IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, May 2025
Impact Factor: 7.67

Web Based Tamil Handwritten Recognition using Machine Learning

Rekha M[1], Srinivas R[2], Rishikesavan A[3], Thirumalai S[4] , Abubaker Sithik Rahuman M[5]
Assistant Professor , Department of Computer Science Engineering[1]
Student , Department of Computer Science Engineering[2,3,4,5]
Anjalai Ammal Mahalingam Engineering College, Thiruvarur, Tamil Nadu, India

Abstract: *A Convolutional Neural Network (CNN) is developed to recognize isolated Tamil handwritten characters. The preprocessing steps include converting images to grayscale, reducing noise, applying adaptive thresholding, and performing segmentation. The model is created and trained with Torch, and it is incorporated into a web application utilizing React, Bootstrap and Flask for managing files and outputting results. The recognized characters are transformed into Unicode. The final goal is to achieve accurate sentence-level recognition using Transformers*

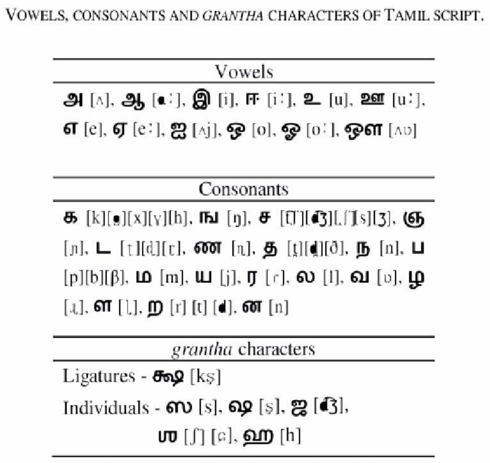
Keywords: Tamil OCR, Handwritten Recognition, CNN, Deep Learning, PyTorch, Flask, Image Processing, React.

I. INTRODUCTION

In the digital age, the task of transforming handwritten material into text that computers can read has gained significant, particularly for safeguarding regional languages such as Tamil. As one of the oldest and most widely spoken classical languages, Tamil features a distinct script characterized by complex characters and various handwriting styles. Conventional Optical Character Recognition (OCR) technologies frequently encounter difficulties in accurately recognizing these handwritten characters because of their intricacies and inconsistencies

This project introduces a deep learning method aimed at identifying Tamil handwritten characters with the help of a Convolutional Neural Network (CNN). The system processes an image of a single handwritten Tamil character, adjusts its size and alignment, and then classifies it with a high level of accuracy. CNNs are particularly effective for this purpose, as they can autonomously learn and identify key visual attributes from images without requiring manual input.

To ensure the model is easy to access and use, a web application has been created utilizing Flask for the backend and React with Bootstrap for the frontend. Through the web interface, users can upload an image of a handwritten Tamil character and receive an immediate prediction along with a confidence score. The image processing workflow includes



VOWELS, CONSONANTS AND *GRANTHA* CHARACTERS OF TAMIL SCRIPT.

Vowels
அ [ʌ], ஆ [ɑː], இ [i], ஈ [iː], உ [u], ஊ [uː], எ [e], ஏ [eː], ஐ [ʌj], ஒ [o], ஓ [oː], ஔ [ʌʋ]
Consonants
க [k][ɡ][x][ɣ][h], ங [ŋ], ச [tʃ][dʒ][ʃ][s][ʒ], ஞ [ɲ], ட [ʈ][ɖ][ɽ], ண [ɳ], த [t̪][d̪][ð], ந [n], ப [p][b][β], ம [m], ய [j], ர [ɾ], ல [l], வ [ʋ], ழ [ɻ], ள [ɭ], ற [r] [t] [d], ன [n]
***grantha* characters**
Ligatures - க்ஷ [kʂ]
Individuals - ஸ [s], ஷ [ʂ], ஜ [dʒ], ஶ [ʃ] [ɕ], ஹ [h]

