ADBA Chaos and Fractals
Cutting-Edge Scientific Solutions

e-ISSN: 3023-8951
RESEARCH ARTICLE
Vol.3 / No.2 / 2026 / pp.92-107
https://doi.org/10.69882/adba.chf.2026073

Integrating Fuzzy Logic and Fractional-Order Operators in Convolutional Neural Networks for Handwritten Digits and Characters Recognition

Akshara Sreenivasan [*,1], **Vinodkumar Arumugam** [*,2], **Sriramakrishnan Pathmanaban** [*,3], **Saravanan Arumugam** [β,4] and **Jehad Alzabut** [α,γ,5]

*Department of Mathematics, Amrita School of Physical Sciences, Coimbatore, Amrita Vishwa Vidyapeetham, India , βDepartment of Mathematics, Sona college of Technology, Salem, 636 005, Tamilnadu, India , αDepartment of Mathematics and Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia, γDepartment of Industrial Engineering, OSTIM Technical University, Ankara 06374, Türkiye.

ABSTRACT Handwritten character recognition is vital for document digitization and autonomous reading systems. This study focuses on enhancing handwritten Devanagari character recognition using deep learning models, specifically fine-tuned Convolutional Neural Networks (CNNs), combined with hybrid mathematical methods for image enhancement. Devanagari script, used for several South Asian languages, presents challenges due to its complexity and structural variations. To improve recognition accuracy, we propose a fuzzy-enabled Power-Law transformation for image enhancement, along with other techniques like Grunwald-Letnikov Fractional Differentiation (GLFD) and Atangana-Baleanu-Riemann (ABR). Experimental results show that the CNN model with Power-Law+ Fuzzy enhancement achieves the highest accuracy (98.02%), surpassing even MobileNetV2 (92.86%). The same method also yields impressive performance on the MNIST dataset (99.15% accuracy), demonstrating its effectiveness across different scripts. These findings highlight the benefits of integrating advanced preprocessing with deep learning for improved handwriting recognition, offering practical applications in multilingual document processing and OCR-based automation.

KEYWORDS
Handwritten character recognition
Devanagari script
Image enhancement techniques
Convolutional neural networks (CNNs)
Fractional-order operators

INTRODUCTION

In numerous practical applications, recognizing handwritten scripts is a key task, particularly for scripts that feature intricate characters like Devanagari. A variety of prominent Indian languages, such as Hindi, Sanskrit, Marathi, Konkani, and Nepali, utilize the Devanagari script. One of its 48 fundamental characters, the shirorekha, is a defining horizontal line that runs along the top of complete letters, and it consists of 14 vowels and 34 consonants. The writing direction is from left to right (Jayadevan *et al.* 2011). Moreover, the addition of diacritical signs (matras), which represent vowel sounds, might expand the set of acceptable character forms. Consonants are used to create compound characters, or ligatures, which add to the script's complexity and result in an extensive character set and tremendous structural variety (Pal and Chaudhuri 2004).

Reliable handwritten Devanagari writing recognition can improve the handling of these scripts by natural language processing systems, automate form reading, and digitize archives. This is quite difficult since the script has many complex components, including combined letters (conjuncts), modifiers, many strokes, and peculiar lines (shirorekha) (Arora *et al.* 2024). In addition, a character may be written by the same person at different times in different sizes, shapes, or orientations because everyone writes differently. Other issues, such as differences in print thickness, noise, or scanning distortions, make recognition even more difficult. For optical character recognition (OCR) to recognize Devanagari text, the image must first be cleaned up (preprocessing), then it must be segmented into words, lines, and characters (segmentation), identify the salient features of each character, classify them (often using

Manuscript received: 28 March 2026,
Revised: 29 June 2026,
Accepted: 4 July 2026.

[1]cb.sc.i5das20139@cb.students.amrita.edu,
[2]a_vinodkumar@cb.amrita.edu.(**Corresponding author**)
[3] p_sriramakrishnan@cb.amrita.edu
[4]mathsaran@gmail.com
[5]jalzabut@psu.edu.sa

machine learning), and fix any errors (posting). Character separation and feature extraction are very difficult tasks in Devanagari OCR because of the complexity of the script.

To improve generalization, besides the Devanagari script, we also applied our methods to the Modified National Institute of Standards and Technology (MNIST) dataset, which is a well-known benchmark for handwritten digit recognition. The MNIST dataset comprises 70,000 grayscale images of handwritten digits (0–9), with each image normalized to 28×28 pixels. First published by LeCun et al., this dataset has since become a de facto standard testbed for evaluating machine learning algorithms in image classification (LeCun *et al.* 1998). Although MNIST digits are more structurally primitive than Devanagari letters, they too pose considerable difficulty because of variability in writing style, stroke width, slant angles, and placement. While these variations are, in fact, simpler than those found in Devanagari, we believe that the standardization known to exist within MNIST makes it an excellent benchmark to test the overall effectiveness of our techniques designed to improve performance on diverse writing systems. Testing our techniques on the intricately detailed Devanagari script, along with the more thoroughly researched MNIST dataset, allows us to test the resilience and flexibility of our method across various recognition problems.

Modern OCR systems prefer to use convolutional neural networks (CNNs), recurrent neural networks (RNNs), or transformer-based models that learn end-to-end from raw image inputs without using handcrafted features (Khaparde and Mankar 2018). Integration with natural language models also improves accuracy by placing context on ambiguous character predictions. Image enhancement plays a notable role in deep learning-based OCR classification and contributes significantly to recognition accuracy. Normal preprocessing steps, such as binarization, noise reduction, deskewing, and contrast correction, help in restraining gradient changes before interfacing with the neural network so that the models can focus on salient features instead of irrelevant image quality discrepancies. High-stabilized feature extraction, reduced fraction of needed training time, and improved generalization in documents of diverse types are observed due to enhanced images. Image enhancement is particularly critical for reliably recognizing poor-quality historical documents, worn receipts, or low-resolution scans.

Prominent other modern OCR systems which implement CNNs are Tesseract v4, which switched from LSTM-based neural networks to Tesseract's LSTM-based neural networks, Google Cloud Vision OCR, and Microsoft Computer Vision API. Overall, these algorithms use CNNs for recognition, as well as attend to character shape patterns that can be exploited as features. This problem is well suited to the operations of CNNs, which learn hierarchical levels of abstraction using convolutional layer frameworks, simple shallow edges in the early layers and progressively intricate character motifs in deeper layers.

Research Gaps and Challenges

Due to the variability and noise in handwritten samples, the recognition accuracy of traditional handwritten character recognition systems for the Devanagari script is low. Blurry scans of documents can capture images of characters that are faint, poorly penned, and disjointed. Exceedingly large datasets containing numerous authors and differing conditions add complexity to the task, and conventional methods of pattern recognition are incapable of dealing with such diversity. Rugged and outline-based approaches, which rely on identified features, as well as template-based methods, due to their lack of flexibility and adaptability, do not stand a chance in more complex situations. The primary obstacle is precisely distinguishing Devanagari characters amidst differing writing styles, stroke thicknesses, angles, and obstructions. Without additional image polishing, machine learning approaches risk classifying unconventional handwriting or low contrast characters as noise. Thus, advanced preprocessing techniques coupled with modern deep learning frameworks stand to improve image quality and recognition performance.

Moreover, most existing methods for script recognition do not effectively recognize handwritten numerals. This work uses deep learning architectures along with image enhancement techniques to automate the detection of variability and noise in digits and the Devanagari characters.

Research Highlights and Contributions

To address the research gap discussed earlier, developing a robust handwritten character recognizer for the Devanagari script would significantly advance information retrieval and cultural heritage preservation by converting analog text into editable, searchable digital formats. This has direct implications for libraries, government digitization initiatives, educational content processing, and accessibility software.

In this work, we propose using a power-law transformation, which employs an exponential function to modify pixel intensities (Gonzalez and Woods 2002). This approach improves the average brightness and contrast of an image by redistributing intensity values. A gamma value of less than one makes dark regions lighter, which can enhance elusive handwriting or thin strokes. On the other hand, gamma values greater than one will reduce exposure to bright regions, thus making some subtle details more pronounced. Images with low contrast and poor lighting are best served with this technique, as through a delicate balance of the transformation parameters, faint strokes in handwritten characters that are weak or faded can be rendered visible and decipherable. With the process completed, the images undergo classification and comparison using two deep learning models: a custom CNN and MobileNetV2. To ensure model stability, dataset ablation techniques were applied using the MNIST dataset.

This study has the following key contributions:

- We design a fuzzy-enhanced Power-Law transformation specifically aimed at improving images for Devanagari character recognition.
- We study the impact of other fuzzy image enhancement techniques with Grunwald-Letnikov Fractional Differentiation (GLFD) and Atangana-Baleanu-Riemann (ABR), on the recognition accuracy of handwritten Devanagari characters.
- With a custom CNN and MobileNetV2 architecture, we apply the proposed image enhancement techniques and analyze the response of the models to enhanced inputs.
- We show the application of preprocessing techniques in handwriting recognition systems and how they can be used to improve classification performance.
- In testing the generalization capabilities of our models, we perform additional benchmark testing using the MNIST dataset.

The rest of the manuscript is organized in this manner: Section 2 examines the current literature on image enhancement methods, cutting-edge deep learning models, and recent progress in handwritten character recognition. Section 3 outlines the datasets and assessment criteria utilized in our experiments, featuring the Devanagari dataset and the MNIST dataset. Section 4 outlines our

approach, including preprocessing procedures, the suggested image enhancement methods, and the deep learning models utilized. Section 5 displays the experimental findings and evaluation for both the Devanagari and MNIST datasets, emphasizing the influence of preprocessing on recognition precision. Section 6 examines the constraints of our method and highlights possible avenues for future investigation. Section 7 concludes the paper by recapping our results and exploring their significance for systems that recognize handwritten characters.

