination with consonants. The vowel set consists of 11 characters in total. For example: अ, आ, इ, ई, उ, ऊ, ऋ, ए, ऐ, ओ, औ.

Consonants (Vyanjan): Dependent phonetic units requiring vowels for articulation. There is total 33 characters in the consonant set. For example: क, ख, ग, घ, च, छ, ज, झ, ट, ठ, ड, ढ, त, थ, द, ध, प, फ, ब, भ, म, य, र, ल, व, श, ष, स, ह.

Numerals: The Devanagari script's numerical digits range from 0 to 9. The numeral set consists of nine characters in total. For example: ०, १, २, ३, ४, ५, ६, ७, ८, ९.

It is simple to process the photographs because they are saved in common file types like.png and.jpg. The grayscale format in which each character image is stored usually has a resolution of 28 by 28 pixels. This guarantees consistency and compatibility with models used in machine learning. There are roughly 1,000 examples in the dataset for every character, guaranteeing equal representation in every category. The dataset is structured into labelled folders for efficient use of machine learning algorithms. There are 11 subfolders comprises with vowels, 33 subfolders with consonants, and 10 subfolders with numerals. Each subfolder includes images of the respective handwritten character. The file naming convention typically follows a structure such as character_label_index.png (e.g., ka_001.png for the first sample of क). The dataset is available for download on IEEE data port that can be extracted on the local machine or server. It will be pre-processed, normalize the images to a standard format (e.g., pixel intensity range [0, 1]), and resize. The dataset is split in the distinct segments such as training, validation, and testing sets.

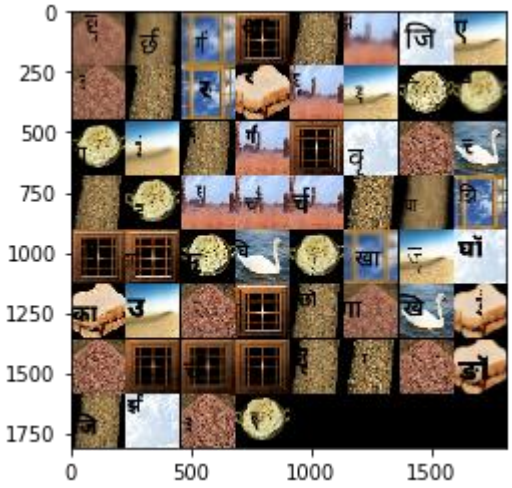



Fig. 3. Devnagari Characters for Captcha

Figure 3 represents a diverse set of Devanagari characters integrated with various natural, synthetic, or textured backgrounds, potentially for use in CAPTCHA generation or robust recognition tasks. Each grid cell contains a combination of a Devanagari script character overlaid on a unique background image or texture. The figure highlights how Devanagari characters can be incorporated into CAPTCHA challenges, requiring users to identify characters in visually complex settings. The purpose is to visualize and simulate real-world scenarios where text recognition systems must handle characters presented with noise, distortions, or non-uniform backgrounds. Both vowels and consonants from the Devanagari script are represented by the characters in the figure, and their arrangement over a range of backdrops simulates potential system issues such occlusion, overlapping, and background noise. Natural landscapes (such as skies, deserts, and bodies of water) as well as artificial settings (such as wooden panels or creative grids) and abstract textures (such as grains, patterns, or stones) are all included in the backdrops.

The ability to generalize the system in various visual situations and extract significant information for classification is tested by this variability. The distinct character and backdrop combinations in each cell point to a dataset created for testing CAPTCHA-solving or character recognition systems in difficult scenarios. The effort to make sure the dataset accurately depicts a range of scenarios is shown in this graphic, which improves the system's capacity to generalize to practical applications. This configuration's diversity serves a number of functions, including the construction of synthetic datasets, CAPTCHA generating, and resilient model training. The models can learn to focus on the character features with the removal of background images. Then combining real-world backgrounds with handwritten or machine-generated characters simulates real-world data while maintaining control over dataset diversity.

