IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Mathematical Modelling of Engineering Problems
Vol. 12, No. 12, December, 2025, pp. 4173-4190
Journal homepage: http://iieta.org/journals/mmep

A Novel Swarm-Based Hybridization of Puma Optimizer and Crested Porcupine Algorithm for Complex Handwriting Recognition in Writer Identification

Asmaa N. Khaleel[1*], Raya Akram Hamdi[2], Husham Y. A. Alameen[3]

[1] College of Electronics Engineering, Ninevah University, Mosul 41002, Iraq
[2] College of Administration and Economics, University of Mosul, Mosul 41002, Iraq
[3] College of Engineering, Mechatronics Engineering Department, University of Mosul, Mosul 41002, Iraq

Corresponding Author Email: asmaa.khaleel@uoninevah.edu.iq

https://doi.org/10.18280/mmep.121207

Received: 7 August 2025
Revised: 15 November 2025
Accepted: 21 November 2025
Available online: 31 December 2025

Keywords:
handwriting recognition, capsule networks, feature reduction, puma-crested porcupine optimizer, deep learning

ABSTRACT

Accurate handwriting recognition is a challenging problem, owing to variability in handwriting styles, distortions in writing patterns, and noise in handwritten documents. These challenges are even more severe in scripts like Devanagari and Arabic, which have complex character forms and high visual similarity among classes, requiring strong feature extraction and semantic knowledge. To address these challenges, we have developed a novel deep-learning-based handwriting recognition system that preserves the intrinsic writing dynamics and recognizes hierarchical spatial cues through multi-level abstraction and attention-driven encoding. Our framework synergistically integrates a Residual Abstraction Block, Spatial Context Encoder, Spatial Attention Generator, and Hierarchical Capsule Encoding Block to capture fine-grained spatial dependencies and contextual semantics efficiently. To further improve efficiency, we propose a hybrid puma-crested porcupine optimizer for Feature Reduction (FR), which significantly reduces the model's complexity without compromising accuracy. Extensive experiments on the Devanagari and KHATT datasets prove the effectiveness of our method. Our proposed model achieves superior recognition accuracy of 98.94% (with FR) and 93.28% (without FR) on Devanagari, and 97.36% (with FR) and 91.91% (without FR) on KHATT, outperforming various baseline methods. These findings demonstrate the robustness of our architecture in achieving high accuracy, compactness, and resilience.

1. INTRODUCTION

Handwritten Character Recognition (HCR) has become a central field of study in pattern recognition and machine learning due to its widespread applications in biometric identification, document authentication, and forensic examination [1]. Specifically, writer identification, or the identification of an individual's distinctive handwriting style, is essential for security systems, historical document analysis, and criminal investigations [2]. However, intricate handwriting styles, inconsistent writing patterns, and noise in handwritten texts pose great challenges for reliable writer identification [3].

Conventional methods were primarily based on handcrafted local descriptors, such as Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), Local Binary Patterns (LBP), and Histogram of Oriented Gradients (HOG), which are robust to small distortions, affine variations, and slight overlaps; however, they fail to capture higher-level semantic information and contextual meaning in handwriting [4-8]. Deep learning techniques, particularly Convolutional Neural Networks (CNNs), facilitated data-driven feature learning by automatically extracting hierarchical representations from raw handwriting images [9]. However, traditional CNNs tend to fail to capture the dynamic, fine-grained, and context-dependent nature of handwriting variations [10]. They primarily emphasize local patterns and lack an explicit mechanism to model spatial relationships and salient regions, which are essential for distinguishing writing styles [11, 12].

To overcome the limitations mentioned above, our study aims to develop a hybrid deep learning architecture named Curve-Aware Capsule Residual (CA-CapsResNet) CNN. The model integrates three modules: (1) Residual Abstraction Block (RAB), (2) Spatial Context Encoder (SCE), and (3) Spatial Attention Generator (SAG), to extract, refine, and boost the hierarchical and spatial nature of handwriting features. First, the RAB employs a combination of zero-padded convolutions and residual connections to preserve fine-grained spatial locality while generating a shallow, discriminative feature map. Then, the SCE utilizes depthwise separable convolutions, lightweight channel attention, and global context aggregation to refine local structures further and selectively enhance semantically rich features. Lastly, the SAG calculates a location-aware attention map that directs the model to emphasize salient spatial areas in separating various

handwriting styles. Finally, a hierarchical capsule network [13] is introduced to extract high-dimensional fine-grained spatial features from the extracted salient regions.