RELATED WORK

This section reviews existing image enhancement techniques, discusses available deep learning models, and summarizes recent advancements in handwritten character recognition.

Convention Image Enhancement Techniques

Over the past few decades, image enhancement has played a crucial role in classification and recognition tasks, particularly with the advent of numerous deep learning models (Krizhevsky *et al.* 2017). These techniques are applied in image processing to improve image quality and emphasize significant features, thereby making images more suitable for analysis or recognition by deep learning algorithms. In the context of handwritten character recognition, image enhancement can be especially vital, as even minor improvements in clarity and sharpness can significantly enhance a model's ability to differentiate between similar characters (Garg and Singh 2012; Sharma and Kumar 2021). Here, we summarize two enhancement methods that are relevant to this work:

GLFD is an extension of traditional differentiation concepts in mathematics, utilizing non-integer orders of differentiation. This method acts as an advanced edge detection filter in imaging (Podlubny 1999). While classical first-order derivatives enhance edges by detecting sharp intensity changes, fractional-order derivatives (with orders between 0 and 1) provide greater flexibility, allowing for more effective enhancement of fine details. The Grunwald–Letnikov method calculates a fractional difference for each pixel, effectively highlighting medium and high-frequency components (such as edges and textures) without eliminating low-frequency (smooth) components. In the case of character images, GLFD has been used to accentuate edges and highlight subtle stroke textures that might otherwise go unnoticed with conventional filters. This enhancement can make delicate curvatures and line tips of handwritten characters more pronounced, potentially enabling recognition models to distinguish characters with near-similar appearances more easily (Das and Gupta 2013).

The ABR technique is designed to make low-contrast photos clearer. It's especially good at highlighting differences in both small areas and the overall image (Atangana and Baleanu 2016). Its strength comes from combining fractal geometry and fractional calculus, which helps it capture fine details without losing important parts of the image, like edges and textures. The ABR method makes sure that subtle changes in the image are emphasized while keeping key structural elements intact by using these mathematical ideas. Both methods help improve image quality, especially when recognizing handwritten characters. ABR is better at preserving texture and increasing contrast, while GLFD focuses on sharpening edges and small details to bring out subtle features more clearly. Together, they create a strong approach for preparing images, making them better suited for deep learning-based recognition systems.

Deep Learning for Character Recognition: Deep learning has revolutionized character recognition by directly extracting important features from raw image data, eliminating the need for time-consuming manual feature engineering. Two well-known types of deep neural networks that are particularly useful in this context are Convolutional Neural Networks (CNNs) and MobileNetV2.

CNNs (Convolutional Neural Networks) have become the preferred choice for image recognition tasks, including Optical Character Recognition (OCR). These networks use multiple layers of convolution filters to process images and detect features like edges, curves, or more complex shapes as they move through deeper layers. Each convolution layer uses a set of learned kernels that activate when specific patterns in the image are detected (LeCun *et al.* 1998). Following certain convolution layers, pooling layers reduce the spatial resolution of feature maps while retaining the most critical information. By stacking convolution and pooling layers, CNNs build a hierarchical representation of features starting with simple edges in early layers and progressing to complex character shapes in later layers. At the output, fully connected (dense) layers utilize these extracted features to classify the image into one of the predefined character classes. CNNs are especially well-suited for character recognition because their learned features are translation-invariant, allowing them to handle minor variations and distortions in handwriting effectively. Over the past decade, CNN-based methods have consistently outperformed traditional approaches for handwritten character recognition across various scripts.

MobileNetV2 is a specialized CNN architecture designed for mobile and embedded systems (Sandler *et al.* 2018). It is a lightweight network that achieves high performance with significantly fewer parameters and computational operations compared to larger models. The key innovation in MobileNetV2 is the use of depth-wise separable convolutions, which break down a standard convolution into two steps: first, a depth-wise convolution processes each input channel independently, and second, a point-wise (1×1) convolution combines the outputs across channels. MobileNetV2 reduces computation while still learning important features. It uses an inverted residual design with linear bottlenecks: each block expands the channels, applies depth-wise convolution, and then compresses them again. Linear bottlenecks preserve information, and shortcuts between blocks improve gradient flow and reuse features. After training on large datasets like ImageNet, it can be fine-tuned for character recognition, offering high accuracy with low computational needs and making it ideal for handwriting recognition on mobile or low-power devices (Howard *et al.* 2017).

Works with Handwritten Devanagari Characters Recognition

Several strategies for detecting handwritten Devanagari characters have been developed over the years. Early techniques relied on typical machine learning classifiers with manually produced features. Researchers used approaches such as SVMs and k-NN to extract features like zoning characteristics or histograms of directed gradients before training classifiers to recognize characters. While moderately effective, these approaches suffered from handwriting variations and noise, particularly as datasets rose in size and diversity. As a result, they frequently failed to scale effectively with rising data complexity (Sethi and Singh 2020).

Deep learning approaches have gained popularity in recent years due to their ability to learn discriminative features directly from raw data without the need for manual feature engineering. The performance of large-scale CNN-based models has outperformed traditional methods when applied to Devanagari character

datasets. These models work very well because human writing naturally varies in handwriting styles and stroke shapes. For example, a study on deep CNNs for large-scale Devanagari character recognition showed that by internally capturing minute aspects of the script, deeper networks could maintain good performance even when the dataset size expanded considerably. Researchers have investigated mixing deep learning with picture preprocessing methods to further increase recognition accuracy (Agrawal and Nair 2019). One such approach is to use fractional differentiation methods, such as the Generalized Local Fractional Derivative (GLFD), before CNN classification. By sharpening the input images with GLFD, studies were able to improve the accuracy of feature extraction and recognition. By highlighting subtle differences in stroke patterns, the sharper images helped CNNs understand character shapes, particularly when it comes to recognizing characters with similar outlines. Another innovative preprocessing method is the application of ABR (Atangana-Baleanu-Reimann) fractal-fractional differentiation techniques. These methods use fuzzy principles to improve local contrast and reduce noise, resulting in crisper handwritten Devanagari characters. Training CNNs on these updated images boosted the models' ability to distinguish different features. Combining deep learning and excellent preprocessing has considerably improved the accuracy of handwritten Devanagari OCR systems, outperforming any single method.

Traditional machine learning classifiers with manually constructed features were employed in early research on Devanagari character detection. However, accuracy and scalability have significantly increased due to recent developments in deep learning, particularly CNNs. A potent method for improving handwritten Devanagari character recognition is the combination of deep learning models with sophisticated preprocessing techniques.

Works with MNIST Digits Recognition

The industry standard for assessing machine learning techniques has long been the MNIST dataset. The accuracy rates of the first methods, which used handcrafted features and conventional classifiers like SVMs and k-NN, were 95–98% (Liu *et al.* 2003). Model robustness and efficiency can still be tested with MNIST, even if it is regarded as a solved problem. The goal of recent research is to maintain good performance while minimizing model size and computing demands, particularly for devices with limited resources (Sandler *et al.* 2018; Howard *et al.* 2017).

In contrast to scripts like Devanagari, image enhancement techniques for MNIST have received relatively less attention because modern CNNs already perform exceptionally well on this benchmark dataset. However, experiments conducted under degraded conditions, such as noise, blur, or low contrast, have shown that effective preprocessing can significantly boost performance. Through the comparison of various improvement techniques on MNIST and other datasets, such as Devanagari, researchers may learn more about how the advantages of preprocessing vary depending on the dataset's characteristics and character complexity. This comparative study clarifies how preprocessing improves model performance at different data quality and script complexity levels.

MATERIALS AND METRICS

This section outlines the dataset sources and performance metrics employed to evaluate the models for handwritten character identification.

Materials

We use a sizable publicly accessible dataset of handwritten Devanagari characters for this study. Over 92,000 grayscale pictures of isolated character glyphs make up this collection, which represents 36 different classes of the Devanagari alphabet (one class per unique letter). The UCI Machine Learning Repository makes the dataset available (Acharya and Gyawali 2015). The collection's images all have a single handwritten character on a simple backdrop. The samples, which cover a broad spectrum of handwriting styles, were gathered from several writers and sources. Variations include variations in slant, where some authors lean to the right or left when writing, and variations in stroke thickness, with some characters being created with thinner pens and others are using thicker markers. Additionally, the images contain imperfections such as faint smudges of ink, background speckles, or uneven scanning lighting, which contribute to their realism.

The Modified National Institute of Standards and Technology (MNIST) dataset serves as a standard yardstick for concept categorization and machine intelligence tasks. It consists of 70,000 in manuscript number concepts (0–9), each accompanying a resolution of 28×28 pixels (LeCun *et al.* 2010). Widely used to judge concept categorization algorithms, the MNIST dataset is divided into 60,000 training representations and 10,000 test images, with about equal likeness across all ten classes (digits 0–9). The digits were normalized to fit inside a 20×20-pel box while maintaining their facet percentage, and therefore focused within a 28×28 field utilizing center-of-bulk predictions. The dataset was built from NIST's Special Database I and Special Database III, which involve digits composed by grades 9–12 juniors and Census Bureau operators (Deng 2012). A notable feature of MNIST is that the countenances have been pre-treated, accompanying antagonistic aliasing, bearing grayscale levels that capture variations in stroke breadth. This preprocessing creates the dataset more sensitive, and disputing distinguished to twofold pel likenesses.