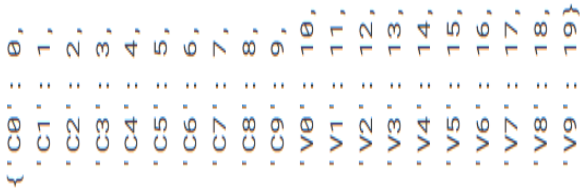



Fig. 4. Code generation for Numbers

Mean for the given data is [0.50745004, 0.46334055, 0.4228201]

Std for the given data is [0.32335126 0.31292862 0.3335115]

The mean and standard deviation (std) values represent the statistical properties of the dataset's pixel values across the three-color channels (likely corresponding to RGB). These values can be utilized for data normalization to ensure that input data has a consistent scale, which helps to improve the performance and stability of models. Where the average pixel value across the dataset for each channel represents the mean value. And the measure of how much the pixel values vary from the mean for each channel represents the standard deviation. The values are in a normalized range of [0, 1], which means the pixel values of the images in the dataset have likely been scaled down (e.g., divided by 255 if the original range was [0, 255]). The std values represent the spread or variation of pixel intensities for the red, green, and blue channels, respectively. A lower standard deviation indicates that pixel values are closely concentrated around the mean, while a higher standard deviation indicates greater variation.

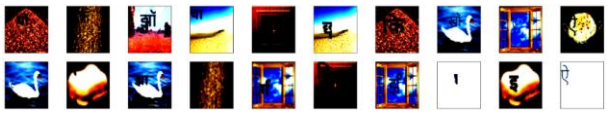


Fig. 5. Sample Image of DHCD dataset

Figure 5 represents a sample visualization of the DHCD dataset, showcasing individual handwritten characters from the Devanagari script overlaid on diverse backgrounds. The grid contains a variety of characters, including both vowels (Swar) and consonants (Vyanjan), which are presented on unique textured or natural backgrounds to simulate challenging recognition scenarios. It seems that each character in the data file is handwritten and emphasizes variations in writing styles, stroke thickness and orientation, reflecting the natural diversity found in handwritten scripts in the real world. The use of various backgrounds, including natural landscapes, such as sky, sand, and water, as well as abstract textures such as grains, wood, and artistic patterns, leads to a complex visual world.

Because of this configuration, the data file is particularly suitable for evaluating algorithms of recognition in noisy or inconsistent settings, such as CAPTCHA prompts, where background noise can go crazy or mix with characters. Accounting unpredictability of the data file is designed to evaluate and increase the durability of machine learning models for applications such as OCR and manually written character recognition. To ensure that the models trained on a data file can effectively generalize to invisible data in the real world, it combines several handwritten samples with a visually diverse background. The ability to train and evaluate systems for complex, noisy settings is illustrated by the visualization of the sample. It draws attention to the emphasis of the data file to maintain variability, allowing models to identify characteristics unique for each character and ignore the noise of the background from different confusing backgrounds. Applications such as multilingual OCR, CAPTCHA recognition, and intelligent systems that need reliable text processing in visually demanding environments depend on this type of data file.

## 4. PROPOSED METHODOLOGY

The methodological process, including preliminary data processing, model creation, training, evaluation, and testing, is necessary for deep categorization of vowels and consonants in DHCD. The use of algorithms of deep learning, vowels, and consonants in the Devnagari script are classified in several phases. The task can be completed in a series of steps (Figure 6) including:

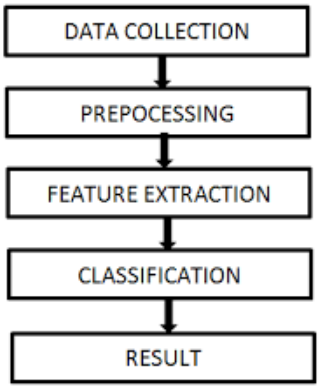



Fig. 6. proposed approach

*Data Collection and Pre-processing:* The DHCD data file must be loaded with handwritten vowels and consonants. It includes all 44 classes with 11 vowels and 33 consonants. The data augmentation techniques need to apply to increase the dataset variability and robustness, such as random rotation, translation, and scaling. These methods can include the noise or adjusting brightness and contrast. The pixel values must be normalized within the range [0, 1] or standardize using the dataset's mean and standard deviation by using an equation 1.