To further enhance classification accuracy and minimize the risk of overfitting, a Hybrid Crested Porcupine-Puma Optimizer (HCPPO) is introduced for optimal feature selection. This approach synergistically combines the exploratory search scheme of Crested Porcupine [14] and the exploitative refinement technique of the Puma algorithm [15] to achieve a better exploration-exploitation balance than traditional feature selection techniques. Furthermore, the integration of phase-based decision-making, binary encoding, adaptive randomization, and a convergence-aware control strategy ensures that the optimizer effectively reduces the larger search space with an optimal feature set and higher accuracy.

The main contributions of this work are briefly stated below:

- We introduce the CA-CapsResNet, which integrates an RAB, an SCE, and an SAG to extract and refine hierarchical handwriting features.
- The RAB retains fine-grained spatial information through zero-padded convolutions and residual connections, generating shallow, discriminative feature maps.
- The SCE enhances semantically dense features by applying depthwise separable convolutions and lightweight attention, and the SAG focuses on key handwriting areas.
- Our proposed HCPPO framework utilizes adaptive phase transitions, binary encoding, and convergence control policies to obtain minimal yet highly discriminative feature subsets, thereby enhancing classification accuracy while reducing complexity.
- We conduct extensive five-fold cross-validation experiments on the Devanagari Hindi and KHATT datasets to examine the robustness and generalization ability of the proposed framework.

The rest of the paper is structured as follows: Section 2 discusses relevant works on metaheuristic algorithms for handwriting recognition and hybrid optimization methods. Section 3 outlines the proposed methodology, including (1) Preprocessing, (2) Feature Extraction (CA-CapsResNet), (3) Optimal Feature Selection using the HCPPO algorithm, and (4) Classification. Section 4 presents the experimental setup, benchmark datasets, and experimental results, along with a comparative analysis of the results with previous methods. Lastly, Section 5 concludes the findings and proposes directions for future research.

2. RELATED WORK

This section provides an overview of recent research into handwritten character recognition and optimization-based learning. To begin, Subsection 2.1 surveys important baseline studies across capsule, CNN, and hybrid-based handwritten character recognition. In turn, Subsection 2.2 applies a comparative review based on the underlying principles and limitations of the two approaches. Finally, Subsection 2.3 explains how the proposed CA-CapsResNet model with HCPPO converges the gaps outlined in previous studies and advances the field beyond the previous work.

2.1 Baseline studies in capsule, CNN, and hybrid models

Yao et al. [16] introduced FOD_DCNet, a deep capsule network for fully overlapping handwritten digit recognition and separation. The model implemented small convolutional kernels and an improved series dual dynamic routing collocation mechanism, increasing routing effectiveness. Accuracy was reported to be 93.53%, with almost half the number of parameters compared to a regular CapsNet. Parcham et al. [17] introduced CBCapsNet by integrating CNN with capsule networks to improve spatial feature learning in the signature verification task. The architecture employed a paired image scheme to reduce the training parameters by half, thereby avoiding the need for dual networks. The proposed network achieved a classification accuracy of 92.94% with a low Feature Acceptance Rate (FAR) and a high Feature Reduction Rate (FRR). Moudgil et al. [18] presented a CapsNet framework for identifying Devanagari characters, comprising 399 classes, with an accuracy of 94.6%, which outperformed conventional models such as CNN, MLP, and KNN.

In handwriting recognition, feature selection has also received considerable attention. In a recent work, Abd Elaziz et al. [19] integrated the Freeman Chain Code (FCC) with the Whale Optimization Algorithm (WOA) to enhance the data representation of handwritten text images. This step improved the feature reduction and convergence rate of the WOA compared to the Flower Pollination Algorithm (FPA). Al-Saffar et al. [20] merged a Dynamically Configurable Convolutional Recurrent Neural Network (DC-CRNN) with the Salp Swarm Optimization Algorithm (SSA) to minimize the complexity of the hyperparameter tuning process, achieving improved results on both English and Arabic datasets.