Sample images from both datasets are illustrated in Figure 1. The Devanagari dataset exhibits blur and low contrast, while the MNIST dataset also contains blurred edges, both of which can negatively impact classification performance. Therefore, preprocessing steps such as image enhancement are crucial before applying deep learning models for classification. To prepare the data for testing and training, we split the datasets into two subsets. Of the Devanagari dataset, roughly 69,000 samples (75%) were used to train the model, with the remaining 23,000 samples, or 25%, left aside for testing. To make it easier to compare the MNIST dataset's results with those of other published studies, we chose the default split of 60,000 training images and 10,000 test images. A fair evaluation of the models' ability to generalize to novel handwriting patterns was ensured by excluding the test images from the training procedure.

Metrics

We use a variety of well-known benchmark metrics that offer an in-depth understanding of the classification behavior during various training and testing stages to fully assess our model. According to recent work on machine learning evaluation, these metrics are particularly helpful for evaluating generalization, identifying overfitting, and resolving issues with imbalanced data (Naidu *et al.* 2023). True positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) are the four classifications into which the confusion matrix divides classification results as a simple diagnostic tool. In situations where there is a class imbalance or asymmetric misclassification costs, this matrix is very useful since it provides a clear picture of where and how the model misclas-

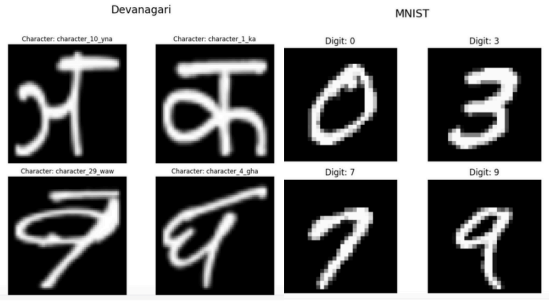

Figure 1 Sample images from datasets

sifies samples. To obtain a more thorough grasp of the model's efficacy, important performance metrics including, precision, recall, and the F1 score, can be extracted from the confusion matrix (Naidu *et al.* 2023).

In applications like spam filtering and fraud detection, where false positives are undesirable, accuracy is especially important. We tracked accuracy and loss curves as a function of training epochs in addition to static performance scores. Accuracy reflects the proportion of correctly classified instances out of the total number of predictions and is mathematically defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

To further refine the performance evaluation, we utilize precision, which measures the proportion of accurate positive predictions out of all instances predicted as positive. It is mathematically expressed as:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

To complement accuracy, we employ recall (or sensitivity), which assesses the model's ability to correctly identify all true positive instances. It is quantified as:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

A high recall is particularly crucial in applications such as medical diagnosis, where failing to detect a positive case can have severe consequences. Since precision and recall often trade off against each other, the F1 score offers a harmonic mean of the two metrics, providing a single composite measure to evaluate classification performance:

$$F_1 - Score = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \tag{4}$$

The F1 score is particularly favored in cases of imbalanced datasets, as it effectively balances the trade-off between false positives and false negatives. Recent literature, such as Yacouby & Axman [10], highlights how extensions of these metrics allow for the assessment of model uncertainty and the quality of probabilistic outputs, making them increasingly important in modern machine learning pipelines. While accuracy is intuitive and easy to interpret, it can be misleading in the presence of class imbalance. In such cases, accuracy may appear high even when the model performs poorly on the minority class. To address this, we also monitored the loss, which quantifies the model's prediction error. Effective learning is usually indicated by a significant drop in loss values over training epochs. During backpropagation, model weights are updated using the loss function, such as cross-entropy loss in classification tasks, and loss curves offer a visual representation of the optimization procedure. Overfitting is frequently indicated by a large divergence between the training and validation loss curves, whereas underfitting or less-than-ideal learning rates may be indicated by plateauing curves (Naidu *et al.* 2023). We have plotted the Receiver Operating Characteristic (ROC) curve, which shows the True Positive Rate (TPR) versus the False Positive Rate (FPR) across different decision thresholds, to further evaluate the robustness of classifiers. The equations for these metrics are as follows:

$$True\ Positive\ Rate(TPR\ or\ Recall)(y\ axis) = \frac{TP}{TP + FN} \tag{5}$$

$$False\ Positive\ Rate(FPR)(x\ axis) = \frac{FP}{FP + TN} \tag{6}$$

A scalar measure that assesses the model's capacity for class distinction is the Area Under the Curve (AUC) of the ROC. AUC values vary from 0.5, which denotes random guessing, to 1.0, which denotes perfect classification. Because it considers all potential decision thresholds and offers a thorough summary of performance across various operating points, the ROC-AUC score is especially useful for assessing models trained on imbalanced datasets. These metrics and visualizations allowed us to evaluate our handwritten character recognition models in a comprehensive and multifaceted way. They ensured a solid evaluation of performance by successfully capturing the model's generalization ability as well as classification accuracy (Naidu *et al.* 2023; Yacouby and Axman 2020).

METHODOLOGY

Figure 2 shows the flow diagram for the suggested implementation. We used the Fuzzy added Power-Law Transformation to improve the image quality. For comparative analysis, we also used ABR-based techniques and GLFD (Grunwald-Letnikov Fractional Derivative). Every enhancement method was used consistently throughout the dataset, and distinct models were trained and assessed with the improved photos associated with each technique. The various stages of the implementation, such as preprocessing, image enhancement, data augmentation, and deep learning models, are all depicted in Figure 2.

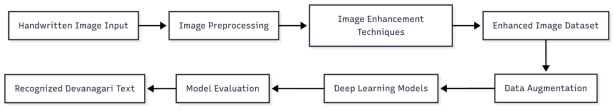

Figure 2 The flow diagram of the proposed implementation

Preprocessing

The first phase of the proposed implementation involves preprocessing. Before feeding the images into the models, a consistent

preprocessing step was applied to all character images. Each image was resized to a fixed resolution to ensure uniformity. For the CNN model, which requires input images of 64×64 pixels, all images were resized to this dimension. This resizing standardizes the aspect ratio and size of the characters, minimizing the impact of variations in the original image sizes on the learning process.

For the MobileNetV2 model, which expects larger color images, the images were upsampled to 224×224 pixels and converted from grayscale to RGB format. This transformation is detailed further in the preprocessing section. After resizing, the images remained grayscale, with pixel intensities ranging from 0 (black) to 255 (white). Binarization was not performed; instead, we retained the grayscale intensity values because subtle grayscale variations can contain valuable information about stroke quality.

Image Enhancement Techniques

Image enhancement plays a crucial role in the preprocessing workflow of this study. We have proposed an enhancement method using Fuzzy added Power-Law Transformation for the input images. Additionally, we implemented GLFD (Grunwald-Letnikov Fractional Derivative) and ABR-based methods with fuzzy to evaluate their impact on recognition. Each method has a distinct function and operation:

Power-Law + Fuzzy (Gamma) Transformation: To enhance the brightness and balance of images, we applied a power-law (gamma) transformation. This transformation is defined as:

$$I_{power\_law} = cI^{\alpha} \tag{7}$$

In equation (7), I represent the pixel intensity in the original grayscale image, α is the gamma exponent, and c is a scaling constant. We typically choose the value of c such that the maximum intensity value of 255 remains unchanged, ensuring that the image's dynamic range is preserved.

The value of α plays a key role in how brightness is redistributed:

- When $\alpha < 1$ the image gets brighter, which helps to bring out darker areas and makes faint strokes more noticeable.
- When $\alpha > 1$ the image darkens, reducing the emphasis on brighter areas and possibly losing some detail in the lighter parts.

As a result of this transformation, the images exhibit generally higher contrast. Backgrounds tend to approach pure white, while text or ink strokes become darker and more distinct. This enhanced visual clarity is expected to assist the neural network by making it easier to differentiate between various character shapes. After applying the power-law transformation, we employed a fuzzy contrast enhancement technique to further emphasize the differences between the foreground and background in grayscale images. This method uses a smooth, non-linear transformation that enhances variations in pixel intensities while preserving structural details. Starting with a grayscale input image, we first normalize the pixel values to fit within the range of [0, 1] by dividing each value by 255:

$$I_{norm} = \frac{I_{power\_law}}{255} \tag{8}$$

Next, we apply the fuzzy transformation, which is defined as:

$$I_{fuzzy} = \frac{I_{norm}^{\gamma}}{\left(I_{norm}^{\gamma} + (I - I_{norm})^{\gamma}\right)} \tag{9}$$

In this equation, γ is a self-intensity factor and is a customizable parameter that adjusts the strength of the contrast enhancement. This setup boosts the contrast around mid-gray levels while ensuring a smooth transition at the extremes. When you increase the value of γ, you get a sharper distinction between light and dark areas. Finally, we rescale the transformed image back to the standard 8-bit range and convert it to an unsigned 8-bit integer format for further processing.

$$I_{enhanced} = I_{fuzzy} \times 255 \tag{10}$$

This fuzzy contrast enhancement technique produces images with sharper feature boundaries, which can significantly improve the performance of tasks such as segmentation or character recognition.

Grunwald–Letnikov Fractional Differentiation (GLFD): Our preprocessing workflow included GLFD (Grunwald-Letnikov Fractional Derivative)-based filtering to enhance image details. To improve intensity variations (like edges and gradients), the GLFD algorithm calculates a fractional derivative of the image intensity function. This is more noticeable than no differentiation, which results in a blurry image, but less aggressive than a full first derivative, which may introduce excessive noise. We successfully sharpened edges in each image by applying a fractional differentiation filter, which made the transitions from strokes to background more defined and steeper. For characters with thin or intricate strokes, this is especially helpful because it keeps them from blending into the background when the image is slightly blurry.

Because it captures fine details without over-accentuating noise, which is a common problem with conventional high-pass filters, the GLFD method is sophisticated. Based on suggestions from the literature, we chose a fractional order (a value between 0 and 1) for the derivative when applying this technique to achieve a noticeable sharpening. GLFD preprocessing produces a sharper image than the original, with improved internal details like holes or loops in the character and more distinct character contours. We expected this to help recognition models better identify the most important structural components of each character class.