$$X_{normal} = \frac{x - \mu}{\sigma} \qquad (1)$$

Where x is the dataset variable, $\mu$ is the mean value, and $\sigma$ is the variance or standard deviation. Then split the dataset in train-test-validation for robust evaluation. The subsequent stage, feature extraction, uses a variety of feature extraction techniques to extract significant features from character images. In pattern recognition, one of the crucial steps is feature extraction [18]. The regional descriptors (Area, Solidity, BoundingBox, Centroid, ConvexArea, Eccentricity, EquivDiameter, EulerNumber, Extent, FilledArea, MajorAxisLength, MinorAxisLength) and moment invariants are utilized to extract the features of characters. With the use of these techniques, a dataset of size 6000 can produce a total of 14 characteristics for every character image.

*Model Architecture:* A deep learning model, typically a CNN, and PNN is designed for classification. The model can use popular deep learning frameworks like TensorFlow or PyTorch. The CNN take an image of a Devanagari character as input and output the recognized character. The architecture may include the input layer comprises with grayscale images, typically resized to 28x28 pixels. The next layer is feature extraction layer that extract spatial features from images using multiple filters (e.g., 32, 64 filters). A non-linear ReLU function is applied with the equation 2.

$$f(x) = \max(0, x) \qquad (2)$$

The down sample feature maps using max pooling to reduce spatial dimensions and computational complexity. Flatten layer is also utilized for converting the 2D feature map to 1D for robust classification. The fully connected dense layer with neurons is utilized to learn the complex patterns. To prevent overfitting, dropouts are applicable. A Softmax activation function is applied for the classification of 44 classes (equation 3).

$$P(Class_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \qquad (3)$$

Where classes of the characters are identified by using input of the $i^{th}$ class.

*Training of the Model:* The convolution layer is responsible for the learning of patterns identified from the input image of the text (Equation 4).

$$P_i = \sigma(X * W_i + b_i) \qquad (4)$$

Where, $P_i$ indicated the pattern of information, $\sigma$ is the activation function, X is the input image, $W_i$ convolutional filter of layer i, and $b_i$ bias term for layer i. The pooling layer can be reducing computational complexity and computed by using spatial dimensions of the feature maps. Where, $F_i$ represents the pooling operation in the case of average pooling or maximum pooling (Equation 5).

$$F_i = Pooling(P_i) \qquad (5)$$

While the fully connected layer evaluates the flattened output of the last pooling or convolutional layer. The fully connected layer (Eq. 3) of output layer fc can be evaluated in terms of $b_{fc}$ bias fully connected layer, $F_{flattened}$ indicates the flattened layer of information pattern, and $W_{fc}$ weight-based fully connected layer. The categorical loss for the classification of multi-class is evaluated with the equation 6.

$$Loss = -\sum_i y_i \log(\hat{y}_i) \qquad (6)$$

Where $y_i$ indicates the true label and $\hat{y}_i$ represents the predicted probabilities. After evaluation of the loss function, the model can be optimized with the help of distinct optimizers such as stochastic gradient descent (SGD), RMSprop, and Adam. The Adam optimizer is evaluated by using the equation 7.

$$\omega = \omega - \alpha.\frac{\delta L}{\delta \omega} \qquad (7)$$

Where $\omega$ represents the loss function with the slope of the loss function deviation. The batch size for the input data is considered 32 while the number of epochs is 50-100 epochs. The learning rate is reduced dynamically based on the validation losses. The performance can be monitored during the validation of the training to prevent overfitting. There are distinct performance parameters that are considered for the evaluation of performance such as precision, recall, and f1-score, accuracy of the model. Confusion matrix can be analyze for measuring the class-specific performance.

