Comparative Analysis of Machine Learning and Deep Learning Models for Handwritten Digit Recognition

Soukaina Chekki, Boutaina Hdioud, Rachid Oulad Haj Thami, Sanaa El Fkihi
Advanced Digital Enterprise Modeling and Information Retrieval (ADMIR) Laboratory-Rabat IT Center, Information Retrieval and Data Analytics Team (IRDA), ENSIAS, Mohammed V University in Rabat, Rabat, Morocco

***Abstract*—Handwritten digit recognition (HDR) forms a key component of computer vision systems, especially in optical character recognition (OCR). This study presents a comparative analysis of Machine Learning (ML) algorithms and Deep Learning (DL) models for HDR tasks. A contour-based segmentation technique was applied in preprocessing to enhance feature extraction by detecting digit boundaries and reducing noise. ML models, including K-Nearest Neighbors (KNN) and Support Vector Machine (SVM), and DL architectures, such as Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs), were evaluated on the Modified National Institute of Standards and Technology (MNIST) and the National Institute of Standards and Technology (NIST) datasets. The results demonstrate that DL models significantly outperform ML algorithms in terms of accuracy and robustness, while the KNN model achieved acceptable results. The results underline the importance of contour-based preprocessing in boosting deep learning techniques for HDR.**

Keywords*—Handwritten digit recognition (HDR); Optical Character Recognition (OCR); Machine Learning (ML); Deep Learning (DL); segmentation

I. Introduction

Handwritten text recognition has long been a challenging and active research field, particularly due to the reliance on handcrafted features and prior knowledge in traditional optical character recognition (OCR) systems. Handwritten digit recognition (HDR) has attracted continuous attention over the decades, with advances in computing, artificial intelligence, and imaging since the 1960s driving progress [1], [2], [3], [4]. Although recent developments in deep learning have significantly improved recognition accuracy, the growing demand for large datasets and computational resources continues to motivate research toward more efficient solutions. Robust methods exist for several languages such as English, Chinese, and Japanese [5], [6], [7], [8], [9]; however, recognition of highly variable handwritten digits remains a significant challenge.

This study focuses on offline HDR, where variability in handwriting, overlapping digits, and noise make segmentation a critical step. In particular, separating connected digits remains a difficult problem that directly impacts recognition performance. To address these challenges, we propose a contour-based segmentation approach combined with machine learning and deep learning models. Five models were evaluated: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Artificial Neural Network (ANN), Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN), all widely used in supervised learning and image-based pattern recognition.

Our system was trained on the MNIST dataset and tested on the NIST dataset, which contains numerous complex examples with challenging segmentation cases, making it a particularly suitable benchmark for evaluation. Morphological and geometric operations were integrated to enhance robustness against noisy handwritten images. The proposed approach achieved high recognition accuracy, demonstrating the effectiveness of combining contour-based segmentation with ML and DL techniques.

This work contributes to HDR research by providing a comparative evaluation of machine learning and deep learning approaches while highlighting the critical role of segmentation in offline recognition systems.

The study is organized as follows: Section II reviews related work. Section III presents the proposed methodology. Section IV describes the approaches used for recognizing handwritten digits, including details of all models employed. Section V presents experimental results, and Section VI concludes the study and outlines future work.

II. Related Work

HDR remains a central topic in machine learning due to its practical relevance and the inherent variability of human handwriting. Recent approaches increasingly rely on data-driven methods that integrate traditional algorithms with deep neural networks to enhance recognition performance. Achieving accurate character recognition from image inputs requires effective techniques, particularly in segmentation-based methods [10], [11].

HDR has been extensively investigated using both traditional ML and modern DL techniques. Early and recent studies have explored handcrafted feature extraction methods for HDR. Techniques such as Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), and Gabor filters were widely used to extract discriminative image features. Classifiers including K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Random Forest (RF), and Gradient-Boosted Decision Trees (GBDT) were then applied. For instance, combining HOG features with SVM achieved an accuracy of 93.32% on the NumtaDB dataset [12], highlighting the moderate yet consistent performance of feature-based ML approaches.

Transfer learning using pre-trained CNNs has also been applied, particularly for non-Latin scripts such as Bengali digits.

Networks including ResNet-50, Inception-v3, and EfficientNetB0 were fine-tuned on the NumtaDB dataset containing 17,000 instances across ten classes. These models achieved 97% accuracy, demonstrating the benefits of leveraging pre-trained representations to manage handwriting variability and limited data scenarios [13].

However, more recent studies have reported slightly lower accuracy, reflecting ongoing challenges in HDR. For example, one study reported 97% accuracy on the NumtaDB dataset for Bangla handwritten digits using a pre-trained CNN model [14], [15], while another study achieved 96.9% accuracy with the YOLOv8x model for digit detection [16].

Recent work has further investigated hybrid and ensemble approaches. Hybrid models combining CNN-based feature extraction with SVM classification, as well as ensembles integrating CNNs with traditional ML classifiers, have achieved higher accuracy, up to 99.30%, on benchmark datasets such as MNIST [17], [18]. Additionally, optimization strategies, including mini-batch and Hessian-free training techniques, have been employed to enhance computational efficiency and generalization [19].

In comparison, limited work has been conducted on recognizing handwritten Farsi digits because of their cursive structure and variation in shape, which differ considerably from Latin-based scripts [20], [21]. A key distinction between the Farsi and Arabic languages is the right-to-left writing orientation, which introduces further complexity to the segmentation and recognition tasks [22], [23], [24].

Despite these advances, limitations remain, particularly regarding segmentation robustness and the handling of complex scripts. These challenges motivate the proposed contour-based segmentation approach, which integrates both machine learning and deep learning models. By addressing overlapping and noisy digits, the system aims to improve recognition accuracy on challenging datasets, including MNIST for training and the more complex NIST dataset for testing. Consequently, this work contributes to overcoming current HDR limitations and strengthening offline recognition capabilities.

III. Methodology

HDR systems rely on preprocessing, feature extraction, and classification. This study assesses SVM, KNN, ANN, RNN, and CNN within a contour-based pipeline to identify the best model.

A. General Architecture

Handwritten digit recognition systems are generally based on the following steps, as shown in Fig. 1.