For calculating the fractional derivative, Grunwald–Letnikov (GL) formulation is utilized. It is given by:

$$D^{\mu} f(x) = lim_{\epsilon \to 0} \frac{1}{\tau(1-\mu)} \sum_{k=0}^{N} (-1)^k \binom{\mu}{k} f(x - k\epsilon) \tag{11}$$

Where μ s the fractional order of the derivative,$f(x)$is the image intensity at position x,$\epsilon$is a small step size (typically set to 1 pixel), and τ is the Gamma function.$\binom{\mu}{k}$ is the generalized binomial coefficient, defined by:

$$\binom{\mu}{k} = \frac{\tau(\mu+1)}{\tau(k+1)\tau(\mu-k+1)} \tag{12}$$

This structure allows intensity values from earlier times to be weighted, with their influence decreasing gradually based on the fractional order. This well-balanced sensitivity enables richer feature extraction from noisy or low-contrast images by effectively distinguishing between edge saliency and smoothness (Acharya and Gyawali 2015).

ABR Enhancement Approach: The other enhancement approach we employed was ABR (Adaptive Block-based Rescaling) enhancement. To address challenges posed by large physical image sizes, we simplified the problem by assuming that, within each image

block (for example, an 8×8 pixel area), lighting conditions and contrast are equally non-uniform. This assumption enabled us to apply adaptive corrections within blocks: darker or lower-contrast blocks compared to the overall image were brightened or had their contrast stretched, while properly exposed blocks remained unchanged. This adaptive processing is highly effective for images where character parts appear lighter due to uneven or soft scan strokes. The ABR enhancement process involves several steps: fuzzy enhancement, fractional derivatives, and the use of the ABR fractal-fractional gradient mask. Enhancement is performed using a generalized fractional derivative strategy, leveraging the Mittag-Leffler function for calculating fractional derivatives. This approach preserves subtle details without exaggerating noise.

ABR fractal-fractional derivative is given as:

$$F_{ABRD_{\alpha,\beta}}[I(t)] = B(\alpha)\frac{1}{(1-\alpha)^{\beta}}\frac{d^{\beta}}{dt^{\beta}}\int_0^t I(s)E_{\alpha}\left(-\frac{\alpha}{1-\alpha}(t-s)^{\alpha}\right)ds \quad (13)$$

Where α and $]\beta$ represent the fractional order and fractal dimension, respectively, I is the image intensity function at the pixel level.

The Mittag-Leffler function E_{α} is defined as:

$$E_{\alpha}(z) = \sum_{k=0}^{\infty}\frac{z^k}{\tau(\alpha k+1)} \quad (14)$$

Here,z is a complex variable, and τ is the Gamma function, which generalizes the factorial function. This function generalizes the exponential function and is widely applicable in fractional calculus. It also helps in addressing memory effects and self-similarity properties of images (Ramesh Babu *et al.* 2024). Among all the strategies we explored, the most effective combination was Fuzzy enhancement followed by Power-Law transformation, which consistently yielded the best recognition accuracy. This step-by-step approach leverages both local adaptivity and global contrast normalization, resulting in clearer character representations and a significant improvement in model performance.

Data Augmentation

In addition to resizing, we performed data augmentation on the training set. Data augmentation involves generating modified versions of the training images to artificially expand the dataset and introduce more variation. By applying random transformations such as rotations, shifts, shearing, and zooming, we helped the model generalize better to new images. During each training epoch, augmented samples were created dynamically: random small rotations (up to ±10 degrees), minor horizontal and vertical translations (up to 10% of the image size), slight shearing (by a factor of 0.1), and moderate zooming in or out (up to 10%). The robustness of the model is increased by these transformations, which mimic natural variations like characters appearing at different positions or orientations on the paper. A solid basis for developing a successful character recognition model is provided by the combination of augmentation techniques and a varied dataset. We utilized data augmentation during training to enhance model generalization and mitigate overfitting to the training data. Random transformations were applied to each training image with a certain probability in every epoch. The specific details of the data augmentation are as follows:

- Rotation: Random rotations of up to ±10 degrees to simulate characters that may be slanted or rotated during scanning.
- Translation: Small translations of up to 10% of the image size in either direction, enabling the model to recognize characters that are not centrally aligned.
- Shear: A shear transformation (tilting the image) by a factor of 0.1, which mimics changes in writing angle.
- Zoom: Random zooming in or out by up to 10%, allowing the model to handle minor size variations in handwriting.

Deep Learning Models

We evaluated both our custom CNN and MobileNetV2 models on datasets enhanced using each method individually, as well as on the original (unenhanced) data to establish a robust baseline for comparison. We also explored hybrid enhancement techniques, particularly combining Fuzzy contrast enhancement with Power-Law transformation, which yielded the most consistent performance improvements. To ensure fairness across all experiments, we maintained consistent training conditions, such as data augmentation, learning rates, and the number of epochs. Importantly, we applied augmentation after the enhancement step to preserve the quality of the improved images. This systematic approach allowed us to isolate the impact of each enhancement technique on model performance. For example, we observed a notable increase in accuracy for the CNN when using GLFD-based enhancement, while MobileNetV2 showed only minor improvements. This suggests that MobileNetV2's deeper architecture is already effective at capturing fine details, making it less reliant on additional enhancements. These comparisons are essential for determining which enhancement methods work best and for identifying their compatibility with specific models. For our handwritten character recognition project, we examined two deep learning models: a custom Convolutional Neural Network (CNN) and the MobileNetV2 architecture, which we adapted using transfer learning.

Custom CNN Architecture: We designed our CNN specifically to process 64×64 grayscale images. The architecture consists of three convolutional layers followed by two fully connected (dense) layers. Each convolutional block employs 3×3 kernels, ReLU activations, and 2×2 max pooling. The dense layers integrate the extracted features and perform the classification task. To mitigate overfitting, we incorporated a dropout layer, and the final softmax layer outputs probabilities across the 36 character classes. This model is lightweight and easy to train, making it ideal for benchmarking and experimenting with image enhancements. The customized CNN architecture is illustrated in Figure 3 and detailed in Table 1.

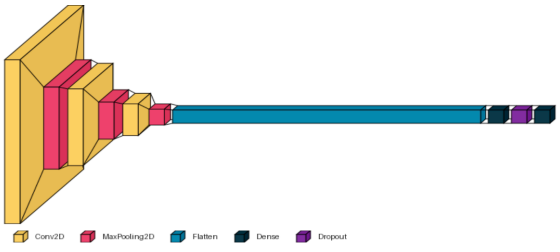

Figure 3 The custom CNN architecture

MobileNetV2: MobileNetV2 is a computationally efficient architecture designed for mobile and embedded systems, offering significant reductions in computational cost without compromising accuracy. We adapted MobileNetV2, a state-of-the-art CNN architecture, for our model. It has been pre-trained on ImageNet, a

Table 1 The custom CNN layered parameters

Layer	Filters / Units	Kernel Size	Activation	Other
Conv2D	32	3×3	ReLU	-
MaxPooling2D	-	2×2	-	-
Conv2D	32	3×3	ReLU	-
MaxPooling2D	-	2×2	-	-
Conv2D	64	3×3	ReLU	-
MaxPooling2D	-	2×2	-	-
Flatten	-	-	-	-
Dense	128	-	ReLU	Dropout (0.5)
Dense (Output)	36	-	-	-

large dataset of labeled images, which enables it to leverage features learned from a wide range of natural images, such as edges, textures, and object parts. This pre-training allows MobileNetV2 to generalize well to new tasks, like handwritten character recognition, especially when working with smaller datasets, using transfer learning.

The MobileNetV2 architecture is detailed in Table 2. Since MobileNetV2 is designed to accept input images of size 224×224 with three color channels (RGB), we modified our grayscale input images accordingly. Specifically, we resized the handwritten grayscale images from 64×64 to 224×224 to match the required input size. To accommodate the three-channel requirement, we duplicated the grayscale channel across all three RGB channels, effectively converting the images into pseudo-RGB format. One of the key innovations in MobileNetV2 is the use of inverted residual bottleneck blocks. These blocks are designed to extract features efficiently while maintaining a lightweight structure. Unlike standard residual blocks, inverted residuals employ depth-wise separable convolutions, which significantly reduce both parameters and computational cost. To accomplish this, a lightweight 1×1 convolution first expands the number of channels; a depth-wise separable convolution then independently convolves each channel; and lastly, another 1×1 convolution shrinks the channels to their initial size. By reducing computations while maintaining high expressiveness, this method allows for rapid and effective processing without sacrificing model accuracy.

The use of ReLU6 activations rather than the conventional ReLU is another noteworthy aspect of MobileNetV2. ReLU6 improves performance and model stability by capping activation values at a maximum of 6, especially in mobile environments where computational efficiency is crucial. Skip connections are another feature of MobileNetV2 that enables the model to learn residual mappings. By avoiding problems like vanishing gradients, these skip connections improve gradient flow during training. Consequently, the model can efficiently learn complex features without experiencing a decline in performance.

We modified MobileNetV2 by adding a unique classification head to the model to tailor it for our handwritten character recognition task. There are two primary parts to the classification head: GAP or global average pooling: By calculating the average value for each channel across the entire map, this layer shrinks the feature map's spatial dimensions (height and width). In this step, the representation is made simpler, fewer parameters are used, and overfitting is avoided without sacrificing important classification-related data. Dense Layer Activated by Softmax: The probabilities for each of the 36 classes in our case, the 36 different Devanagari characters are produced by this layer. Interpretable class probabilities are provided by the Softmax activation, which guarantees that the output values add up to 1.