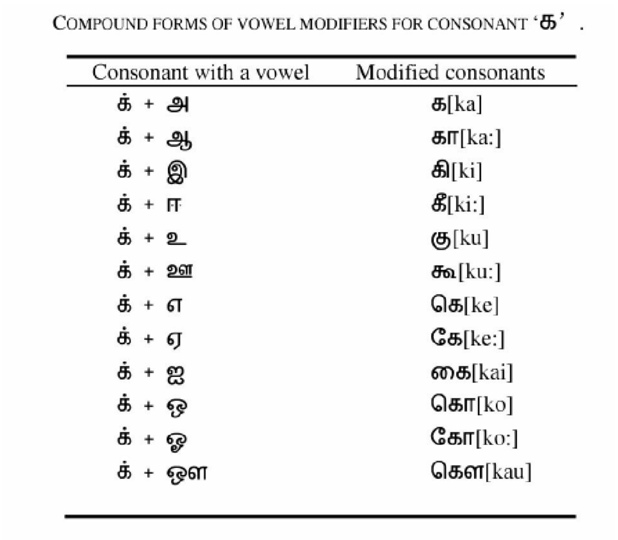
resizing, centering the content, and converting it into a compatible format for the model using libraries such as Pillow, NumPy, and SciPy.

By integrating machine learning with web-based technologies, this project not only simplifies the recognition of Tamil characters but also champions the use of AI in the preservation and processing of regional languages. In the future, the system could be enhanced to recognize entire words or sentences, thereby broadening its applications in digital archiving, educational resources, and language digitization.

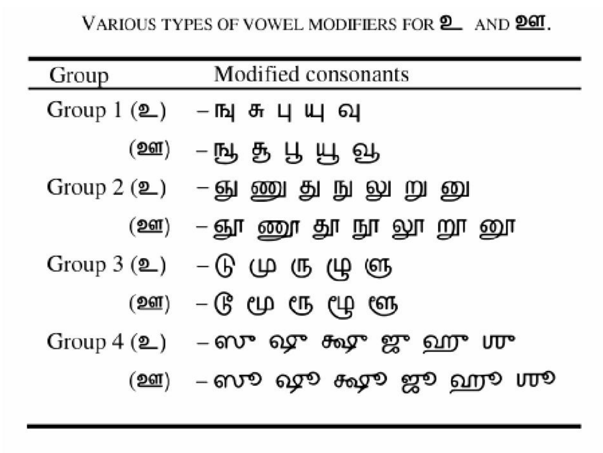
II. TAMIL SCRIPT

The Tamil script is classified as an abugida, also referred to as an alphasyllabary writing system, where sequences of consonants and vowels are represented as a single unit. This script comprises 12 vowels and 18 consonants, along with a unique character known as ayudham, which is categorized in Tamil grammar as neither a consonant nor a vowel. However, it has evolved to be used as a diacritic symbol to indicate foreign sounds.

Tamil is written horizontally from left to right. Each consonant in Tamil is written with a default inherent vowel [a]. Modifiers of the other vowels can be added to the right, left or both sides of the consonants. Pulli, the marker to suppress the inherent vowel is written on top of consonants.


COMPOUND FORMS OF VOWEL MODIFIERS FOR CONSONANT 'க' .

Consonant with a vowel	Modified consonants
க் + அ	க[ka]
க் + ஆ	கா[ka:]
க் + இ	கி[ki]
க் + ஈ	கீ[ki:]
க் + உ	கு[ku]
க் + ஊ	கூ[ku:]
க் + எ	கெ[ke]
க் + ஏ	கே[ke:]
க் + ஐ	கை[kai]
க் + ஒ	கொ[ko]
க் + ஓ	கோ[ko:]
க் + ஔ	கௌ[kau]


VARIOUS TYPES OF VOWEL MODIFIERS FOR உ AND ஊ.

Group	Modified consonants
Group 1 (உ)	– ஙு சு பு யு வு
(ஊ)	– ஙூ சூ பூ யூ வூ
Group 2 (உ)	– ஞு ணு து நு லு று னு
(ஊ)	– ஞூ ணூ தூ நூ லூ றூ னூ
Group 3 (உ)	– டு மு ரு ழு ளு
(ஊ)	– டூ மூ ரூ ழூ ளூ
Group 4 (உ)	– ஸு ஷு க்ஷு ஜு ஹு ஶு
(ஊ)	– ஸூ ஷூ க்ஷூ ஜூ ஹூ ஶூ

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, May 2025
Impact Factor: 7.67