B. Image Acquisition

The acquisition phase captures handwritten text using devices like scanners or digital cameras, converting it into digital images (e.g., JPEG, PNG) with minimal information loss. These images are then preprocessed to improve quality, correct distortions, and remove noise, preparing them for reliable analysis in tasks such as optical character recognition (OCR).

C. Image Preprocessing

Preprocessing prepares images for recognition by applying binarization, noise reduction, normalization, and slant/skew correction [25]. It may also extract handwritten fields and remove backgrounds to isolate the target content.

1) Binarization (Thresholding): Applies either global or adaptive thresholding based on pixel intensity.

2) Noise reduction and normalization: Uses filtering and morphological operations. Normalization corrects slant, skew, and size to reduce variability and improve accuracy [26], [27].

3) Mathematical morphology: Employs operations like erosion, dilation, opening, closing, and morphological gradients for noise removal and contour enhancement.

D. Segmentation

This phase aims to separate graphemes, words, or numerical strings. Segmenting handwritten characters is one of the most challenging tasks in text recognition. A typical application is the recognition of numerical amounts in bank checks, which is particularly difficult because of the presence of noise, overlapping digits, and even the fragmentation of digits into multiple disconnected components, making them difficult to identify [28], [29].

E. Feature Extraction

Feature extraction is vital in character recognition, as classifiers cannot compensate for poor features. It captures relevant information to reduce within-class variability and enhance class separability, yielding numerical feature vectors [30]. Extracted features are generally classified as structural, statistical, global (e.g., transformations), or topological/metric features [31], [32].

F. Classification

After segmentation and feature extraction, classification assigns inputs to classes based on feature vectors, typically through supervised learning involving training and recognition stages. Outcomes include successful classification, confusion between similar classes, or rejection when no adequate match exists. Classifiers vary in performance and robustness, with main types including shape matching, structural methods, neural networks, and statistical classifiers [33].

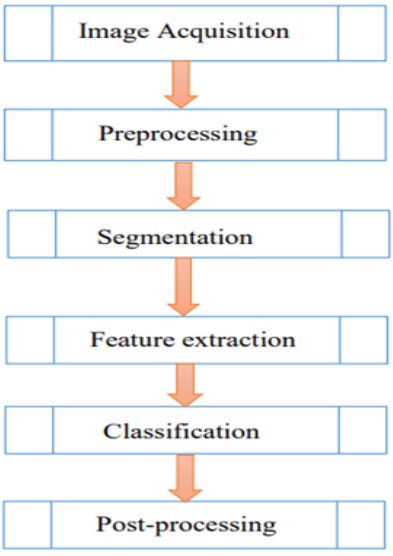

Fig. 1. General diagram of a handwritten digit recognition system.

G. Post-Processing

Post-processing refining recognition results through linguistic, contextual, and structural analysis. It includes:

1) Lexical verification: Checks recognized words against a dictionary, often with edit distance metrics.

2) Orthographic correction: uses n-gram models to correct likely letter sequence errors, such as "tble" to "table".

3) Syntactic validation: Ensures grammatical correctness and suggests corrections [34].

4) Semantic adaptation: Validates word meaning based on context.

5) Pragmatic adaptation: Detects anomalies by comparing recognized words to expected formats.

Post-processing is essential for handling recognition failures, especially those caused by upstream errors or limited training data [35].

IV. Approaches used in Recognizing Handwritten Digits

Various classifiers are used in OCR development, including parametric, non-parametric statistical, and hybrid methods. In this study, after segmenting the relevant images, we applied ML and DL models such as SVM, KNN, ANN, RNN, and CNN on segmented images from the NIST dataset to recognize handwritten digits. Each model was selected for its strengths in handling different data aspects. Performance was evaluated using accuracy, precision, recall, F1-score, confusion matrix, and ROC/AUC, allowing clear comparison to identify the best model.

A. Machine Learning

ML allows computers to learn from data without explicit programming [36]. In handwritten digit recognition, ML typically requires feature extraction before classification. Traditional algorithms such as SVM, decision trees, and Bayesian networks are flat models that depend heavily on feature quality.

1) Support Vector Machine (SVM): SVMs are effective classifiers in HDR, achieving up to 93,32 % accuracy using projection histograms and HOG [37]. As illustrated in Fig. 2, HDR has posed a significant challenge since the 1970s. Its importance lies in applications, such as writer identification, automated data extraction from scanned documents, and various other domains that require reliable character interpretation.

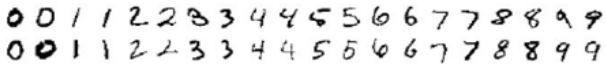

Fig. 2. Digits to recognize [38].

This study aims to accurately identify digits amid symbols and noise, despite overlapping class boundaries caused by similar digit shapes.

Formally, the objective function to be minimized in the context of SVM is computed, as shown in Eq. (1):

$$f(\varepsilon, w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\varepsilon_i \quad (1)$$

where,

- ε_i represents the slack variables for misclassified samples (i.e., points located on the incorrect side of the separating hyperplane).
- $\boldsymbol{C}$ is a regularization parameter that balances the trade-off between maximizing the margin and minimizing classification error.
- $\boldsymbol{w}$ denotes the weight vector that defines the hyperplane orientation.

In such cases, margin-based methods that maximize the distance between the support vectors and the hyperplane are preferred. As illustrated in Fig. 3, this involves optimizing two margin boundaries (e.g., defined by vectors w_1 and w_2) to ensure robust separation, even in the presence of overlapping classes.

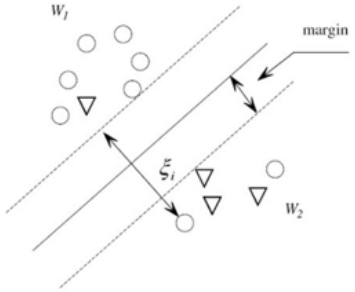

Fig. 3. Feature space accompanied by the margin of error [38].

```
accuracy: 0.9348412698412698

[[1156    0    5    1    0    2   12    6    5    1]
 [   0 1389    4    3    4    0    2    6    5    0]
 [   2    4 1170   19    5    0    9   35   15    1]
 [   1    4   19 1237    0   20    2   32   13    3]
 [   0    4   25    0 1133    0    9   17    3   27]
 [   1    9    5   42    3 1047   17   11    8    6]
 [  13    2    5    0    2   15 1156   24    2    0]
 [   3   10    9    3   15    0    0 1256    1   33]
 [   5   19   14   25    7   21    8   15 1095    8]
 [   4    4    6   30   23    2    1   61    4 1140]]
```

Fig. 4. Confusion matrix of our SVM model.