Following our dataset training, we adjusted the weights in the deeper layers at a low learning rate to refine the model. The model can be fine-tuned to accommodate the unique characteristics of handwritten characters, such as shapes and stroke patterns, without changing the previous layers that capture more general features like edges and textures. By utilizing ImageNet's strong pre-trained weights, this method helps prevent overfitting, particularly when dealing with small datasets. Our handwritten character recognition task was completed by MobileNetV2 with less overfitting and less computational overhead, thanks to its effective architecture, pre-trained weights, and fine-tuning.

Model Environmental Setup and Evaluation

Preprocessing: Before feeding the original or enhanced images into the neural network models, we performed some preprocessing steps to ensure the data was in an optimal format. Here's a breakdown of the key preprocessing techniques: Resizing:

- For the CNN model, all images were resized to 64×64 pixels in grayscale.
- For MobileNetV2, the images were upscaled to 224×224 pixels. To make the grayscale images compatible with MobileNetV2, which expects three-channel (RGB) input, we duplicated the single grayscale channel across all three channels, creating a pseudo-RGB image.
- Resizing ensures that all input images have a consistent size, maintaining the aspect ratio. Since the original images are mostly square and centered, scaling them to a uniform size did not significantly distort the characters. Standardizing the input size also reduces any resolution-related differences that could impact model performance.

Data Augmentation:

- Data augmentation was applied only to the training images to increase variability and improve model robustness. Examples of augmentations include slight rotations (both left and right),

Table 2 MobileNet V2 Architecture

Layer Type	Input Size	Output Size	Kernel Size	Stride	Expansion Factor
Initial Convolution	224×224×3	112×112×32	3×3	2	-
Inverted Residual Block	112×112×32	112×112×16	3×3	1	1
Inverted Residual Block x2	112×112×16	56×56×24	3×3	2	6
Inverted Residual Block x3	56×56×24	28×28×32	3×3	2	6
Inverted Residual Block x4	28×28×32	14×14×64	3×3	2	6
Inverted Residual Block x3	14×14×64	14×14×96	3×3	1	6
Inverted Residual Block x3	14×14×96	7×7×160	3×3	2	6
Inverted Residual Block x1	7×7×160	7×7×320	3×3	1	6
Final Convolution	7×7×320	7×7×1280	1×1	1	-
Global Average Pooling	7×7×1280	1×1×1280	-	-	-
Fully Connected	1×1×1280	1×1×1000	-	-	-

shifts, and other transformations that simulate natural variations in handwriting. These transformations create multiple versions of the same character, helping the model learn more resilient features.
- It is important to note that no augmentation was applied to the test images. The test set remained untouched and static to ensure accurate performance measurement. By expanding the training set through augmentation, we effectively increased its diversity, making the models more adaptable to real-world variations.

Normalization:

- After resizing (and applying augmentations during training), we normalized the pixel intensity values. All pixel values, which originally ranged from 0 to 255 in 8-bit grayscale, were divided by 255 to scale them to floating-point numbers between 0.0 and 1.0.
- This normalization ensures that all input features (pixel intensities) are on the same scale, leading to more stable and efficient training for the neural network. It also prevents large numerical values in raw pixel data from biasing the updates of the network weights.
- In this case, we did not perform mean subtraction or standard deviation scaling because the grayscale data was already quite uniform, and simple division by 255 was sufficient to normalize the inputs effectively.

By carefully preprocessing the images in this manner, we ensured that the data was well-prepared for both the CNN and MobileNetV2 models, enabling them to learn effectively and generalize well to new handwritten characters.

Model Training: We trained both the CNN and MobileNetV2 models under two scenarios: using original images and enhanced images, following a supervised learning approach. Below is a summary of the training configuration and hyperparameters for the custom CNN model:

- Optimizer: We used the Adam optimizer with a learning rate of 0.001. Adam is known for its adaptive learning rate, which typically leads to faster and more stable convergence compared to standard stochastic gradient descent, making it well-suited for our application.
- Loss Function: We employed Categorical Cross-Entropy loss, which is ideal for multi-class classification problems with one-hot encoded labels for our 36-character classes. This loss function measures how far the predicted class probabilities are from the true class (ground truth), guiding the model toward better performance.
- Metrics: During training, we closely monitored accuracy as our primary metric to assess how well the model was fitting the data. Accuracy represents the ratio of correctly classified characters.
- Batch Size: We used mini batches of 32 images during training. This batch size strikes a balance between reducing gradient noise and managing memory requirements. It is small enough to introduce some randomness (which helps the model avoid local minima) but large enough to ensure stability while training on our hardware.
- Epochs: The CNN was trained for a maximum of 30 epochs. We also implemented early stopping if the validation accuracy did not improve for a certain number of consecutive epochs, training would halt to prevent overfitting and unnecessary computation. In practice, the CNN often reached its best performance well before the 30-epoch threshold, due to the dataset size and the use of augmentation.

The configuration and hyperparameters for the MobileNetV2 model were as follows:

- Optimizer: For MobileNetV2, we also used the Adam optimizer, but with a lower learning rate of 0.0001. Since MobileNetV2 was pre-trained, a lower learning rate helped prevent drastic updates that could disrupt the pre-trained features. This subtle update mechanism allowed the model to fine-tune effectively.
- Regularization: Regularization was crucial for improving generalization in both models. For the CNN, we applied dropout in the dense layer, randomly skipping neurons during each training update. This technique prevents the network from over-relying on specific features or feature co-adaptation. MobileNetV2, on the other hand, benefits from its architecture, which includes batch normalization applied during pre-training.
- Early Stopping: To avoid overfitting, we controlled the training duration using early stopping. Additionally, data augmentation served as a regularization method by providing the model with diverse training samples.

During training, we evaluated the model's performance on a validation set after every epoch. This validation set was a subset of the training data reserved specifically for monitoring progress. We saved the model weights when the validation accuracy peaked at any epoch, which we later used to measure performance on the test set and compute final metrics. We repeated this training process for each combination of model and enhancement technique, as well as for a baseline setup without any enhancements. This systematic approach ensured comprehensive evaluation of all configurations.

Model Evaluation: Model performance comparison across different image conditions is a key component of our experimental setup. Specifically, we will compare the performance of both models (CNN and MobileNetV2) on original raw images versus enhanced images processed using three enhancement methods: Fuzzy-Power-Law, GLFD, and ABR-Fuzzy. The original images serve as the baseline for reference. If an enhancement method is effective, the model's accuracy and other performance metrics on the test set should improve compared to the baseline. Based on preliminary observations, Fuzzy-Power-Law, GLFD, and ABR-Fuzzy enhancements have shown promise in increasing recognition accuracy for each model. It remains to be seen whether each model has a unique preference for a specific enhancement method or if one approach consistently outperforms the others.

To present our findings, we plan to provide both quantitative results and qualitative visualizations clearly and understandably. Quantitative results will be condensed using charts. For example, a bar chart might compare CNN's test accuracy across four conditions: No Enhancement, Power-Law, GLFD, and ABR-Fuzzy side by side. Similarly, we will create comparable visualizations for MobileNetV2. These charts will immediately highlight which enhancement method provided the largest improvement for each model.

Qualitative visualizations will complement the quantitative results by illustrating the advantages of individual enhancement techniques. We will present before-and-after images to demonstrate the impact of each enhancement method. For instance, a particularly challenging sample character image can be shown in its raw state alongside versions enhanced using Fuzzy-Power-Law, GLFD, and ABR-Fuzzy. Such comparisons might reveal that ABR-Fuzzy reduces background noise, GLFD emphasizes edges more effectively, and Power-Law makes the image significantly brighter and clearer. These visual representations provide insight into the correlation between image processing and model performance: if an enhancement technique makes a previously illegible character legible, it is reasonable to expect improved model performance on that image.

RESULTS AND DISCUSSION

The experiment was conducted on a computer system equipped with a Python 3.10.12 environment, utilizing TensorFlow 2.15.0, Keras 2.15.0, OpenCV 4.8.0, NumPy 1.24.3, SciPy 1.11.3, Matplotlib 3.7.1, and scikit-learn 1.2.2. GPU acceleration was leveraged to expedite model training and inference, significantly reducing computation time. All preprocessing algorithms and deep learning models were executed within this standardized environment to ensure reproducibility and consistency across different experimental runs. Initial development and testing of the code were performed on an M1 chip. The enhancement techniques were developed and validated using various datasets to assess their performance stability. The MNIST dataset was used as a secondary dataset for further validation in this study, which mainly used the Devanagari script dataset. The findings from these datasets are shown in the ensuing subsections.

Results of Devanagari script

This section presents the CNN and MobileNetV2 models' classification performance after they have been trained and assessed on the original and improved datasets. We methodically implemented three techniques: Power-Law + Fuzzy, GLFD + Fuzzy, and ABR + Fuzzy to evaluate how image enhancement methods affected classification performance. Every result was contrasted with a baseline in which no improvement was made. Table 4 summarizes the overall accuracy of each of these approaches.

Baseline (without enhancement): As shown in the results of Table 4, the CNN model outperformed the MobileNetV2 model on the original dataset without any image enhancements. Despite MobileNetV2's pre-trained, deeper architecture, the CNN achieved an accuracy of 92.12%, surpassing MobileNetV2's baseline accuracy of 90.03%. This outcome is somewhat unexpected, as it suggests that the CNN's simpler architecture may have been better suited to the specific characteristics of handwritten characters. These results provide a crucial baseline for evaluating the improvements observed after applying our image processing techniques.

Power-Law + Fuzzy: The Power-Law + Fuzzy enhancement technique yielded the highest accuracy gains, particularly for the CNN model. To determine the optimal level of contrast adjustment, we tested various values of the power-law parameter α. The results with the CNN are summarized in Table 3. These outcomes indicate an improvement of approximately 6% over the CNN baseline. At $\alpha = 0.6$, the best accuracy of 98.02% was achieved, suggesting that moderate contrast enhancement is most beneficial. Values closer to one yielded diminishing return, whereas values that were too low led to excessive brightening. This improvement highlights how effective preprocessing can compensate for simpler model architecture, enabling the CNN to surpass its baseline and even approach MobileNetV2's baseline accuracy.