Altwaijry and Al-Turaiki [21] presented the Hijja dataset, which contains 47,434 Arabic characters written by children aged 7 to 12. They trained a CNN architecture on the Hijja and AHCD datasets and reported 97% accuracy on AHCD and 88% on Hijja, showing a promising approach for Arabic handwriting recognition. Kavitha and Srimathi [22] employed CNNs in the offline Handwritten Tamil Character Recognition (HTCR) problem, highlighting the networks' capacity for discriminative feature learning. Their CNN model, trained on a dataset generated by HP Labs India, achieved an accuracy of 95.16%, outperforming baseline methods. Similarly, Assael et al. [23] introduced ITHACA, a deep-learning framework built for automating restoration and chronological accreditation of Greek historical inscriptions. Ithaca improved the restoration performance from 25% to 72%, surpassing state-of-the-art methods.

Shifting towards multilingual and multimodal environments, Das et al. [24] introduced a hybrid deep learning framework combining a transfer-learning-based CNN and a Random Forest classifier for the recognition of Bangla Sign Language (BSL). This model was validated on the Ishara-Bochon and Ishara-Lipi datasets, achieving 91% accuracy for characters and 97% accuracy for digits across both datasets, respectively. Chauhan et al. [25] leveraged transfer learning to develop HCR-Net, a lightweight, script-agnostic model to realize both faster convergence and better generalization. The performance of HCR-Net was compared with 26 baseline methods on 40 public datasets, with up to 11% higher accuracy and 99% first-epoch convergence.

In parallel, optimization-driven techniques have gained

traction for improving feature selection and convergence behavior. Hadadi and Arabani [26] developed a deep learning-based method utilizing various handwritten samples for Parkinson's diagnosis, which was optimized using the Harris Hawks Optimization (HHO) algorithm. This model achieved 94.12% accuracy, outperforming five pre-trained networks, and converged to a cost function value of 0.0084746 in just 10 iterations. Mohammad et al. [27] proposed a metaheuristic method based on the Honey Badger Algorithm (HBA) with the Freeman Chain Code (FCC) for HCR feature extraction, achieving efficient route length and computational time on the CEDAR dataset. However, dependency on the FCC's initial points influences the consistency of extraction. To counter this, the suggested HB-FCC effectively balanced exploration and exploitation dynamically, increasing the robustness of feature extraction.

The baseline studies on capsule, CNN, and hybrid models are summarised in Table 1, along with their datasets, performance levels, and limitations, which served as motivation for the proposed CA-CapsResNet and HCPPO framework.

2.2 Synthesis and comparative analysis

The given related work can be CATEGORIZED into three research areas: (1) CNN-based, (2) Capsule-based, and (3) hybrid or optimization-based approaches. CNN architectures [21, 22, 24, 25] have effectively demonstrated hierarchical learning capabilities and generalizability across scripts; however, their lack of spatial awareness has limited their ability to model detailed relationships among handwriting strokes adequately. Capsule networks [16-18] were motivated to address the issue of spatial awareness by retaining spatial hierarchies and establishing part-whole relationships using dynamic routing. Capsule networks demonstrated a greater representation of local dependencies than CNNs; they were computationally heavy and do not inherently provide the possibility for adaptive decrease in features. Hybrid and optimization approaches [19, 20, 26, 27] aim to balance recognition accuracy and computational effectiveness by integrating deep models with bio-inspired algorithms, such as WOA, SSA, HHO, and HBA. Despite this, the majority of these approaches focus on optimization and feature selection, without considering context-sensitive spatial refinement, a crucial aspect of complex handwriting styles.

2.3 Comparative advantage of the proposed work

Compared to existing CNN, capsule, and hybrid handwriting recognition approaches, our CA-CapsResNet with HCPPO is improved at both the architectural and optimization stages. Conventional CNNs process only local features, resulting in a loss of spatial hierarchy. Capsule networks, on the other hand, maintain spatial relationships but have significant architecture routing complexity and limited adaptability. Our proposed CA-CapsResNet with HCPPO addresses these challenges through residual abstraction, spatial context encoding, and attention-guided refinement, enabling it to model the continuity of curves, stroke dynamics, and contextual dependencies in handwriting more effectively.