Probabilistic Neural Network (PNN) is also utilized for the classification of characters. Pattern recognition makes extensive use of this unique kind of feed-forward neural network. The inlet layer, layer of pattern, class layer and output layer are four layers that form a multilayer network with feed that PNN uses to structure their activities. There is no calculation in the input layer; It is only a distribution layer. The first layer provides the distance between the input vector and the input vectors of the training when the entry is present. This creates a vector with items that indicate the proximity of entry into the training input. Multi -dimensional cores are used by a layer of patterns, also known as a hidden layer and its neurons to estimate the probability density function. As a probability vector, it provides a contribution for each input class and leads to a net output. The summary layer is the third layer, also known as a layer of classes. Every character in Devanagari represents a class. As a result, there are 24 classes that correspond to 24 characters, 10 of which are numeric. The maximum of these probabilities is then selected by applying a transfer function to the second layer's output. Due to PNN classification, the output class is the recognized character [19] [20]. Figure 7 shows an illustration of the PNN's general architecture.

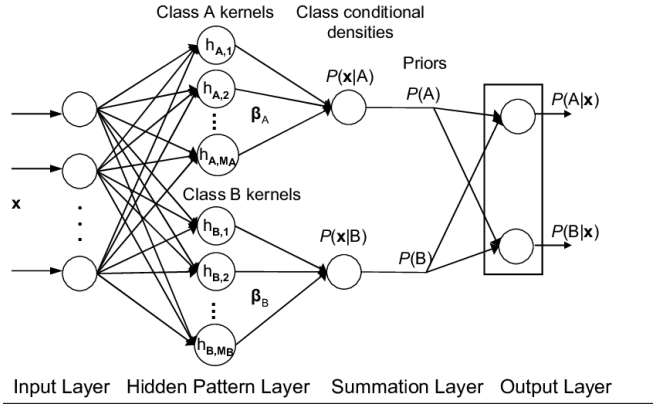



Fig. 7. PNN Architecture

*Testing and Deployment:* The model is tested on the trained model for unseen dataset to ensure the generalization. Hyperparameter tuning of the distinct attribute is also done for the optimization of the parameters such as learning rate, batch size, number of filters, and dropout rate using techniques. The model is deployed to train in CAPTCHA recognition and fine tune of the Devnagari characters.

## 5. RESULTS AND DISCUSSION

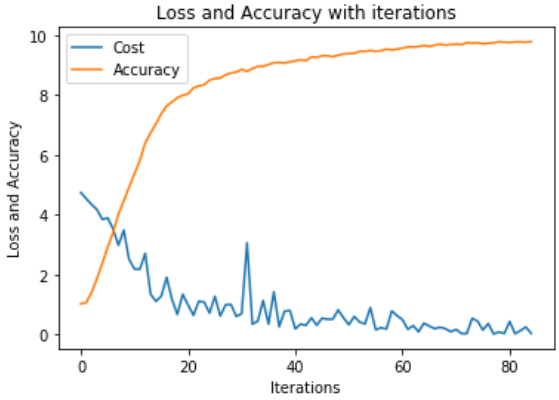
In this section, distinct results are discussed that can be utilized for the classification of vowel and consonant. Due to the wide range of font sizes and faces, character identification is always a difficult task. Therefore, the goal is to increase character recognition accuracy. A novel approach to the character recognition problem is made possible by the suggested method. To demonstrate the efficacy of the suggested system for character recognition in Devanagari scripts, CNN and PNN classifiers are used in the tests. Table I below shows the experimental results of 24 characters using a CNN classifier (12 no-bar characters, 2 middle-bar characters, and 10 numerals). It indicates distinct characters with training, testing, and cross validation accuracy by using CNN model. The average accuracy for the model is 97.95, 94.93, and 94.35 respectively.