Power-Law + Fuzzy preprocessing also enhanced the performance of MobileNetV2, although the improvement was less pronounced, yielding an accuracy of 92.86%. The gain was noticeable but relatively modest because MobileNetV2 is a more robust and pre-trained model that naturally handles a wide range of variations. Nonetheless, this result validates the overall effectiveness of the Power-Law enhancement technique.

Table 3 The Power-Law + Fuzzy enhancement performance comparison over α on Devanagari script

α Value	Accuracy (%)
0.5	97.82
0.6	98.02 (highest)
0.7	97.94
0.8	97.78
0.9	97.82

GLFD + Fuzzy: From the results in Table 4, the GLFD + Fuzzy approach significantly improved recognition accuracy, particularly for the CNN model. The test accuracy of the CNN increased to 95.23%, which is approximately 3% higher than the baseline. This improvement highlights how well GLFD enhances edge features and fine character details, a benefit that is especially advantageous for CNNs, which rely heavily on localized spatial features.

For MobileNetV2, however, the GLFD + Fuzzy approach resulted in a poor accuracy of 76%, likely due to the model's convolutional filters already being highly effective at extracting edge and shape details. This outcome further illustrates the diminishing returns observed when applying enhancements to already high-performing models. While the additional sharpening provided by GLFD + Fuzzy helped slightly, it did not lead to significant improvements. Compared to the baseline, this approach yielded a negative accuracy difference of -14%.

Table 4 Quantitative analysis of Enhancement techniques vs. Deep learning accuracy on the Devanagari dataset

Enhancement Type	CNN Accuracy (%)	Enhancement Type vs. CNN Baseline	MobileNetV2 Accuracy (%)	Enhancement Type vs. MobileNetV2 Baseline
Baseline (Original)	92.12	–	90.03	–
Power-Law + Fuzzy (α=0.6)	98.02	+5.9	92.86	+2.8
GLFD + Fuzzy	95.23	+3.1	76.02	–14.0
ABR + Fuzzy	92.80	+0.6	87.00	–3.03

ABR + Fuzzy: The CNN's accuracy increased slightly to 92.80% with the ABR + Fuzzy enhancement technique. While this represents an improvement over the raw input, it is less effective compared to the results achieved with Power-Law or GLFD. This suggests that ABR + Fuzzy may not always enhance all image types, despite its ability to remove localized noise and strengthen weak strokes. It is possible that some artifacts introduced during the enhancement process offset its advantages, particularly in cleaner samples. When using ABR + Fuzzy preprocessing, MobileNetV2 occasionally showed minimal changes or even slight degradation,

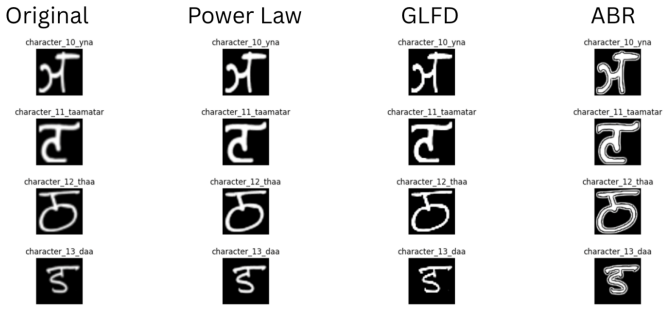

Figure 4 Qualitative results of proposed image enhancement with Devanagari images

resulting in an accuracy of 87%, which is 3.03% lower than the baseline. Over-processing or the introduction of unwanted artifacts may have contributed to this decline, potentially confusing the finely tuned model.

Qualitative results of the proposed image enhancement techniques on original images are presented in Figure 4. Based on the findings from Table 4 and Figure 4, Power-Law + Fuzzy emerges as the most significant improvement overall. It substantially boosts CNN accuracy and narrows the performance gap between CNN and MobileNetV2. GLFD + Fuzzy also offer important improvements, particularly for characters that rely heavily on edges. Although less consistent across all scenarios, ABR + Fuzzy demonstrates certain benefits when dealing with noisy or degraded inputs.

Therefore, Power-Law + Fuzzy emerge as a well-suited enhancement technique for the Devanagari script. The other qualitative metrics are presented in Table 5. The final CNN model's classification report shows a macro-averaged precision of 98.12%, recall of 98.09%, and an F1-score of 98.09% across the 36-character classes, indicating good overall generalization. Several classes, such as character_13_daa, character_22_pha, and character_36_gya, achieved perfect precision and recall, demonstrating that the model effectively generalized the distinct structural features of these characters. The results in Table 5 were computed using the confusion matrix obtained from the Power-Law + Fuzzy enhancement. The confusion matrix is shown in Figure 5, which reveals that most misclassifications occur between structurally similar characters, particularly in noisy or lightly stroked samples.

Table 5 Quantitative analysis of Power-Law + Fuzzy enhancement on Devanagari script

Metric	Value
Accuracy	98.02%
Precision	98.14%
Recall	98.09%
F1-Score	98.09%

The ROC curves for each class, displayed in Figure 6, show near-perfect separation for most classes, indicating strong confidence calibration. The training and validation loss curves demonstrate smooth convergence with minimal overfitting, reflecting the effectiveness of our preprocessing steps in stabilizing the learning process.

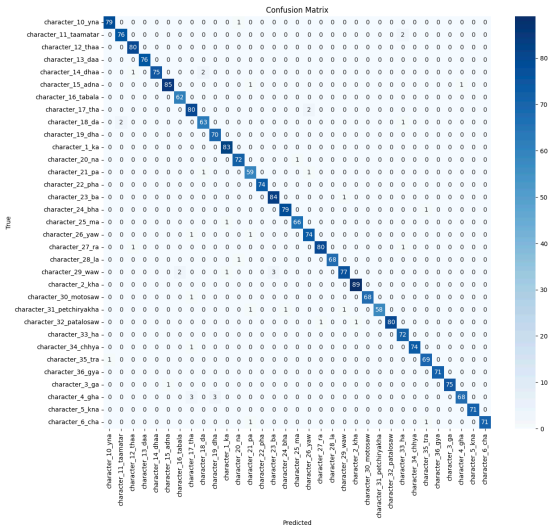

Figure 5 The confusion matrix of Power-Law + Fuzzy enhancement on Devanagari script

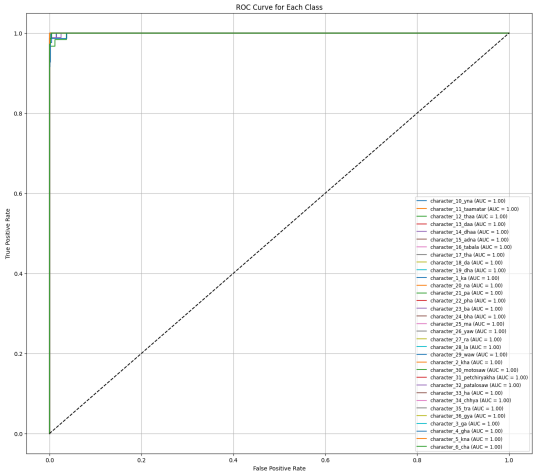

Figure 6 The ROC curve of Power-Law + Fuzzy enhancement on Devanagari script

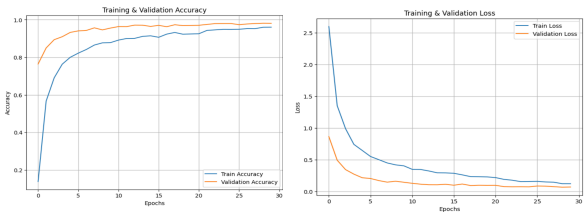

Figure 7 The accuracy and loss curve vs. epoch on Devanagari script using Power-Law + Fuzzy enhancement

Most importantly, these findings show that, with appropriately augmented input, a lightweight CNN can match or even surpass the performance of a heavier model. This has significant practical implications in environments with limited computational resources, where preprocessing at the input level can serve as an extremely effective and cost-efficient method for enhancing recognition accuracy.

Table 6 Quantitative analysis of Power-Law + Fuzzy enhancement on Devanagari script

Research	Method	Accuracy (%)
Narang et al. Narang *et al.* (2021)	Custom CNN	93.73
Sarangi et al. Sarangi *et al.* (2020)	SVM	95.65
Hasan et al. Hasan *et al.* (2023)	Two-stage VGG16	96.55
Proposed Work	Power-Law + Fuzzy and CNN	98.02

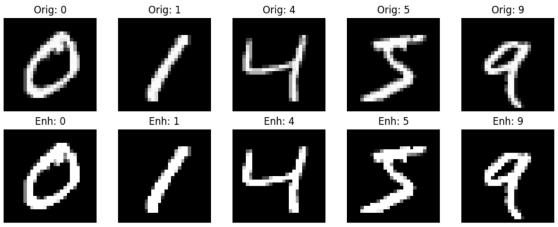

Figure 8 The qualitative results of Power-Law + Fuzzy enhancement on the MINST dataset

Table 6 summarizes recent literature and its findings related to the Devanagari script. Narang et al. developed a specialized model called DeepNetDevanagari for recognizing ancient Devanagari characters, achieving an accuracy of 93.73% (Narang *et al.* 2021). Their work addressed specific challenges associated with degraded ancient documents but did not incorporate any preprocessing techniques to improve image quality before classification. In contrast, Sarangi et al. took a different approach by using feature selection methods combined with SVM classifiers instead of deep learning (Sarangi *et al.* 2020). Their solution achieved an accuracy of 95.65% on the dataset, demonstrating that effective feature engineering can enable older machine learning methods to yield competitive results, though still not matching the performance of our enhancement-based method. Hasan et al. achieved an accuracy of 96.55% using a two-stage VGG16 network with adaptive gradient methods (Hasan *et al.* 2023). Their technique leveraged transfer learning and fine-tuning to reduce the number of trainable parameters, making it computationally efficient, although it lacked the preprocessing advantages offered by our method.