Fig. 4 shows the SVM confusion matrix on the MNIST dataset, with 93.48% accuracy. High diagonal values indicate strong overall performance, though some misclassifications occur, notably for digit '4'. Performance improvements can be achieved via data augmentation, hyperparameter tuning, feature engineering, or hybrid classifiers. The following sections compare these results with deep learning models on the same dataset.

2) K-Nearest Neighbors (KNN): Among the various classification methods, the KNN is notable for its simplicity and effectiveness. It classifies new observations by identifying the closest instances in the training dataset and assigning the most common class to them [39].

KNN retains all available training examples and uses a distance-based similarity metric to perform classification. It is widely recognized as a *non-parametric lazy learning* algorithm [40] (see Algorithm 1).

Algorithm 1 KNN Algorithm (K-Nearest Neighbors)

Input: A labeled training set $L = \{(x', c)\}$, and a new instance x

Output: Predicted class label for x

1. **For** each example $(x', c) \epsilon L$
   Calculate the distance $D(x, x')$
   End for
2. Select the k nearest neighbors of x, denoted as $kNN(x)$, based on the smallest distances
3. **For** each $x' \epsilon kNN(x)$
   count the occurrences of each class
   End for
4. Assign x the class with the highest frequency among neighbors

This approach measures similarity between test samples and training data, classifying based on a threshold from the k-nearest neighbors. KNN's performance depends on the number of neighbors, similarity metric, threshold, and data representativeness [41]. Trained and tested on MNIST, the Scikit-learn implementation used six neighbors with uniform weights.

B. Deep Learning

1) Artificial Neural Networks (ANN): Fig. 5 illustrates the architecture of the proposed ANN, which consists of an input layer, hidden layer, and output layer [42]. The input layer is connected to the hidden layer through weighted links, denoted by W_{ij}, where i refers to the input neuron and j to the hidden neuron. Similarly, the connections between the hidden and output layers are represented by the weights W_{jk}, where k indicates the output neuron.

A bias term of +1 was integrated into the network to enhance the flexibility of the parameter adjustment. The structure follows a multi-layer perceptron (MLP) configuration.

In this model, the sigmoid activation function was applied to both hidden and output layers. As shown in Eq. (2), the sigmoid maps input values to the [0, 1] range, allowing smooth, bounded output transitions and supporting effective training.

$$g(x) = \frac{1}{1+e^{-x}} \quad (2)$$

In this context, $g(x)$ denotes the sigmoid activation function, and x represents the net input, that is the weighted sum of the inputs to a neuron.

2) Réseaux De Neurones Récurrents (RNN) : Unlike MLPs, RNNs (Fig. 6) have cycles in their computational graphs, enabling memory retention [43]. Unrolled over time (Fig. 7), they form chains of MLPs with recurrent links, capturing temporal dependencies, which makes them well-suited for HDR tasks.

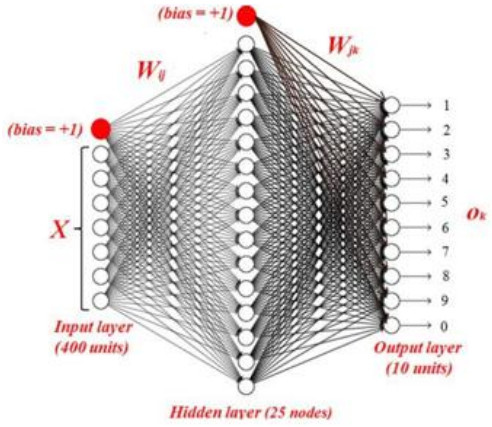

Fig. 5. The suggested architecture for the neural network.

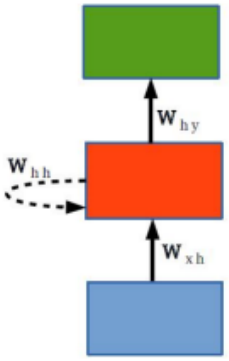

Fig. 6. Compact representation of RNN. All arrows depict complete connections. The dashed arrow represents connections with a temporal shift (t − 1).

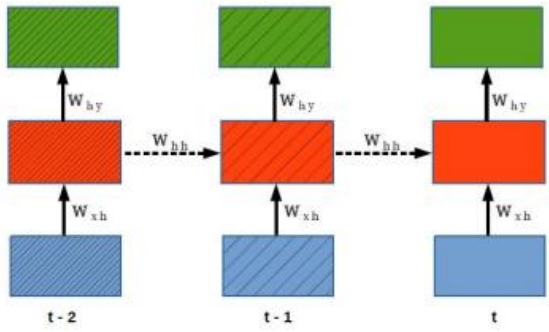

Fig. 7. Unfolded representation of RNN [43].

According to this model, an RNN takes a sequence of events $x = (x1, x2, \dots, xT)$ as the input and defines the sequence of hidden states $h = (h1, h2, \dots, hT)$ to produce the sequence of output vectors $y = (y1, y2, \dots, yT)$ by iterating from t=1 to T :

$$h_t = H(W_{x_h} x_t + W_{h_h} h_{t-1} + b_h) \quad (3)$$

$$y_t = W_{h_y} h_t + b_y \quad (4)$$

where, T is the total number of input vectors, $W_{\alpha\beta}$ is the weight matrix between layers α and β, and $b\beta$ is the bias vector in layer β. The activation function H used in RNNs is generally a hyperbolic tangent (tanh).

Traditional backpropagation, designed for non-sequential networks, cannot capture temporal dependencies, as shown in Eq. (3) and Eq. (4). RNNs address this limitation by unrolling over time, enabling Backpropagation Through Time (BPTT) to train the network effectively [44].