Table I. Result of characters and digits using CNN Classifier

| SN | Input Character | Training Accuracy | Testing Accuracy | Cross Validation |
|---|---|---|---|---|
| 1 | इ | 98.69 | 89.66 | 85.71 |
| 2 | उ | 96.62 | 91.52 | 97.50 |
| 3 | ए | 98.78 | 84.48 | 79.41 |
| 4 | क | 95.74 | 94.74 | 89.65 |
| 5 | फ | 100 | 100 | 94.87 |
| 6 | छ | 97.57 | 96.55 | 88.89 |
| 7 | ट | 96.72 | 94.12 | 94.44 |
| 8 | ठ | 97.16 | 93.10 | 100 |
| 9 | ड | 98.50 | 96.30 | 94.12 |
| 10 | ढ | 92.95 | 89.85 | 92.31 |
| 11 | द | 95.16 | 90.14 | 100 |
| 12 | र | 97.95 | 95.59 | 100 |
| 13 | ह | 100 | 100 | 97.05 |
| 14 | ळ | 98.68 | 96.67 | 97.05 |

| | | | | |
|---|---|---|---|---|
| 15 |  | 97.48 | 92.42 | 94.44 |
| 16 |  | 98.89 | 98.04 | 96.00 |
| 17 |  | 98.67 | 97.30 | 93.55 |
| 18 |  | 100 | 98.56 | 98.56 |
| 19 |  | 97.57 | 92.54 | 93.94 |
| 20 |  | 99.87 | 98.41 | 93.10 |
| 21 |  | 99.47 | 98.43 | 93.02 |
| 22 |  | 98.27 | 97.26 | 100 |
| 23 |  | 97.89 | 96.88 | 97.78 |
| 24 |  | 98.25 | 96.22 | 93.02 |
| **Average Accuracy** | | **97.95** | **94.93** | **94.35** |

The figure 8 depicts a training curve that plots cost (loss) and accuracy over the training epochs during the training of a deep learning model.




Train Accuracy: 97.94733825683594

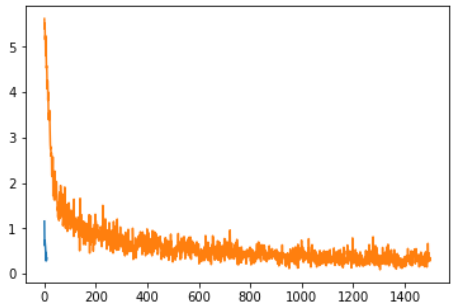
Fig. 8. Accuracy Vs cost

The losses or cost measures the difference between the model's predictions and the actual labels. A decreasing trend in the loss curve indicates that the model is learning and minimizing errors during training. The occasional spikes in the loss curve may occur due to sudden updates in weights during optimization. And Stochastic gradient descent or mini-batch training variations. The accuracy of the model evaluated as the percentage of correct predictions made by the model. The figure shows a steady increase over epochs, indicating that the model is improving in its classification task as training progresses. The classes are identified according to the ImageID of the characters (Figure 9).

| | ImageId | Class |
|---|---|---|
| 0 | 7851.png | V7_C8 |
| 1 | 4481.png | V4_C4 |
| 2 | 7561.png | V7_C5 |
| 3 | 4286.png | V8_C2 |
| 4 | 4730.png | V4_C7 |

| | ImageId | Class |
|---|---|---|
| 7391 | 9995.png | V7_C0 |
| 317 | 9996.png | V9_C9 |
| 6757 | 9997.png | V9_C9 |
| 626 | 9998.png | V9_C9 |
| 9762 | 9999.png | V0_C7 |

Fig. 9. ImageID correspondence with Class




CPU time: user 9min 36s, sys: 5min 29s, total: 15min 5s

Wall time: 15min 9s

Fig. 10. Losses and Training losses of the Model

The code execution took a total CPU time of 15 minutes and 5 seconds (9 minutes 36 seconds in user space and 5 minutes 29 seconds in system space), while the actual elapsed time (wall time) was 15 minutes and 9 seconds. The figure 10 illustrates the progression of a loss function (Training and validation loss) over the course of training epochs for CNN model. The vertical axis represents the loss, which quantifies the error between the predicted outputs and the actual target values. The horizontal axis represents the epochs, indicating the number of iterations over the entire training dataset. The curve represents the training loss, and its fluctuations suggest the use of stochastic gradient descent (SGD) or mini-batch training, where each batch causes small variations in the loss. The loss starts at a high value, indicating that the model's initial predictions are far from the true target values. A steep decline is observed in the early epochs, signifying rapid learning as the model adjusts its weights significantly in response to the loss. The loss continues to decrease but at a slower rate compared to the initial phase. This indicates that the model is approaching convergence and making smaller updates to its parameters. The loss stabilizes at a relatively low value, suggesting that the model has nearly converged and the training process has reached its optimal state. The consistent small fluctuations in the orange curve could be due to noise introduced by mini-batch SGD, which is common in training deep learning models.