The proposed Power-Law + Fuzzy enhancement method demonstrates superior performance compared to other recent approaches in Devanagari character recognition. The key innovation

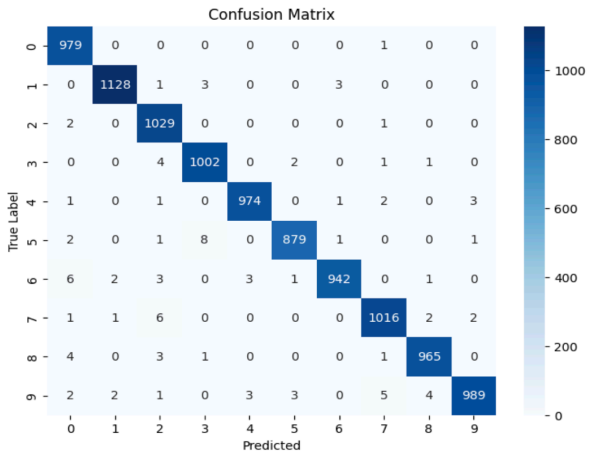

Figure 9 The confusion matrix of Power-Law + Fuzzy enhancement on the MINST dataset

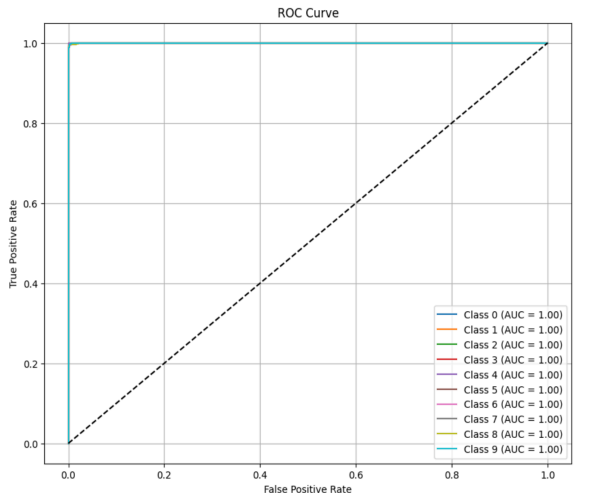

Figure 10 The ROC curve of Power-Law + Fuzzy enhancement on the MINST dataset

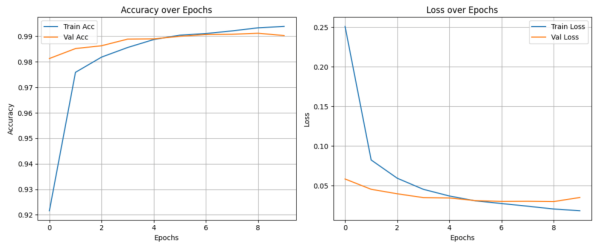

Figure 11 The accuracy and loss curve vs. epoch on MINST dataset using Power-Law + Fuzzy enhancement

of our approach lies in its focus on advanced image preprocessing, rather than relying heavily on complex neural network architectures or intricate feature engineering. A unique contribution of our research is the combination of Power-Law transformation with Fuzzy enhancement, which improves character contrast and edge definition before classification. By prioritizing preprocessing, our method achieves exceptional accuracy (98.02%) using a simple CNN model, making it both computationally efficient and effective when compared to more elaborate methodologies.

Results of the MNIST Dataset: To further evaluate the robustness and general applicability of the proposed Power-Law + Fuzzy enhancement technique integrated within the CNN model, we extended our experiments to include the MNIST dataset. This dataset consists of grayscale images of handwritten digits. Following the preprocessing steps previously applied to the Devanagari dataset, MNIST images were resized to 64×64 pixels to match the CNN input requirements. The enhancement process was carried out in two sequential stages first applying fuzzy enhancement, followed by the adaptive Bernstein Re-weighted (ABR) derivative ensuring a consistent enhancement approach across datasets. Importantly, the CNN architecture remained unaltered to ensure fair and unbiased comparison between the original and enhanced image inputs.

The MNIST dataset used in our experiments comprises 70,000 grayscale digit images (from 0 to 9), categorized into 10 classes. We adopted the standard data split: 60,000 images for training and 10,000 for testing, maintaining a 6:1 train-test ratio. Each digit class is evenly represented, with about 6,000 training images and 1,000 test images per class. This balanced distribution ensures that the model is adequately trained and evaluated across all classes. The standardized nature of MNIST also enables direct benchmarking against existing research.

We trained two CNN models under identical conditions: one using the original MNIST images and the other using the enhanced versions. Both models were trained over 10 epochs using the same data splits and hyperparameter settings. As shown in Figure 8, a visual comparison reveals that the enhanced images exhibit crisper edges and improved contrast, aiding the CNN in capturing class-specific features more effectively.

The final classification results on the MNIST dataset showed that the baseline CNN achieved 99.11% accuracy without enhancement, whereas the model using Power-Law + Fuzzy enhancement achieved a slightly higher accuracy of 99.15%. Although the numerical improvement is small, it demonstrates the consistent benefits of the enhancement technique, even on datasets that are already well-structured and optimized, like MNIST. This modest gain underlines the method's ability to amplify discriminative features, thereby supporting better generalization by the CNN.

To support the previously reported accuracy results, the corresponding confusion matrix is presented in Figure 9. The matrices indicate that the model produced very few misclassifications overall. Notably, the enhanced model demonstrated reduced confusion between visually similar digits, further validating the effectiveness of the enhancement technique.

Figure 10 displays the ROC curve for the Power-Law + Fuzzy enhancement applied to the MNIST dataset. The Receiver Operating Characteristic (ROC) curves for each digit class indicate that the model achieved near-perfect class separation, with area under the curve (AUC) scores reaching or approaching 1.0. This result highlights the effectiveness of the enhancement technique in improving feature discrimination. Figure 11 presents the accuracy and loss curves for the MNIST dataset. Throughout the training

epochs, the model trained on enhanced images consistently outperformed the baseline in both validation accuracy and validation loss, demonstrating more stable and reliable convergence behavior.

Overall, the findings indicate that while the numerical improvements may appear modest, the enhancement methods consistently contribute to increased model robustness and improved convergence behavior. Importantly, the proposed image enhancement approach proves effective not only for complex scripts such as Devanagari but also generalizes well to simpler, widely used datasets like MNIST. These outcomes collectively affirm the efficacy and adaptability of the enhancement technique in the context of handwritten character recognition.

Table 7 Quantitative analysis of Power-Law + Fuzzy enhancement on Devanagari script

Research	Method	Accuracy (%)
Al-Ataby et al. Al-Ataby *et al.* (2021)	Standard SVM	96.30
Kumar et al. Kumar *et al.* (2021)	Custom CNN	97.07
Zhao and Liu Zhao and Liu (2020)	LeNet-5	98.00
Proposed Work	Power-Law + Fuzzy and CNN	99.15

Table 7 presents a comparison between the proposed results and selected recent studies. Al-Ataby et al. explored classical machine learning approaches for handwritten digit recognition, achieving an accuracy of 96.30% using a Support Vector Machine (SVM) model (Al-Ataby *et al.* 2021). Their findings highlighted the limitations of traditional machine learning techniques when applied to the MNIST dataset, especially compared to deep learning methods. Kumar et al. implemented a basic CNN architecture for MNIST digit classification and reached an accuracy of approximately 97.07% (Kumar *et al.* 2021). Their experiments investigated how variations in the number of convolutional layers and filter sizes affected performance. However, their results indicated that even straightforward CNN configurations without specific tuning often fail to surpass the 98% accuracy mark. Zhao and Liu adopted the well-known LeNet-5 deep learning model for MNIST digit recognition, reporting an accuracy of 98.10% (Zhao and Liu 2020). While their study demonstrated the effectiveness of this established architecture, it also introduced added complexity. In contrast, our Power-Law + Fuzzy enhancement approach outperforms these methods, further underscoring the importance of robust preprocessing in boosting classification accuracy, even when applied to well-established CNN models.

The proposed Power-Law + Fuzzy enhancement method achieved a test accuracy of 99.15% on the MNIST dataset, significantly outperforming the alternative approaches discussed. These comparisons highlight the strength of our enhancement technique in improving input data quality through effective image preprocessing. By enhancing discriminative features, the method enables even a relatively simple CNN architecture to attain superior recognition performance.

LIMITATIONS AND FUTURE WORK

While the study demonstrates successful implementation of novel techniques and impressive system performance, it is important to acknowledge several limitations. No single enhancement method consistently produced the best results across all images. For instance, the ABR-Fuzzy approach sometimes diminished performance when applied to already clean images, while an overly aggressive Power-Law adjustment occasionally eliminated important image details. This suggests that our enhancements were tuned to improve the average performance across the dataset, but not necessarily optimized for every individual image. In some cases, certain images may have received little to no benefit from the enhancement process. This indicates that a general, one-size-fits-all approach may not be ideal, and incorporating adaptive enhancement strategies could yield even better results.

During training, particular care was needed to prevent overfitting, especially when using the enhanced dataset with augmented inputs. Feeding the CNN with higher-quality and augmented images increased the risk of the model memorizing the training data, particularly when training was extended for too many epochs. To mitigate this, we implemented dropout layers and early stopping techniques, which ultimately enabled good generalization to the test set. However, the need for such precautions highlights how powerful enhancements can potentially exaggerate training set noise or artifacts, leading to overfitting. Interestingly, overfitting was less problematic in experiments involving the pre-trained MobileNetV2 model, likely due to its prior exposure to broader data and reduced capacity gains from our modifications.