II. LITERATURE REVIEW

Year	Title of Paper	Authors	Journal/Conference	Key Focus
2024	Tamil and English handwritten character segmentation and recognition using deep learning	Varshini C, Yogeshwaran S, Mekala, V	2024 International Conference on Communication, Computing and Internet of Things (IC3IoT)	Deep learning-based segmentation and recognition for bilingual handwritten characters
2024	Text detection and style classification from images using Vision Transformer and Transformer Decoder	Yajima H, Leow C. S, Nishizaki H	2024 IEEE 13th Global Conference on Consumer Electronics (GCCE)	Image-based text detection and style classification using Transformer models
2023	Categorical boosting machine for Tamil character recognition using shape-based features	Nevatia, I, Mishra T. K	2023 7th International Conference on Computing, Communication, Control and Automation (ICCUBEA)	Tamil character recognition using CatBoost and shape-based features
2023	Recognition of Tamil handwritten characters using Scrabble GAN	Sasipriyaa N, Natesan P, Gothai E, Madhesan G, Madhumitha E, Mithun K. V	2023 International Conference on Computer Communication and Informatics (ICCCI)	GAN-based model for handwritten Tamil character recognition
2023	An efficient unsupervised approach for OCR error correction of Vietnamese OCR text	Nguyen Q. D, Phan N. M, Krömer P, Le D. A	IEEE Access	Unsupervised OCR error correction approach for Vietnamese, adaptable for regional OCR systems
2022	Handwritten Tamil character recognition	Harini M. L, et al.	International Journal of Health Sciences	General overview and implementation of Tamil handwritten character recognition
2022	Identification of Tamil characters using deep learning	Akashkumar S, Dyaram A. N, Anand M.	Machine Learning and Autonomous Systems, Springer	Deep learning techniques for identifying Tamil characters
2019	Recognition of Tamil handwritten character using modified neural network with aid of elephant herding optimization	Kowsalya S, Periasamy P. S	Multimedia Tools and Applications	Optimization-based neural network for Tamil character recognition

IV. SYSTEM ARCHITECTURE

The system proposed for recognizing Tamil handwritten characters is built on a modular framework that combines deep learning classification with a web-based user interface.
It consists of four primary elements input management and preprocessing, the Convolutional Neural Network (CNN) model, a backend server utilizing Flask, and a frontend interface created with React and Bootstrap.

Input Management and Preprocessing

The process starts with the user uploading an image of a single handwritten Tamil character via the interface. To maintain consistent input dimensions and enhance model accuracy, the image undergoes preprocessing before it is fed into the neural network.

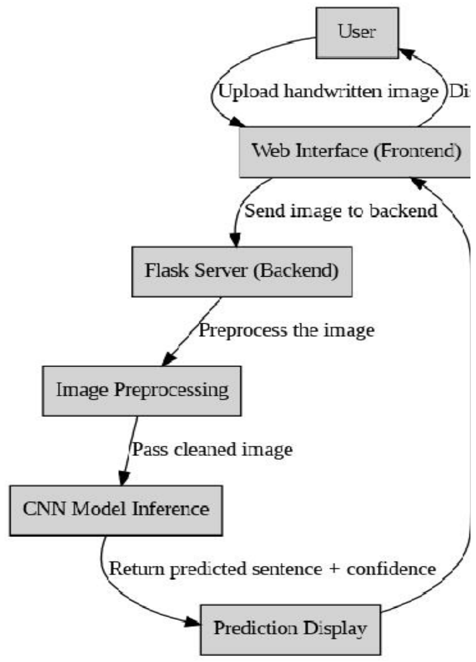
This stage includes:

Resizing the image to a specific resolution suitable for the CNN (for instance, 28×28 or 32×32 pixels) utilizing the Pillow library.

Centering the character in the image by calculating the center of mass and adjusting the pixel data accordingly, which employs functions from the scipy.ndimage module.

Converting the image to a single-channel grayscale format and normalizing it with NumPy arrays to scale pixel intensities, thereby promoting stable model performance.

This preprocessing step ensures that the CNN receives uniform input, regardless of variations in handwriting styles, image sizes, or orientations.



Convolutional Neural Network (CNN) Model

At the heart of the system lies a CNN developed with PyTorch. The model is intended to autonomously extract features hierarchically from the input image and carry out character classification. Its architecture includes:

Six convolutional layers that utilize 3×3 kernels along with ReLU activation functions to identify local patterns like curves, edges, and strokes typical of Tamil script.

Batch normalization layers to stabilize learning and speed up convergence.

Three max-pooling layers that down sample feature maps while preserving the most critical features and reducing computational demands.

A flattening layer that transforms multi-dimensional feature maps into a one-dimensional vector.