The distinctive feature of this representation, as opposed to a 'classic' non-recurrent neural network, is the presence of multiple shared parameters. For instance, a common weight matrix W_{h_h} conveys information through diagonal connections. Additionally, the weight matrices, W_{x_h} and W_{h_y} depicted by the vertical connections, are shared across time steps.

3) Convolutional Neural Network (CNN): CNNs have gained popularity in recent years due to their effectiveness in image recognition. A typical CNN includes convolutional, pooling, and fully connected layers, often with ReLU activation. Training involves a feed-forward phase to extract features and a back-propagation phase for weight updates [45].

In the feed-forward phase, an input image passes through the network, undergoing convolutional operations where neurons apply learned filters to extract features. This process transforms the input through successive layers until the final output is produced [46], [47].

As shown in Fig. 8, a CNN consists of an input layer, an output layer, and multiple hidden layers, including convolutional, pooling (e.g., max and average pooling), normalization, and fully connected layers. Filters extract spatial features, while non-linear activations add complexity. As data flows through the network, spatial dimensions shrink and feature channels grow, leading to the final prediction.

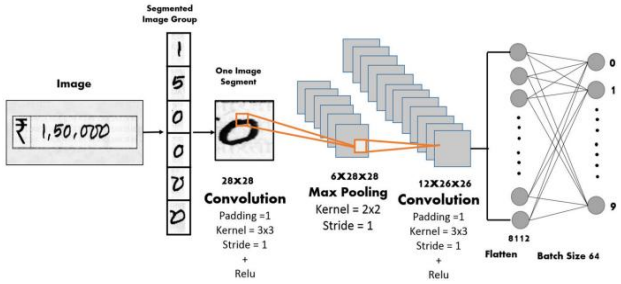

Fig. 8. CNN architecture for handwritten amount digit recognition [47].

V. Implementation and Results

The primary objective of this study was to implement and evaluate various models and architectures for HDR. Initially, contour-based segmentation was applied to extract the digit regions from the input images. Subsequently, a range of machine learning and deep learning models, namely, SVM, KNN, ANN, CNN, and RNN, were employed to assess their performance and determine the most effective approach for this recognition task.

In this study, the models were trained on the MNIST dataset and tested on the NIST dataset, since NIST contains handwritten digits that are more challenging to classify. This setup was designed to evaluate the cross-dataset generalization capability of the proposed approach. The differences in style and noise between the two datasets represent a real challenge, and our methodology highlights the robustness of the proposed solution under such conditions. Furthermore, we discuss the potential of applying domain adaptation techniques to further enhance performance on the NIST dataset.

A. Dataset Description

1) MNIST dataset: Fig. 9 presents the MNIST dataset, a benchmark collection of 70,000 grayscale images of handwritten digits compiled by NIST. Each 28×28 image represents one of ten digit classes (0 to 9). The dataset is split into 60,000 training and 10,000 testing samples, each labeled accordingly.

MNIST was selected for this study due to its standardized format, broad use in literature, and relevance for offline HDR. Its consistency, ease of use, and compatibility with various machine learning models make it ideal for performance evaluation.

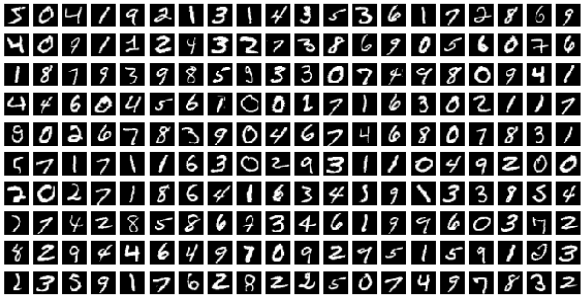

Fig. 9. Handwritten digits from the MNIST database.

2) NIST dataset: The specialized database illustrated in Fig. 10 encompasses the complete training dataset used by NIST for the recognition of handwritten text and characters in printed documents. It contains handwritten samples contributed by 3,600 writers and includes 810,000 isolated character images extracted from these forms. The dataset also provides ground-truth classifications for each image, reference templates to support future data-collection efforts, and a suite of software tools for image processing and annotation.

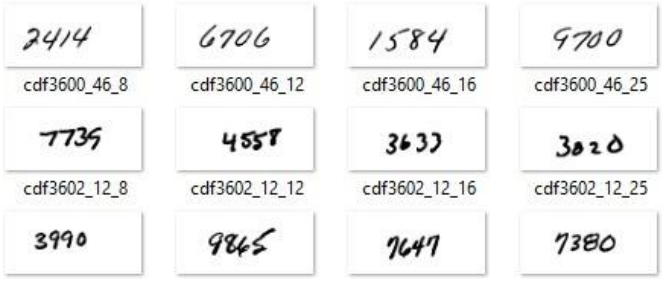

Fig. 10. Handwritten digits from the NIST database.

B. Contour Segmentation

Contour-based segmentation proposed in this study is more effective than classical methods. Our approach implements multiple specific steps for digit segmentation, which allows better isolation of each character, reduces noise impact, and improves the quality of segmented regions. Unlike standard methods that use simple binarization or contour detection, our

multi-step process preserves digit integrity and enhances performance, particularly in cross-dataset generalization trained on MNIST and tested on NIST.

This section presents handwritten digit recognition using ML and DL models, including KNN, SVM, ANN, RNN, and CNN. MNIST was used for training, NIST for testing, and OpenCV for preprocessing the digit images.

Steps in the Handwritten Digit Recognition Pipeline Using Contour Segmentation:

1) Image acquisition: Handwritten digit images were collected from the MNIST dataset.

2) Dataset splitting: The data were separated into a training set (MNIST) and a testing set (NIST).

3) Preprocessing: Contour-based segmentation techniques were applied to clean and prepare the images.

4) Normalization: Pixel intensity values were scaled to a range between 0 and 1.

5) Batching: The training dataset was divided into mini-batches to enhance training efficiency.

6) Model training: ML and DL models were trained using labeled data.

7) Classification: The trained models were used to classify digits in the testing dataset.

8) Performance evaluation: Recognition accuracy and processing time were evaluated for each model variant.