Our experiments were conducted on a specific Devanagari character dataset comprising 92,000 samples. Although sizable and well-curated, this dataset may not capture the full spectrum of handwriting styles or scanning variations. Consequently, the models might have internalized certain biases unique to this dataset. For example, performance might degrade when classifying characters written with different instruments or on rougher paper. Additionally, our enhancement techniques, such as the Power-Law adjustment were calibrated within a fixed parameter range, which may not be optimal for other datasets. Thus, one key limitation is that the proposed system was not evaluated on additional datasets or real-world inputs outside of the current scope. Stronger generalization would require testing across a more diverse range of handwriting conditions and data sources.

Despite these limitations, the current study lays a strong foundation for future improvements. The identified weaknesses point to several promising directions, such as incorporating adaptive enhancement strategies, expanding the number of character classes, and testing on varied dataset sizes. One intriguing possibility is to tailor enhancement techniques based on stroke characteristics, such as thickness or noise level, allowing the system to adaptively select preprocessing methods. For instance, a fuzzy logic controller could be introduced to determine the most suitable enhancement type: Power-Law for dark images, GLFD for overly bright and blurry ones and ABR for noisy inputs. A conditional processing pipeline like this could optimize preprocessing on a per-image basis, potentially leading to more robust classification outcomes.

In this work, MobileNetV2 was chosen as an efficient and compact CNN model. Future research could explore alternative architectures, such as EfficientNet, known for its strong accuracy-to-parameter ratio, or ResNet50, which offers deeper representation capabilities. Additionally, vision transformers might be considered, especially if the research shifts toward phoneme or full-text recognition. Each architecture may respond differently to prepro-

cessing strategies, and certain models could be more resilient to noise or benefit uniquely from specific enhancements. Another worthwhile direction is to extend beyond isolated character recognition to word- and line-level recognition in Devanagari text. Ultimately, our character recognition module could be integrated into a larger OCR pipeline, where image enhancement also supports segmentation tasks.

In conclusion, there remains a significant opportunity for future work to build upon this research. A recurring theme is the synergistic interplay between image preprocessing and deep learning, which has proven beneficial in this study. The results affirm that enhancing input quality through intelligent preprocessing can meaningfully boost recognition performance. Future studies can further explore and refine this synergy, expanding its application to more languages, character sets, and real-world handwritten recognition systems. The foundation laid here demonstrates the potential of such integrated approaches to substantially elevate the accuracy and generalizability of handwriting recognition technologies.

CONCLUSION

Handwritten character recognition is essential in many fields, such as digitizing documents, preserving manuscripts, and developing autonomous text-reading systems. This study focuses on recognizing handwritten Devanagari script, a traditional writing system that originated from the Brahmi script and is widely used for languages like Hindi, Marathi, Nepali, and Sanskrit across South Asia. Because Devanagari is extensively used in government records, educational materials, and cultural texts, accurately identifying these characters is crucial for preserving linguistic heritage, automating Optical Character Recognition (OCR) systems, and enabling multilingual document processing and data entry. To achieve these goals, this research employs deep learning techniques, particularly optimized Convolutional Neural Networks (CNNs), and combines them advanced image processing methods to enhance recognition performance.

This study presents fuzzy logic-based image enhancement techniques, such as Power-Law transformation, Grunwald–Letnikov Fractional Differentiation (GLFD), and Atangana–Baleanu–Riemann (ABR) differentiation, to improve recognition accuracy. By enhancing contrast, sharpening edges, and lowering noise, these techniques aim to improve image features. Two classification models, the lightweight MobileNetV2 and a customized CNN architecture were assessed. A publicly available dataset of handwritten Devanagari characters was used to train and assess both models. Results from experiments show that image preprocessing significantly affects classification performance. When paired with Power-Law + Fuzzy enhancements, the custom CNN produced an astounding accuracy of 98.02%, significantly surpassing MobileNetV2's 92.86%. Overall, the Power-Law + Fuzzy combination produced the best results, even though GLFD and ABR also helped to improve performance.

These findings demonstrate that incorporating well-designed preprocessing methods into even relatively simple CNN architectures can result in high recognition accuracy. The proposed approach is not only effective for complex scripts such as Devanagari, but it also generalizes well, as evidenced by its 99.15% accuracy on the MNIST handwritten digit dataset. This demonstrates the versatility and robustness of the enhancement techniques across diverse datasets. Overall, this framework shows great promise for practical applications in OCR systems for Devanagari and other scripts, providing an efficient and scalable solution for multilingual document analysis and recognition.

Acknowledgments

J. Alzabut expresses his sincere thanks to Prince Sultan University for supporting this research.

Funding

This research received no external funding.

Conflicts of interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical standard

The authors have no relevant financial or non-financial interests to disclose.

LITERATURE CITED

Acharya, S. and P. Gyawali, 2015 Devanagari handwritten character dataset. UCI Machine Learning Repository.

Agrawal, K. K. and M. S. Nair, 2019 Fractional order edge detection with gl fractional derivative mask. Signal, Image and Video Processing **13**: 27–34.

Al-Ataby, A., W. Al-Nuaimy, and M. Al-Taee, 2021 Handwritten digit recognition using support vector machine and other traditional machine learning approaches. Engineering Technology Journal **39**: 1131–1141.

Arora, S., L. Malik, S. Goyal, D. Bhattacharjee, M. Nasipuri, *et al.*, 2024 Devanagari character recognition: A comprehensive literature review. IEEE Access .

Atangana, A. and D. Baleanu, 2016 New fractional derivatives with nonlocal and non-singular kernel: Theory and application to heat transfer model. Thermal Science **20**: 763–769.

Das, S. and P. Gupta, 2013 Edge detection using fractional order differentiation and its applications. Procedia Computer Science **17**: 301–308.

Deng, L., 2012 The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**: 141–142.

Garg, V. and K. Singh, 2012 An improved grunwald-letnikov fractional differential mask for image texture enhancement. International Journal of Advanced Computer Science and Applications **3**: 130–135.

Gonzalez, R. C. and R. E. Woods, 2002 *Digital Image Processing*. Prentice Hall, second edition.

Hasan, M. M., M. A. Hossain, A. Y. Srizon, and A. Sayeed, 2023 Devanagari handwritten character recognition using fine-tuned deep convolutional neural network on trivial dataset. Multimedia Tools and Applications **82**: 20671–20686.

Howard, A. G. *et al.*, 2017 Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 .

Jayadevan, R., S. R. Kolhe, P. M. Patil, and U. Pal, 2011 Offline recognition of devanagari script: A survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **41**: 782–796.

Khaparde, M. S. and V. H. Mankar, 2018 A comprehensive review on ocr for handwritten devanagari script. International Journal of Engineering and Technology **7**: 31–37.

Krizhevsky, A., I. Sutskever, and G. E. Hinton, 2017 Imagenet classification with deep convolutional neural networks. Communications of the ACM **60**: 84–90.

Kumar, R., A. Gaur, K. Bhushan, and M. Singh, 2021 Knowledge extraction in digit recognition using mnist dataset: Evolution in handwriting analysis. International Journal of Applied Engineering Research **16**: 845–852.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner, 1998 Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**: 2278–2324.

LeCun, Y., C. Cortes, and C. J. C. Burges, 2010 Mnist handwritten digit database. Available at http://yann.lecun.com/exdb/mnist.

Liu, C. L., K. Nakashima, H. Sako, and H. Fujisawa, 2003 Handwritten digit recognition: Benchmarking of state-of-the-art techniques. Pattern Recognition **36**: 2271–2285.

Naidu, G., T. Zuva, and E. M. Sibanda, 2023 A review of evaluation metrics in machine learning algorithms. In *Artificial Intelligence Application in Networks and Systems*, edited by R. Silhavy and P. Silhavy, Springer, Cham.

Narang, S. R., M. Kumar, and M. K. Jindal, 2021 Deep net devanagari: A deep learning model for devanagari ancient character recognition. Multimedia Tools and Applications **80**: 20671–20686.

Pal, U. and B. B. Chaudhuri, 2004 Indian script character recognition: A survey. Pattern Recognition **37**: 1887–1899.

Podlubny, I., 1999 *Fractional Differential Equations*. Academic Press.

Ramesh Babu, N., A. S. Joshua, P. Balasubramaniam, and A. Tiwari, 2024 Fuzzy-driven image enhancement via abr-fractal-fractional differentiation. Information Sciences **675**: 120741.

Sandler, M., A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, 2018 Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Sarangi, P. K., S. Singh, and C. Singla, 2020 Feature selection for character recognition of handwritten devanagari and odia scripts. International Journal of Engineering Research and Technology **13**: 1974–1982.

Sethi, N. and A. Singh, 2020 Handwritten devanagari character recognition using deep convolutional neural network. Procedia Computer Science **173**: 215–222.

Sharma, S. and D. Kumar, 2021 A fuzzy-based abr image enhancement technique for improved handwritten character recognition. Journal of Ambient Intelligence and Humanized Computing **12**: 10457–10469.

Yacouby and Axman, 2020 Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In *Proceedings of Eval4NLP*.

Zhao, H. H. and H. Liu, 2020 Multiple classifiers fusion and cnn feature extraction for handwritten digits recognition. Granular Computing **5**: 411–418.

How to cite this article: Akshara, S., Vinodkumar, A., Sriramakrishnan, P., Saravanan, A., and Alzabut, J. Integrating Fuzzy Logic and Fractional-Order Operators in Convolutional Neural Networks for Handwritten Digits and Characters Recognition. *Chaos and Fractals*, 3(2), 92-107, 2026.