Performing Metrics

| | Training | | | Cross Validation | | | Testing | | |
|---|---|---|---|---|---|---|---|---|---|
| Model Name | MSE | r | Correct | MSE | r | Correct | MSE | r | Correct |
| PNN | 0.000144 | 0.998219 | 99.78% | 0.001069 | 0.990502 | 96.78% | 0.001761 | 0.975215 | 96.07% |

Performance Metrics

| | Training | Cross Val. | Testing |
|---|---|---|---|
| # of Rows | 3600 | 900 | 1500 |
| MSE | 0.000144 | 0.001069 | 0.001761 |
| Correlation (r) | 0.998219 | 0.990502 | 0.975215 |
| # Correct | 3592 | 871 | 1441 |
| # Incorrect | 8 | 29 | 59 |
| % Correct | 99.78% | 96.78% | 96.07% |

Fig. 11. Performance Metrics for PNN model

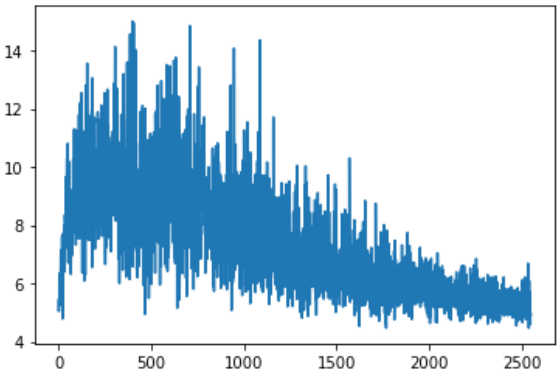



Fig. 12. Training and Testing Losses for PNN model

The PNN model demonstrates outstanding performance in training, cross-validation, and testing phases, with low error rates, high correlation coefficients, and high accuracy percentages. This suggests that the model is well-trained, generalizes effectively, and performs reliably on unseen data. The figure 11 presents performance

metrics for a PNN model evaluated on training, cross-validation, and testing datasets. The figure includes the following metrics for the three phases (Training 99.78, Cross-Validation 96.78, and Testing 96.07). Mean squared error (0.000144) quantifies the error between predicted and actual values for training, testing, and cross validation. The MSE indicates the very low prediction error during training phase. The correlation coefficient measures the strength and direction of the linear relationship between predictions and actual outputs. The value of correlation coefficient 0.998219, suggesting an almost perfect linear relationship between predicted and actual values. The correct represents the number of correctly classified samples. The 3592 samples were classified correctly out of 3600 number of rows which 99.78% samples are correctly classified.

For the cross-validation phase, 900 samples were used to assess the model's ability to generalize. The MSE value is 0.001069, slightly higher than training, which is expected as this dataset is unseen during training. The correlation coefficient is 0.990502, shows very strong but slightly lower than the training phase. While, 871 samples are correctly classified out of 92 samples that represents 96.78%, showing a slight decline compared to the training phase but still very strong. In the 1500 samples of testing phase, 1441 samples are correctly classified. The MSE value 0.001761 is slightly higher than both training and cross-validation. It indicates 96.07% samples are good generalization and robust performance. The average recognition accuracy achieved with CNN and PNN classifiers is 94.93% and 96.07%, respectively. When Faisal et al. [21] utilized an ANN classifier, they were able to recognize printed letters without bars with an average recognition rate of 70.27%. The average recognition rate of all Devanagari characters, as determined by Bhagat et al. [22] using ANFIS, is about 95%. The findings of the research includes greater recognition of Devanagari with the proposed methods.