C. Implementations

1) Image preprocessing: Fig. 11 illustrates a sample image from the NIST database, which may exhibit noise, spots, or other unwanted artifacts. To mitigate these imperfections, a Gaussian filter was applied to reduce the noise, followed by a derivative filter for edge detection. A sharpening technique based on a blur mask was employed to enhance image clarity, as shown in Fig. 12.

Fig. 11. Initial image.

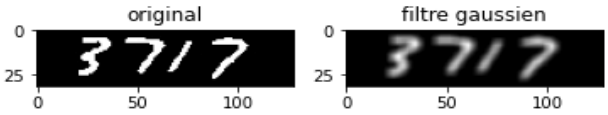

Fig. 12. Gaussian smoothing on the image.

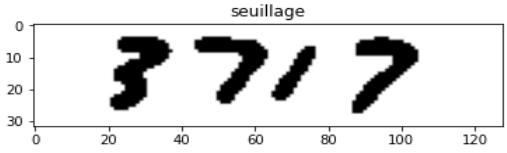

Fig. 13. Thresholding "seuillage" application.

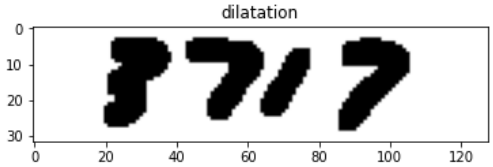

Fig. 14. Applying dilation to the image.

Thresholding was applied, as illustrated in Fig. 13, using Otsu's method to determine the optimal global threshold based on the image histogram. This binary thresholding approach enhances the contrast between foreground and background. As shown in Fig. 14, dilation was selectively applied to images containing small handwritings to capture their contours more effectively. In addition, the Canny edge detector, known for its effectiveness in edge detection, was employed in the preprocessing phase, as illustrated in Fig. 15.

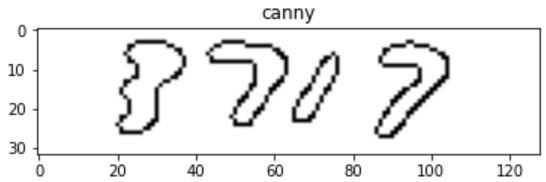

Fig. 15. Application of Canny Edge Detection.

2) Contour segmentation application: In the segmentation phase, we applied a contour detection algorithm to isolate individual digits within the image. A contour is defined as a closed curve that connects all the continuous points along the boundary of an object, representing its shape, as illustrated in Fig. 16. To accomplish this, one function was used to locate the contours, and the other was used to draw them.

Subsequently, as shown in Fig. 17, we combined contour-based segmentation with vertical segmentation. The latter divides the image into vertical slices to separate connected digits, thereby facilitating more accurate digit recognition.

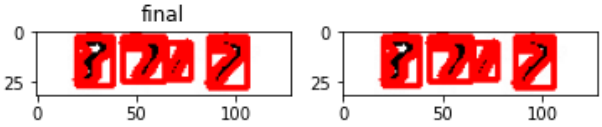

Fig. 16. Contour segmentation on the NIST dataset image digits.

Fig. 17. Vertical segmentation application.

D. Model hyperparameters

The CNN model includes a convolutional layer with 32 filters and ReLU activation, followed by a flattening layer, and then two dense layers: the first with 100 units and ReLU activation, and the second with 10 units and softmax activation. We used the SGD optimizer (learning rate of 0.01, momentum of 0.9) and the 'categorical_crossentropy' loss function, as shown in Table III.

TABLE I. ANN HYPERPARAMETERS

Number of epochs	100
Batch size	1000
Activation function	Softmax & Sigmoid
Optimizer	Adam
Loss	sparse_categorical_crossentropy

TABLE II. RNN HYPERPARAMETERS

Number of epochs	50
Batch size	128
Units	256
Dropout	0.2
Activation function	Softmax
Optimizer	SGD
Loss	categorical_crossentropy

TABLE III. CNN HYPERPARAMETERS

Number of epochs	100
Batch size	1000
Activation function	Relu & Softmax
Optimizer	Adam
Loss	categorical_crossentropy

As illustrated in Table I, the ANN model was trained for 100 epochs with a batch size of 1000 using Softmax and Sigmoid activation functions and the Adam optimizer, with 'sparse_categorical_crossentropy' as the loss function.

To prevent overfitting, the RNN model, which includes 256 units and a dropout rate of 0.2 to prevent overfitting, uses Softmax activation and is optimized with SGD, as shown in Table II. The loss function used is also 'categorical_crossentropy', which is optimized for multi-class classification tasks.

The hyperparameters for all models were selected based on experimental validation to ensure stable convergence, prevent overfitting, and achieve optimal performance on both MNIST and NIST datasets. Ablation studies were performed to evaluate the impact of varying key parameters (e.g., number of units, activation functions, learning rate, dropout), confirming that the chosen configuration consistently provides robust and reproducible results.

E. Analysis of Results

During testing, this study proposes a comprehensive method to evaluate all digit cases from the NIST dataset across multiple models. A structured framework was developed to facilitate testing on different models by grouping each test case within a structure that consolidates the results of all applied models. Unlike previous studies, which performed tests on the entire dataset without a method allowing an easy linkage of each model to its corresponding test, this approach ensures clear, systematic, and robust performance comparisons, as illustrated below. This methodology confirms the relevance of the observed improvements.

We performed handwritten digit recognition using five different models (KNN, SVM, CNN, RNN, and ANN) to identify the most accurate model. As shown in Fig. 18, the deep learning models (ANN, RNN, and CNN) outperformed the others in both test cases. Similarly, Fig. 19 shows that the RNN and CNN achieved the best results across all the digits in the test case. Furthermore, Fig. 20 illustrates that the RNN consistently outperformed the CNN and other models across all three test scenarios.

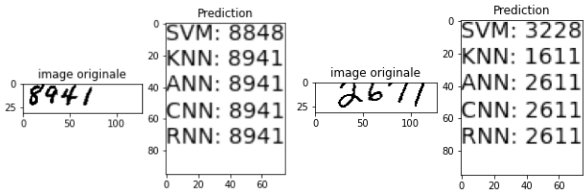

Fig. 18. In both test cases, Deep Learning models {ANN, RNN, CNN} showed superior performance.