Furthermore, unlike previous hybrid or metaheuristic optimizer techniques, such as WOA, SSA, HHO, and HBA, the HCPPO selects compact yet highly discriminative feature subsets by employing adaptive phase transitions to optimize the balance between exploration and exploitation, thereby reducing computational load. In summary, the CA-CapsResNet and HCPPO form an integrated pipeline, resulting in a richer encoding of spatial representations with minimal feature redundancy and improved classification accuracy compared to baseline methods.

Table 1. Summary of the related works done for handwriting character recognition

Study	Model/Technique	Datasets	Novelty	Accuracy	Limitations
Yao et al. [16]	FOD_DCNet	Overlapping digit dataset	Dual dynamic routing for capsule efficiency	93.53%	Limited adaptability, parametric heavy
Parcham et al. [17]	CBCapsNet (CNN + CapsNet)	Signature dataset	Paired-image scheme	92.94% (low FAR/FRR)	Lacks attention-based refinement
Moudgil et al. [18]	CapsNet	Devanagari dataset	Capsule routing across 399 classes	94.6%	Lacks feature optimization
Abd Elaziz et al. [19]	WOA + FCC	Handwritten text	Whale Optimization for FCC representation	Improved path length	No spatial learning, dependent on FCC initialization
Al-Saffar et al. [20]	DC-CRNN + SSA	IAM, IFN/ENIT	SSA for CRNN structure learning	Outperformed conventional CRNN	Limited contextual refinement; high parameter count
Altwaijry and Al-Turaiki [21]	CNN	AHCD, Hijja	CNN trained on a new Arabic dataset	97% (AHCD), 88% (Hijja)	Poor generalization on noisy data
Kavitha and Srimathi [22]	CNN	HP Labs Tamil dataset	CNN trained from scratch	95.16%	No spatial hierarchy modeling; limited robustness
Assael et al. [23]	Ithaca (DL)	Greek inscriptions	Restoration and dating model	62% (restoration)	Limited handwriting variability
Das et al. [24]	CNN + Random Forest	Ishara-Bochon, Ishara-Lipi	Hybrid model with background removal	91% (chars), 97% (digits)	Weak spatial consistency; dual-stage complexity
Chauhan et al. [25]	HCR-Net (transfer learning)	40 datasets	Lightweight, script-independent	+11% vs. baselines	Limited fine-grained spatial encoding
Hadadi and Arabani [26]	DL + HHO	Parkinson's handwriting	HHO-optimized DL model	94.12%	Focused on diagnosis; lacks feature-level optimization
Mohamad et al. [27]	HB-FCC (HBA + FCC)	CEDAR dataset	Honey Badger optimization for FCC features	Improved route length	Sensitive to initial points; limited context-awareness

3. PROPOSED METHODOLOGY

In this section, we briefly cover the proposed methodology in four categories: (1) Image preprocessing, (2) Multi-dimensional handwriting feature extraction, (3) Feature reduction using the proposed PO-CPA approach, and (4) Classification using various machine learning classifiers. Our proposed classification model is illustrated in Figure 1.