## 6. CONCLUSION

Classification of consonants and vowels in the Devanagari script using CNN and PNN shows the potential of deep learning and probability approaches to accurately recognize complex handwritten characters. The high accuracy of the classification is achieved by the CNN method, which is known for its ability to extract spatial properties through the convolution. It successfully captures complex patterns and distinctive shapes of Devanagari characters. It is a strong model for the tasks of character recognition due to its hierarchical extraction of elements, which ensures that small variations are identified among related features. On the other hand, PNN depends on the statistical characteristics of input data and uses Bayes' rule and its probability nature to accurately categorize characters. The finding shows how well the two models work, with CNN excellent in learning complicated features through iterative optimization and PNN providing faster training because it is irrelevant. Experimental findings demonstrate the reliability of models by showing that during training, cross validation, and testing phase they will achieve excellent accuracy with low MSE coefficients and high correlation coefficients. The ability of models to effectively generalize to unknown data is confirmed by the accuracy of training, which continuously over 98%, and cross validation and testing phases that create accuracy exceeding 96%. Comparative analysis also shows that PNN offers computing effective replacement for data files with moderate complexity, while CNN is better to extraction of elements and can manage larger data sets. In addition, the results emphasize how important it is to set the hyperparameter to maximize the power of the model. The advantages of these two models are emphasized by a small difference in performance measures, which means that the decision between CNN and PNN is based on specific application requirements such as computing restrictions or data file complexity.

**References**


1. S. Singh, A. Sharma, and V. K. Chauhan, “Indic script family and its offline handwriting recognition for characters/digits and words: a comprehensive survey,” Artificial Intelligence Review, vol. 56, no. S3. Springer Science and Business Media LLC, pp. 3003–3055, Sep. 19, 2023. doi: 10.1007/s10462-023-10597-y.
2. H. Kaur and M. Kumar, “A comprehensive survey on word recognition for non-Indic and Indic scripts,” Pattern Analysis and Applications, vol. 21, no. 4. Springer Science and Business Media LLC, pp. 897–929, Jul. 25, 2018. doi: 10.1007/s10044-018-0731-2.
3. M. Sethi, M. Kumar, and M. K. Jindal, “Forensic handwriting analysis: a hybrid classification framework for writer identification in Devanagari script,” Multimedia Tools and Applications. Springer Science and Business Media LLC, Jan. 03, 2025. doi: 10.1007/s11042-024-20574-4.
4. S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, “A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning,” Archives of Computational Methods in Engineering, vol. 27, no. 4. Springer Science and Business Media LLC, pp. 1071–1092, Jun. 01, 2019. doi: 10.1007/s11831-019-09344-w.
5. A. Chahi, Y. El merabet, Y. Ruichek, and R. Touahni, “WriterINet: a multi-path deep CNN for offline text-independent writer identification,” International Journal on Document Analysis and Recognition (IJDAR), vol. 26, no. 2. Springer Science and Business Media LLC, pp. 89–107, Oct. 14, 2022. doi: 10.1007/s10032-022-00418-3.
6. S. T. Nabi, M. Kumar, and P. Singh, “DeepNet-WI: a deep-net model for offline Urdu writer identification,” Evolving Systems, vol. 15, no. 3. Springer Science and Business Media LLC, pp. 759–769, Apr. 21, 2023. doi: 10.1007/s12530-023-09504-1.
7. Ms. S. Tantarpale and C. N. Deshmukh, “An Approach to Pattern Recognition for Identification of Devnagari Script Based on Fingertips and Palm,” Journal of Physics: Conference Series, vol. 2327, no. 1. IOP Publishing, p. 012032, Aug. 01, 2022. doi: 10.1088/1742-6596/2327/1/012032.
8. M. Yadav, R. K. Purwar, and M. Mittal, “Handwritten Hindi character recognition: a review,” IET Image Processing, vol. 12, no. 11. Institution of Engineering and Technology (IET), pp. 1919–1933, Nov. 2018. doi: 10.1049/iet-ipr.2017.0184.