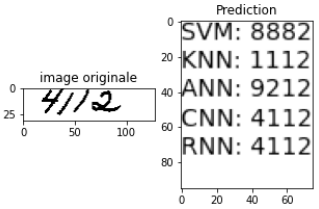

Fig. 19. RNN and CNN models achieved the best results for all digits in this test.

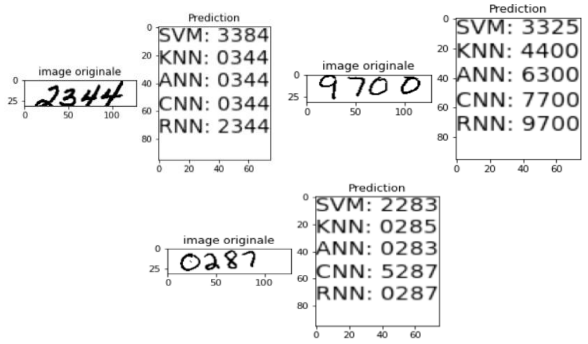

Fig. 20. RNN models outperformed CNN and other models across all digits in these three tests.

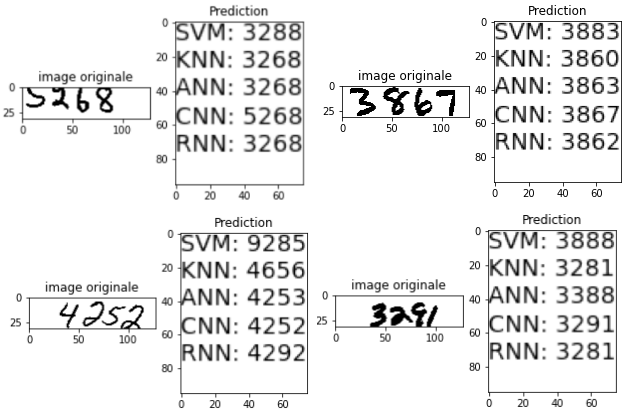

Fig. 21. CNN models performed better than other models for all digits in these examples.

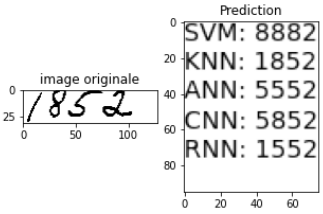

Fig. 22. In this test example, KNN models outperform others and correctly identify at least 3 out of 4 digits in other tests.

In the test examples shown in Fig. 21, the CNN models outperformed the other models for all the digits. As shown in Fig. 22, the KNN models also demonstrated superior performance for all the digits. In addition, in other test examples, the KNN models provided acceptable results, correctly identifying at least three of the four handwritten digits.

TABLE IV. RESULTS TABLE DISPLAYING THE ACCURACY OF EACH MODEL

Models	Accuracy (%)
SVM	93.48
KNN	92.9
ANN	99.56
RNN	98.74
CNN	98.53

Table IV lists the accuracy results for each model on the MNIST image dataset. In our handwritten digit recognition project using the MNIST dataset, we evaluated several ML and DL models: SVM, KNN, ANN, CNN, and RNN. The results showed that the DL models outperformed the traditional machine learning models. In particular, ANN achieved the highest accuracy of 99.56%, followed by RNN at 98.74% and CNN at 98.53%. SVM and KNN achieved accuracies of 93.48% and 92.9%, respectively. Prediction analysis for the digit sequence '1852' highlighted KNN's strength in certain specific cases, whereas other models had variable prediction accuracies. Overall, the DL models proved to be more effective for this task.

TABLE V. COMPARISON OF RESULTS WITH RECENT WORKS

Models	Our Result (%)	Previous Results (%)	References of Previous Works
SVM	93.48	92.3- 93	[48]
KNN	92.9	91.8- 92.3	[49]
ANN	99.56	97 - 97.4	[50]
RNN	98.74	97 - 98	[51]
CNN	98.53	98 – 98.12	[52]

Our model outperformed previous approaches across all tested algorithms, as shown in Table V, with significant improvements in the ANN (99.56%) and RNN (98.74%). These enhancements suggest effective optimization of the architecture and hyperparameters. Slight increases were also observed for SVM, KNN, and CNN, confirming the robustness of our approach compared to previous works.

VI. CONCLUSION AND FUTURE WORK

This study developed an offline handwritten digit recognition system and compared various ML and DL models to identify the most effective. After preprocessing, a contour-based segmentation method was implemented and evaluated across multiple classifiers. The study assessed features and models to determine the best HDR approach.

Segmentation is challenging. Our system uses morphological and geometric operations with contour detection to isolate digits and reduce noise, producing segmented images of single digits for training.

We performed experiments on the NIST dataset using contour-based segmentation and a range of algorithms: KNN, SVM, ANN, CNN, and RNN. The results showed that CNN and RNN delivered the best recognition performance, with high accuracy and consistency. Among the ML methods, KNN achieved acceptable results despite the difficulty in recognizing unclear digits in the NIST dataset.

SVM demonstrated the fastest training time and achieved the highest training accuracy among the models evaluated. However, its simplicity limits its ability to classify ambiguous or complex inputs compared with DL models. By contrast, the DL models (CNN, RNN, and ANN) provided significantly better results, reaching up to 99% accuracy during both training and testing.

Overall, nearly all the algorithms yielded acceptable results on the NIST test set, confirming their suitability for HDR tasks. While the SVM had the shortest execution time, the CNN required the longest execution time.

In future work, we plan to enhance recognition accuracy by focusing entirely on deep learning techniques, exploring alternative architectures, and optimizing learning strategies. Our approach will be applied to more complex, real-world datasets, such as handwritten digits from bank checks. Furthermore, we aim to investigate adaptive segmentation methods to better handle overlapping digits and challenging handwriting styles, providing a clear roadmap for advancing offline handwritten digit recognition systems.

REFERENCES

[1] M. Islam, A. Rahman, and S. Khan, “Pre-trained CNN Models for Bengali Handwritten Digit Recognition,” arXiv preprint arXiv:2209.13005, 2022.