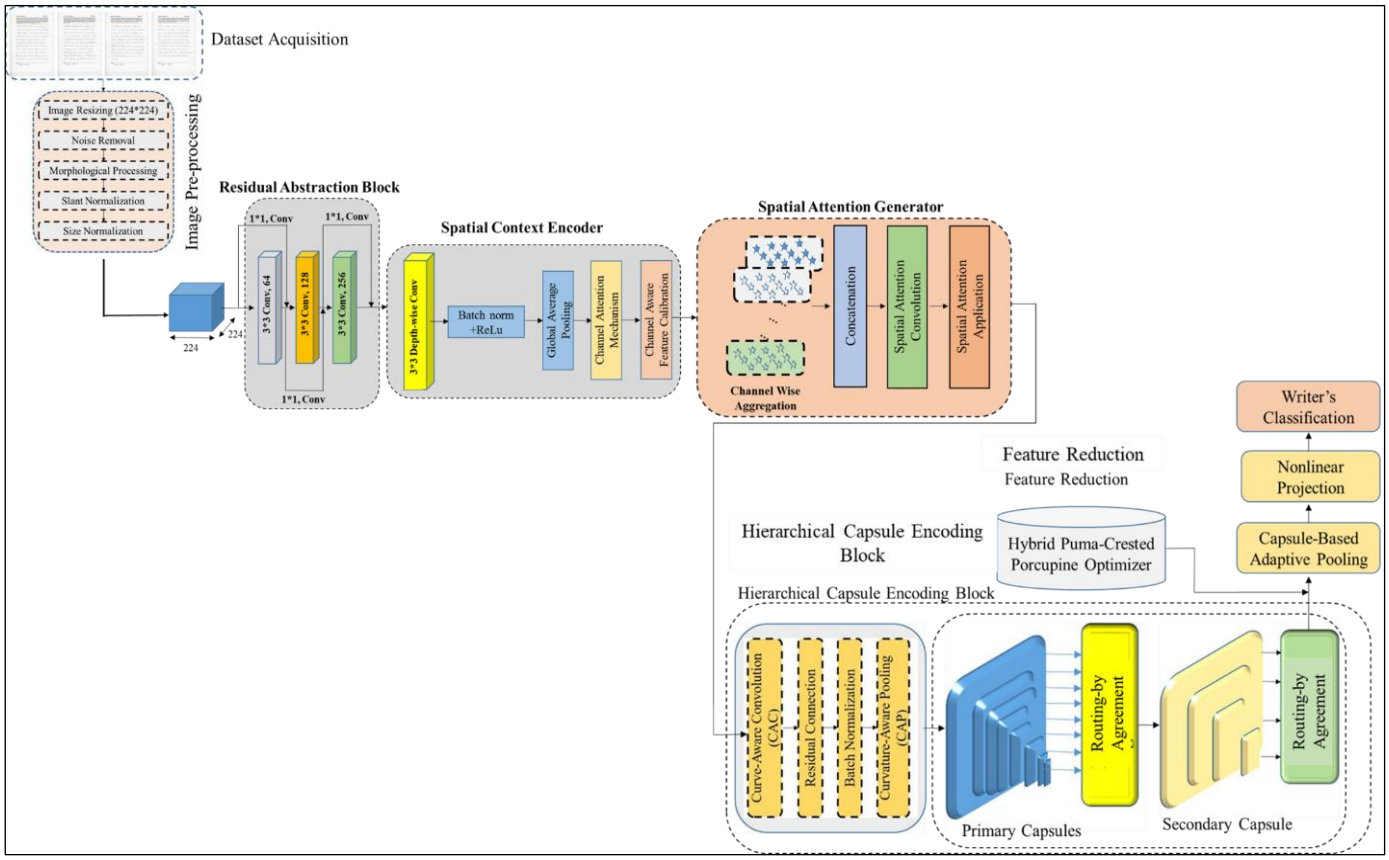

Figure 1. Flowchart of the proposed handwriting recognition and writer's classification framework

3.1 Image preprocessing

Preprocessing is essential for enhancing the quality of handwriting images and preparing them for feature extraction. The operations are carried out step by step, following the mathematical equation.

3.1.1 Image resizing

The input handwriting image is resized to a size of 224 × 224 pixels to maintain uniformity across the dataset. This standardization ensures that all images have the same dimensions for consistent processing. The resized images are then stored for further preprocessing steps, such as noise removal and normalization. The resized sample images are illustrated in Figure 2.

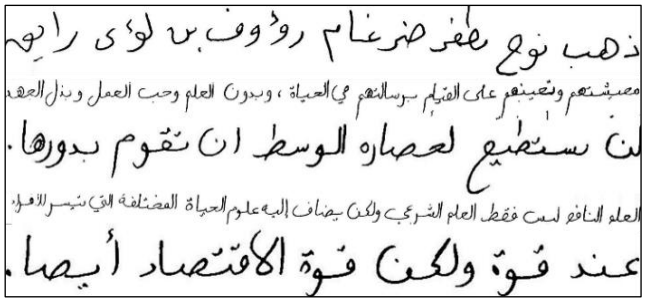

Figure 2. Sample writing samples after performing the resizing operation in 224 × 224 pixels

3.1.2 Noise removal

We apply noise removal methods to remove unwanted artifacts, such as salt-and-pepper noise, providing a clean, binarized image. A median filter is used:

$$I'(x,y) = median\{(x+i, y+j)|(i,j) \in N\} \quad (1)$$

where, N represents the neighborhood window (3 × 3). This step enhances image clarity for further processing. The clean sample images are illustrated in Figure 3.