Two fully connected layers, containing 1024 and 512 neurons respectively, further modifying features for the ultimate classification.

A concluding softmax layer that provides probabilities for each character category, allowing the model to predict the most likely Tamil character with a corresponding confidence score.

Backend Server (Flask)

The backend, built on Flask, acts as the bridge between the frontend interface and the machine learning model, performing the following tasks:

Receiving image inputs from the frontend via HTTP requests.

Calling the preprocessing module to adapt the image into a CNN-compatible format.

Running the trained CNN model to generate predictions.

Sending the predicted character along with its confidence score back to the frontend in a structured format (like JSON).

This streamlined backend facilitates effective communication and model inference while avoiding unnecessary overhead.

Frontend Interface (React and Bootstrap)

The user interface is crafted using React for dynamic content rendering and Bootstrap for responsive design. It offers a straightforward platform for users to upload images of handwritten characters and receive real-time predictions from the model. The frontend is designed to be accessible and user-friendly, especially for individuals who may not have a technical background.

Data Flow Summary

The workflow of the system can be outlined as follows:

A user uploads an image of a handwritten character through the frontend interface.

The image is forwarded to the Flask backend for processing.

After preprocessing, the image is sent to the CNN model for classification.

The predicted character and its confidence score are returned to the frontend.

The outcome is then presented to the user.

This modular architecture is crafted to be efficient and adaptable, with potential future extensions for recognizing complete words or sentences in Tamil handwriting. It lays a scalable groundwork for deploying intelligent handwriting recognition systems for regional languages.

V. METHODOLOGY

This section describes the systematic procedure undertaken for developing the Tamil Handwritten Character Recognition system. The process includes managing datasets, preprocessing images, designing and training models, and deploying them via a web application. The methodology is divided into the following stages:

Dataset Collection and Preparation

The system is trained on a specially curated dataset that contains isolated Tamil handwritten characters. Each image in this dataset illustrates a single character written by different individuals to capture a variety of handwriting styles. The dataset is pre-labeled and organized according to the corresponding Unicode representations of Tamil script.

To ensure consistency, the raw images are standardized in resolution and format before being utilized for training.

Images undergo filtering to eliminate noise, incomplete samples, and characters that fall outside the designated area of interest.

Image Preprocessing

Efficient preprocessing is essential to ensure that the CNN model is provided with clean, uniform, and accurately aligned input data. The preprocessing workflow consists of the following steps:

- *Resizing:* Each image is resized to set dimensions (for example, 32×32 pixels) using the Pillow library. This standardization guarantees compatibility with the CNN's input layer.

- *Grayscale Conversion:* Color images are converted to grayscale, which simplifies input to a single channel and reduces computational costs without losing key information necessary for recognizing handwritten strokes.
- *Centering:* To accommodate variations in character placement, the character is centered within the image using scipy.ndimage.center_of_mass(). This reduces positional bias in the dataset.
- *Normalization:* Pixel values are scaled to the range of [0, 1] to facilitate smoother convergence during training, implemented through NumPy.

The processed images are maintained in NumPy arrays for use in model training and inference.

CNN Model Design

A deep Convolutional Neural Network (CNN) is constructed to capture spatial features from the preprocessed character images and classify them into predefined Tamil character categories. The model architecture comprises:

- *Convolutional Layers:* Six convolutional layers featuring 3×3 filters and ReLU activation functions, which identify visual patterns like curves, edges, and corners typical of handwritten Tamil characters.
- *Batch Normalization:* This is applied after select convolutional layers to stabilize and speed up the training process.
- *Max Pooling:* Three max pooling layers are utilized to reduce the spatial dimensions of the feature maps while preserving significant features.
- *Fully Connected Layers:* After flattening the output from the convolutional layers, two fully connected layers with 1024 and 512 neurons respectively process the extracted features.
- *Softmax Classifier:* The last layer employs a softmax function to produce a probability distribution across the character classes. The class with the highest probability is taken as the predicted output.

The model is implemented in PyTorch and trained using a cross-entropy loss function along with optimizers like Adam or SGD. Training occurs over multiple epochs, with validation to monitor overfitting and performance.

Model Evaluation

Following training, the model is assessed using a separate test set. Metrics such as accuracy, precision, recall, and F1-score are utilized to evaluate its performance. Confusion matrices are generated to pinpoint classes that exhibit high misclassification rates. The model's confidence scores are analyzed to understand prediction certainty.