9. Chaudhuri A, Mandaviya K, Badelia P and Ghosh S K 2017 Optical character recognition systems In: Proceedings of the Optical Character Recognition Systems for Different Languages with Soft Computing 9–41.
10. Sharma, R., Mudgal, T. (2019). Primitive Feature-Based Optical Character Recognition of the Devanagari Script. In: Panigrahi, C., Pujari, A., Misra, S., Pati, B., Li, KC. (eds) Progress in Advanced Computing and Intelligent Engineering. Advances in Intelligent Systems and Computing, vol 714. Springer, Singapore. https://doi.org/10.1007/978-981-13-0224-4_23.
11. S. R. Narang, M. K. Jindal, and M. Kumar, "Devanagari ancient character recognition using DCT features with adaptive boosting and bootstrap aggregating," Soft Computing, vol. 23, no. 24. Springer Science and Business Media LLC, pp. 13603–13614, Mar. 08, 2019. doi: 10.1007/s00500-019-03897-5.
12. S. M. Pande and B. K. Jha, "Character Recognition System for Devanagari Script Using Machine Learning Approach," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC). IEEE, Apr. 08, 2021. doi: 10.1109/iccmc51019.2021.9418028.
13. A. El-Sawy, H. EL-Bakry, and M. Loey, "CNN for Handwritten Arabic Digits Recognition Based on LeNet-5," Advances in Intelligent Systems and Computing. Springer International Publishing, pp. 566–575, Oct. 18, 2016. doi: 10.1007/978-3-319-48308-5_54.
14. M. Sethi, M. Kumar, and M. K. Jindal, "Gender prediction system through behavioral biometric handwriting: a comprehensive review," Soft Computing, vol. 27, no. 10. Springer Science and Business Media LLC, pp. 6307–6327, Feb. 16, 2023. doi: 10.1007/s00500-023-07907-5.
15. Adnan, M., Rahman, F., Imrul, M., Al, N., & Shabnam, S., "Handwritten Bangla character recognition using inception convolutional neural network", Int. J. Comput. Appl, 181(17), 48-59, 2018.
16. B. K. Y. Panchal and A. Shah, "NLP-Based Spellchecker and Grammar Checker for Indic Languages," Natural Language Processing for Software Engineering. Wiley, pp. 43–70, Jan. 10, 2025. doi: 10.1002/9781394272464.ch4.
17. Sanjay Pate, Rakesh Ramteke, October 6, 2022, "Handwritten Devanagari Characters Dataset –(Vowels, Consonants and Numerals) of 44,000 images for Devanagari CAPTCHA Generation and Recognition.", IEEE Dataport, doi: https://dx.doi.org/10.21227/9zpv-3194.
18. J. Bati and P. Raj Dawadi, "Ranjana Script Handwritten Character Recognition using CNN," JOIV : International Journal on Informatics Visualization, vol. 7, no. 3. Politeknik Negeri Padang, p. 984, Sep. 10, 2023. doi: 10.30630/joiv.7.3.1725.
19. V. V. S. Dingari, G. Kosanam, D. S. S. Chavatapalli, and C. N. Devi, "A Printed Character Recognition System for Meetei-Mayek Script Using Transfer Learning," Lecture Notes in Electrical Engineering. Springer Nature Singapore, pp. 519–528, Oct. 03, 2023. doi: 10.1007/978-981-99-4713-3_50.
20. J. Talebi and Z. Azizi, "Enhancing land feature classification with the BTR Extractor: A novel software package for high-accuracy analysis of aerial laser scan data," MethodsX, vol. 14. Elsevier BV, p. 103090, Jun. 2025. doi: 10.1016/j.mex.2024.103090.
21. A. N. M. Fahim Faisal, Md. A. Rahman, and T. Farah, "A Rule-Based Bengali Grammar Checker," 2021 Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4). IEEE, pp. 113–117, Jul. 29, 2021. doi: 10.1109/worlds451998.2021.9514031.
22. B. Bhagat and M. Dua, "Improved spell corrector algorithm and deepspeech2 model for enhancing end-to-end Gujarati language ASR performance," e-Prime - Advances in Electrical Engineering, Electronics and Energy, vol. 7. Elsevier BV, p. 100441, Mar. 2024. doi: 10.1016/j.prime.2024.100441.