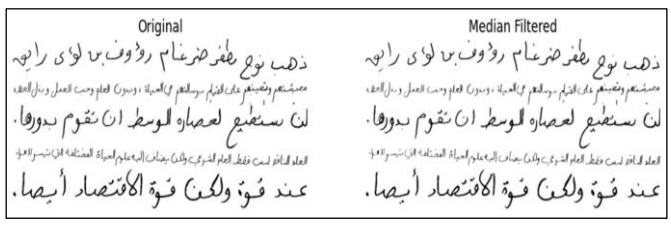

Figure 3. Sample writing samples after performing the denoising operation using median filtering

3.1.3 Morphological processing

We apply morphological operations, including dilation and erosion, to smooth the boundaries of handwriting. Dilation fills small gaps in strokes using Eq. (2):

$$D(I) = I \oplus K \quad (2)$$

where, K is a structuring element. Erosion removes small noises and refines character edges using Eq. (3):

$$E(I) = I \ominus K \tag{3}$$

These tasks strengthen stroke cohesion while maintaining writing characteristics. The processed sample images are illustrated in Figure 4.

Figure 4. Sample writing samples after performing the morphological operation

3.1.4 Slant normalization

Writer-dependent variations of stroke orientation are removed by correction through slant normalization via shear transformation using Eq. (4):

$$\begin{bmatrix} x' \\ y^i \end{bmatrix} = \begin{bmatrix} 1 & K \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4}$$

where, K is the corrected slant factor. It standardizes the handwriting to achieve a uniform appearance across various writers. The processed sample images are illustrated in Figure 5.

Figure 5. Sample writing samples after performing the slant normalization

3.1.5 Size normalization

We apply size normalization to standardize handwriting sizes, ensuring uniform input for recognition models. The rescaling transformation is defined by:

$$I'\left(x', y'\right) = I(x.\alpha, y.\beta) \tag{5}$$

where, α and β are scale factors calculated according to the standard character size. The processed sample images are illustrated in Figure 6.

Figure 6. Sample writing samples after performing the size normalization

3.2 Proposed model architecture

Handwriting recognition for author identification is highly challenging because handwriting style, stroke dynamics, and non-regular spatial features are very difficult to handle due to the presence of intricate intra-personal and inter-personal variability. To address this complexity, we introduce the hybrid CA-CapsResNet, which incorporates curvature-aware adaptive convolutions, residual propagation of features, and capsule-style spatial encoding to enhance the robustness of features while preserving hierarchical structuring for handwriting. The architecture's mathematical formulation is outlined below, with a description of each significant computational stage.

Algorithm 1. Pseudocode for RAB

function Residual Abstraction Block (X):
# Input: $X \in \mathbb{R}^{H \times W \times C}$: original preprocessed image tensor
# Output: $X_0 \in \mathbb{R}^{H \times W \times C}$: shallow feature map
# Step 1: **Apply Zero Padding**
X_ZP = ZeroPad (X, padding=1) # Pad with 1 pixel on all sides
# Step 2: **First Layer**
Conv1 = Conv2D (X_ZP, kernel_size=3x3, stride=1, padding='valid')
Res1 = Conv2D (X_ZP, kernel_size=1x1, stride=1, padding='valid')
Out1 = Conv1 + Res1
# Step 3: **Second Layer (Residual Repeat)**
Conv2 = Conv2D (Out1, kernel_size=3x3, stride=1, padding='same')
Res2 = Conv2D (Out1, kernel_size=1x1, stride=1, padding='same')
Out2 = Conv2 + Res2
# Step 4: **Third Layer (Residual Repeat)**
Conv3 = Conv2D (Out2, kernel_size=3x3, stride=1, padding='same')
Res3 = Conv2D (Out2, kernel_size=1x1, stride=1, padding='same')
Out3 = Conv3 + Res3
# Step 5: **Batch Normalization and ReLU**
BN = Batch Normalization (Out3)
X_0 = ReLU (BN)
return X_0