[2] A. Fateh, R. Singh, and P. Sharma, “Handwritten Digits Recognition Using Transfer Learning with Attention Mechanisms,” *ResearchGate*, 2024.

[3] S. Keshari, P. Jain, and R. Kumar, “Transfer Learning for Handwritten Character Classification with Limited Training Data,” *Journal of Image and Graphics*, vol. 11, no. 1, 2023. [Online]. Available: https://www.joig.net/uploadfile/2023/JOIG-V11N1-21.pdf.

[4] M. Ullah, A. Khan, S. Ali, and J. Kim, “Hybrid CNN-SVM Model for Handwritten Digit Recognition on MNIST Dataset,” arXiv preprint arXiv:2503.06104, 2025.

[5] A. Fateh, R. T. Birgani, M. Fateh, and V. Abolghasemi, “Advancing Multilingual Handwritten Numeral Recognition With Attention-Driven Transfer Learning,” *IEEE Access*, vol. 12, pp. 41381–41395, 2024, doi: 10.1109/ACCESS.2024.3245678.

[6] D. Ghosh, D. Chaurasia, S. Mondal, and A. Mahajan, “Handwritten Documents Text Recognition with Novel Pre-processing and Deep Learning,” in *Proc. 2021 Grace Hopper Celebration India (GHCI)*, IEEE, 2021, pp. 1–5, doi: 10.1109/GHCI.2021.00012.

[7] D. Gupta and S. Bag, "CNN-Based Multilingual Handwritten Numeral Recognition: A Fusion-Free Approach," *Expert Systems with Applications*, vol. 165, 113784, 2021, doi: 10.1016/j.eswa.2020.113784.

[8] B. Vidhale, G. Khekare, C. Dhule, P. Chandankhede, A. Titarmare, and M. Tayade, "Multilingual Text & Handwritten Digit Recognition and Conversion of Regional Languages into Universal Language Using Neural Networks," in *Proc. 2021 6th International Conference for Convergence in Technology (I2CT)*, IEEE, 2021, pp. 1–5, doi: 10.1109/I2CT51072.2021.9413545.

[9] A. Fateh, M. Fateh, and V. Abolghasemi, "Multilingual Handwritten Numeral Recognition Using a Robust Deep Network Joint with Transfer Learning," *Information Sciences*, vol. 581, pp. 479–494, 2021, doi: 10.1016/j.ins.2021.01.097.

[10] S. Alghyaline, "Arabic optical character recognition: A review," *Computer Modeling in Engineering & Sciences*, vol. 135, no. 3, pp. 1825–1861, 2023, doi: 10.32604/cmes.2022.024555.

[11] H. Kusetogullari, A. Yavariabdi, J. Hall, and N. Lavesson, "DIGITNET: A deep handwritten digit detection and recognition method using a new historical handwritten digit dataset," *Big Data Research*, vol. 23, p. 100182, Feb. 2021, doi: 10.1016/j.bdr.2020.100182.

[12] M. Rahman et al., "Handwritten Digit Recognition Using Feature-Based Machine Learning Approaches," arXiv preprint arXiv:2201.10102, 2022.

[13] S. Chakraborty et al., "Bengali Handwritten Digit Recognition Using Pre-trained Deep Learning Models," arXiv preprint arXiv:2209.13005, 2022.

[14] M. Islam, S. A. Shuvo, M. S. Nipun, R. B. Sulaiman, J. Nayeem, Z. Haque, M. M. Shaikh, and M. S. Ullah Sourav, "Efficient approach of using CNN based pretrained model in Bangla handwritten digit recognition," *arXiv preprint arXiv:2209.13005*, 2022.

[15] A. Choudhary and S. Ahlawat, "Hybrid CNN-SVM Classifier for Handwritten Digit Recognition," *ResearchGate*, 2025.

[16] D. O. I. Effendi, "Handwritten Digits Detection Using Convolutional Neural Network," Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI), vol. 11, no. 2, pp. 346–356, Jun. 2025.

[17] A. Kumar et al., "Hybrid CNN-SVM Approach for Handwritten Digit Recognition with Data Augmentation," arXiv preprint arXiv:2503.06104, 2025.

[18] J. Smith et al., "Ensemble Methods for Handwritten Digit Recognition: Combining CNN and Traditional ML Classifiers," arXiv preprint arXiv:2503.06105, 2025.

[19] L. Zhang et al., "Optimization of CNN Models for Handwritten Digit Recognition Using Mini-batch and Hessian-Free Techniques," arXiv preprint arXiv:2311.01022, 2023.

[20] H. Noori, "Improving Persian Handwritten Digit Recognition using Convolutional Neural Network," *Journal of Algorithms and Computation*, vol. 56, no. 1, pp. 55–72, 2024, doi: 10.22059/jac.2024.374614.1213.

[21] F. Jawad and N. R. Hamza, "Farsi Digit Recognition Using GAN-Generated Data and Convolutional Neural Networks," *Journal of Kufa for Mathematics and Computer*, vol. 12, no. 1, pp. 6–11, May 2025, doi: 10.31642/JoKMC/2018/120102.

[22] A. Nasr-Esfahani, M. Bekrani, and R. Rajabi, "Robust Persian Digit Recognition in Noisy Environments Using Hybrid CNN-BiGRU Model," *arXiv preprint arXiv:2412.10857*, Dec. 2024.

[23] A. Zohrevand, Z. Imani, J. Sadri, and C. Y. Suen, "Does color modalities affect handwriting recognition? An empirical study on Persian handwritings using convolutional neural networks," *arXiv preprint arXiv:2307.12150*, Jul. 2023.

[24] M. Bonyani, S. Jahangard, and M. Daneshmand, "Persian handwritten digit, character and word recognition using deep learning," *International Journal on Document Analysis and Recognition*, vol. 24, no. 1–2, pp. 133–143, Apr. 2021, doi: 10.1007/s10032-021-00368-2.

[25] *Machine Vision Inspection Systems: Image Processing, Concepts, Methodologies, and Applications*, dokumen.pub. [Online]. Available: https://dokumen.pub/machine-vision-inspection-systems-image-processing-concepts-methodologies-and-applications-1119681804-9781119681809.html. Accessed: Nov. 20, 2023.