Web Application Integration

To enhance user accessibility to the recognition system, it is deployed as a web application. The deployment architecture features:

- *Backend (Flask):* A lightweight Flask server manages image uploads from users, carries out preprocessing, invokes the trained model for predictions, and returns the results.
- *Frontend (React & Bootstrap):* The user interface enables individuals to upload images of handwritten characters and view predictions. Built with React for component-based rendering and styled with Bootstrap for responsiveness.

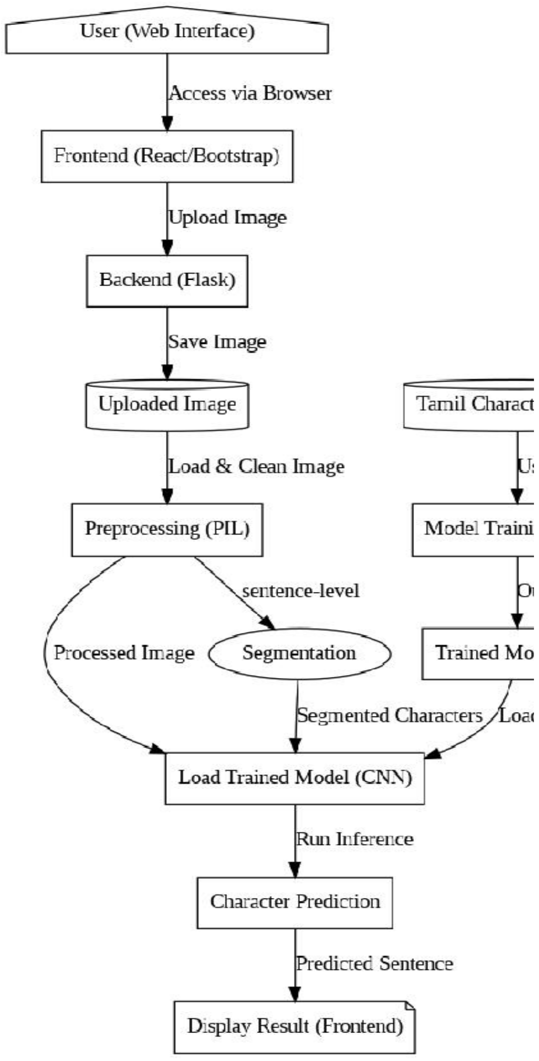
Prediction Flow:

- The user uploads an image through the frontend.
- The image is sent to the Flask backend.
- Preprocessing and model inference are executed.
- The predicted character and its confidence score are returned and displayed on the frontend.

Deployment and Usability Testing

The system undergoes testing for responsiveness, user-friendliness, and performance under various input conditions (e.g., illegible handwriting, low-contrast images). Usability feedback is gathered to evaluate interactions with the system by non-technical users, and adjustments are made based on this feedback.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, May 2025
Impact Factor: 7.67



Implementation

The Tamil Handwritten Character Recognition system is structured into three primary components: model training, backend API development and frontend user interface This system is with an emphasis onity and effectiveness, utilizing open-source tools and libraries.

Model Training

At the heart of this system lies a Convolutional Neural Network (CNN), created with the PyTorch deep learning framework. The training dataset comprises images of separate handwritten Tamil characters, showcasing a wide range of writing styles. Each image is resized to 32×32 pixels and converted to grayscale prior to being input into the model.
The architecture of the CNN includes:
Six convolutional layers utilizing 3×3 kernels and ReLU activation functions.
Batch normalization after certain convolutional layers to enhance learning stability.
Three max pooling layers to minimize spatial dimensions.
Two fully connected layers containing 1024 and 512 neurons, respectively.
A concluding softmax classification layer with 156 output nodes, corresponding to Tamil characters.

Training employs the cross-entropy loss function and the Adam optimizer, set to a learning rate of 0.001. The model undergoes training for several epochs with a batch size of 64 and implements early stopping based on validation loss.

Preprocessing Pipeline

Effective preprocessing is essential for maintaining consistency in the input data. Using Pillow, NumPy, and scipy.ndimage, the preprocessing pipeline executes the following tasks:
Resizing images to a standard dimension.
Converting RGB images into grayscale.
Centering characters by calculating their center of mass and adjusting their positions accordingly.
Normalizing pixel values to the range of [0,1].