[26] S. Amiri, "Image Denoising to Enhance Character Recognition Using Deep Learning," ResearchGate, 2021. [Online]. Available: https://www.researchgate.net/publication/352147349_Image_Denoising_to_enhance_Character_Recognition_using_Deep_Learning.

[27] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, and L. Shao, "Normalization Techniques in Training DNNs: Methodology, Analysis and Application," *arXiv preprint arXiv:2009.12836*, 2020.

[28] A. Chakraborty, R. De, S. Malakar, F. Schwenker, et R. Sarkar, *Full segmentation system for handwritten digit string recognition*, 2021.

[29] D. Prabha Devi, R. Ramya, P. S. Dinesh, C. Palanisamy, and G. Sathish Kumar, "Design and simulation of handwritten recognition system," *Materials Today: Proceedings*, vol. 45, pp. 626–629, Jan. 2021, doi: 10.1016/j.matpr.2020.02.720.

[30] Wang Chao, "Research on Features Extraction and Classification for Images based on Transformer Learning," *Proc. 2024 International Conference on Machine Learning and Intelligent Computing*, PMLR, vol. 245, pp. 67–75, Apr. 2024.

[31] Rafal Ali Sameer, "A General Overview on the Categories of Image Features Extraction Techniques: A Survey," *J. of Al-Qadisiyah for Computer Science and Mathematics*, vol. 15, no. 1, pp. 179–188, Apr. 2023.

[32] M. Ferdous Wahid, Md. Fahim Shahriar, and Md. Shohanur Islam Sobuj, "A Classical Approach to Handcrafted Feature Extraction Techniques for Bangla Handwritten Digit Recognition," *arXiv preprint arXiv:2201.10102*, Jan. 2022.

[33] A. Adhesh Garg et al., "An Efficient CNN-ELM Classifier for Handwritten Digits Recognition and Classification," *Symmetry*, vol. 12, no. 10, Article 1742, 2020, doi: 10.3390/sym12101742.

[34] عنابت - مختار باجي جامعت, [Online]. Available: https://docplayer.fr/120345916-Jm`t-bjy-mkhtr-`nbt.html. Accessed: Nov. 20, 2023.

[35] A. Joshi, I. [surname missing], H. Pandey, and Y. R. Choudhary, "Handwritten Character Recognition by Using Machine Learning: A Review," SSRN, 2023, [Online]. Available: https://ssrn.com/abstract=4489054.

[36] *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers* [PDF]. [Online]. Available: https://www.researchgate.net/publication/277299933_Efficient_Learning_Machines_Theories_Concepts_and_Applications_for_Engineers_and_System_Designers. Accessed: Nov. 20, 2023.

[37] M. Wahid, M. M. Rahman, M. M. Hasan, et M. A. Hossain, "Handwritten Digit Recognition Using Feature-Based Machine Learning Approaches," *arXiv preprint arXiv:2201.10102*, 2022. [Online]. Available: https://arxiv.org/abs/2201.10102.

[38] W. Homenda, A. Jastrzebska, and W. Pedrycz, "Rejecting foreign elements in pattern recognition problem — reinforced training of rejection level," in *Proc. International Conference on Agents and Artificial Intelligence*, Lisbon, Portugal, 2015, pp. 90–99, doi: 10.5220/0005207900900099.

[39] S. Arya, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions."

[40] R. Karakaya and S. Kazan, "Handwritten digit recognition using machine learning," *Sakarya University Journal of Science*, vol. 25, no. 1, pp. 65–71, Feb. 2021, doi: 10.16984/saufenbilder.801684.

[41] S. Chauhan, S. Mahmood, and T. Poongodi, "Handwritten digit recognition using deep neural networks," *Proc. [Nom de la conférence]*, May 2023, pp. 1–6, doi: 10.1109/ICIEM59379.2023.10167391.

[42] S. Al-Mansoori, "Intelligent handwritten digit recognition using artificial neural network," 2015, doi: 10.13140/RG.2.1.2466.0649.

[43] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, Apr. 1990, doi: 10.1016/0364-0213(90)90002-E.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, D. E. Rumelhart and J. L. McClelland, Eds., Cambridge, MA, USA: MIT Press, 1988, pp. 399–421, doi: 10.1016/B978-1-4832-1446-7.50035-2.

[45] Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). *A Survey of the Recent Architectures of Deep Convolutional Neural Networks.* Artificial Intelligence Review, 53, 5455–5516.

[46] Nalluri, S. D., Thota, S. M., & Thota, A., "Handwritten Digit Recognition Using CNN," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 11, no. 1, pp. 8849–8855, 2023.

[47] P. Agrawal *et al.*, "Automated bank check verification using image processing and deep learning methods," *Multimedia Tools and Applications*, vol. 80, no. 4, pp. 5319–5350, Feb. 2021, doi: 10.1007/s11042-020-09818-1.

[48] S. Ahlawat, A. Choudhary, A. Nayyar, S. Singh, and B. Yoon, "Improved handwritten digit recognition using convolutional neural networks (CNN)," *Sensors*, vol. 20, no. 12, p. 3344, Jun. 2020, doi: 10.3390/s20123344.

[49] "Comparisons of KNN, SVM, BP and CNN for handwritten digit recognition," ResearchGate, doi: 10.1109/AEECA49918.2020.9213482.

[50] A. Ashiquzzaman and A. K. Tushar, "Handwritten Arabic numeral recognition using deep learning neural networks," arXiv preprint arXiv:1702.04663, Feb. 15, 2017, doi: 10.48550/arXiv.1702.04663.

[51] S. Sarraf, "French word recognition through a quick survey on recurrent neural networks using long-short term memory RNN-LSTM," arXiv preprint arXiv:1804.03683, Apr. 10, 2018, doi: 10.48550/arXiv.1804.03683.

[52] M. Husnain et al., "Recognition of Urdu handwritten characters using convolutional neural network," Applied Sciences, vol. 9, no. 13, Art. no. 132758, Jan. 2019, doi: 10.3390/app9132758.