Backend API Development

The trained model is loaded and served via a Flask application. The backend manages:
Receiving images uploaded by users.
Executing the preprocessing module.
Passing the processed image through the CNN for analysis.
Returning the predicted character alongside its confidence score in JSON format.
The model is serialized and loaded using PyTorch's torch.load() function.

Frontend Development

The frontend interface is developed using React.js and styled with Bootstrap. It enables users to upload images, view predicted characters, and check confidence levels. The interface interacts with the backend through REST API calls, providing real-time display of results.
The frontend is designed to be responsive and optimized for use on both desktop and mobile devices.

VI. RESULTS AND DISCUSSION

Model Accuracy

The CNN model demonstrated an accuracy of around 93.7% on the test dataset, signifying effective performance in classifying handwritten Tamil characters. There were slight variations in accuracy across different characters, with simpler characters yielding better results and more complex or visually similar ones showing slightly reduced accuracy.

Confusion Analysis

A confusion matrix was utilized to spot frequent misclassifications. Characters that share similar strokes or shapes (e.g., க and ங) displayed a higher likelihood of being misclassified. This indicates a need for enhanced data augmentation or the inclusion of contextual information in future iterations.

Confidence Scores

The system also provides a confidence score for each prediction. In most instances, accurate predictions coincided with high confidence scores (greater than 90%). Predictions with lower confidence levels generally involved characters that were noisy or poorly handwritten.

User Testing

Preliminary testing with non-technical users indicated that the web interface was user-friendly and straightforward. Users could easily upload images and receive predictions without needing any technical knowledge of the system. This highlights the platform's accessibility and usability for educational and cultural preservation initiatives.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to everyone who contributed to the success of this project. Special thanks to our faculty advisor, M. Rekha, for their guidance and support throughout the development of this system. We also acknowledge the creators of the datasets used for training the machine learning model, as well as the developers of the machine learning libraries and tools that made this research possible. Additionally, we thank our peers and colleagues for their valuable feedback and encouragement. Finally, we would like to express our appreciation to the authors of the academic papers and research articles that provided inspiration and foundational knowledge for this work.

REFERENCES

[1]. Varshini, C., Yogeshwaran, S., & Mekala, V. (2024). Tamil and English handwritten character segmentation and recognition using deep learning. 2024 International Conference on Communication, Computing and Internet of Things (IC3IoT), 1–5. https://doi.org/10.1109/IC3IoT60841.2024.10550221

[2]. T.M., V. M., Nevatia, I., & Mishra, T. K. (2023). Categorical boosting machine for Tamil character recognition using shape-based features. 2023 7th International Conference on Computing, Communication, Control and Automation (ICCUBEA), 1–6. https://doi.org/10.1109/ICCUBEA58933.2023.10392044

[3]. Sasipriyaa, N., Natesan, P., Gothai, E., Madhesan, G., Madhumitha, E., & Mithun, K. V. (2023). Recognition of Tamil handwritten characters using Scrabble GAN. 2023 International Conference on Computer Communication and Informatics (ICCCI), 1–5. https://doi.org/10.1109/ICCCI56745.2023.1012856

[4]. Harini, M. L., et al. (2022). Handwritten Tamil character recognition. International Journal of Health Sciences. https://doi.org/10.53730/ijhs.v6nS5.9102

[5]. Akashkumar, S., Dyaram, A. N., & Anand, M. (2022). Identification of Tamil characters using deep learning. In Chen, J. I. Z., Wang, H., Du, K. L., & Suma, V. (Eds.), Machine Learning and Autonomous Systems (Vol. 269). Springer. https://doi.org/10.1007/978-981-16-7996-4_16

[6]. Kowsalya, S., & Periasamy, P. S. (2019). Recognition of Tamil handwritten character using modified neural network with aid of elephant herding optimization. Multimedia Tools and Applications, 78, 25043–25061. https://doi.org/10.1007/s11042-019-7624-2

[7]. Yajima, H., Leow, C. S., & Nishizaki, H. (2024). Text detection and style classification from images using Vision Transformer and Transformer Decoder. 2024 IEEE 13th Global Conference on Consumer Electronics (GCCE), 619–623. https://doi.org/10.1109/GCCE62371.2024.10761042

[8]. Nguyen, Q. D., Phan, N. M., Krömer, P., & Le, D. A. (2023). An efficient unsupervised approach for OCR error correction of Vietnamese OCR text. IEEE Access, 11, 58406–58421. https://doi.org/10.1109/ACCESS.2023.3283340