3.2.1 RAB

Initially, the preprocessed image is transformed into a 3D tensor representation: $X \in R^{H\times W\times C}$, where H, W, and C refer to height, width, and the number of channels, respectively. A convolution layer of size 3 × 3, preceded by a zero-padding operation, is applied to the input tensor to preserve the spatial locality. Further, a residual operation with a convolution window of size 1 × 1 is applied between the original tensor and the convoluted tensor. This step is repeated in the second and third layers. Finally, the output is passed through a Batch Normalization (BN) and ReLU activation layer to compute the shallow feature map: $X_0 \in R^{H\times W\times C}$ can be represented as follows:

$$X_0 = ReLu\left(F_{BN}\left(Res_{1*1}\left(\left(Conv_{3*3}\left(X_{ZP}\right)\right)\otimes X_{ZP}\right)\right)\right) \quad (6)$$

where, X_{ZP} shows zero-padding operation on input tensor X and $F_{BN}(.)$ stands for BN. The pseudocode of this substep is given in Algorithm 1.

3.2.2 SCE

To more finely adjust the spatial representations and enhance global contextual perception, the SCE plays a critical role in the proposed architecture. It aims to capture spatial relationships at various scales without compromising computational efficiency. The input of this block is the feature map obtained from Eq. (1), and a depthwise separable convolution using a 3 × 3 kernel is utilized channel-wise to enable the network to extract subtle local patterns separately within each channel. Mathematically, the depthwise convolution operation can be described as:

$$F_d = DWConv_{3\times 3}\left(X_0\right) \quad (7)$$

where, $DWConv_{3\times 3}\left(X_0\right)$ represents the depthwise convolution operator that is used to minimize the number of hyperparameters while preserving spatial specificity and local structural integrity. Further, BN and ReLU operations are applied. Next, a Global Average Pooling (GAP) is implemented to F_d for incorporating global context into the local descriptors. This step yields a condensed channel-wise descriptor $z \in R^c$ using Eq. (8):

$$z_c = \frac{1}{H^{'}\times W^{'}}\sum_{i=1}^{H^{'}}\sum_{j=1}^{W^{'}} F_d\ (i,j,c), \forall c \in (1,2,\ldots N) \quad (8)$$

This descriptor captures the worldwide spatial distribution of every channel and is an input to a light channel attention module. The attention mechanism is a two-layer fully connected (FC) bottleneck network with a reduction ratio r, ReLU activation, and a final sigmoid gate:

$$\alpha = \sigma(W_2.\delta(W_1 z_c)) \quad (9)$$

where, W_1 and W_2 are learnable weights, $\delta(\cdot)$ is the ReLU activation function, and $\sigma(\cdot)$ is the sigmoid function that outputs the final attention weights $\alpha \in [0,1]^C$. The attention vector α is transmitted over spatial dimensions and used on the depthwise convoluted features $F_{d,}$ resulting in improved context-aware output $F_s' \in R^{H'\times W'\times C}$. This step is mathematically represented in Eq. (10):

$$F_s^{'}(i,j,c) = \alpha_c F_{d,}(i,j,c) \quad (10)$$

This selective boost of informative channels enables the network to amplify semantically relevant spatial features and filter out irrelevant or noisy elements. The pseudocode of this substep is given in Algorithm 2.

Algorithm 2. Pseudocode for SCE

function Spatial Context Encoder (X_0):
Input: $X_0 \in \mathbb{R}^{H\times W\times C}$, feature map from previous block
Output: $F_s' \in \mathbb{R}^{H\times W\times C}$, context-aware spatially refined output
Step 1: **Depthwise Separable Convolution**
$F_d = \mathrm{DWConv}_{3\times 3}(X_0, \mathrm{kernel_{size}} = 3\times 3, \mathrm{stride} = 1, \mathrm{padding} = 'same')$
Step 2: **Batch Normalization and ReLU Activation**
F_d= BatchNormalization (F_d)
F_d= ReLU (F_d)
# Step 3: **Global Average Pooling to obtain channel descriptor** $z \in \mathbb{R}^C$
z = GlobalAveragePooling (F_d)
# Step 4: **Lightweight Channel Attention via 2-layer Fully Connected Network**
hidden_dim = C // r # r is the reduction ratio
F_C1 = Fully Connected (z, out_dim=hidden_dim)
F_C1 = ReLU (F_C1)
F_C2 = FullyConnected (F_C1, out_dim=C)
α = Sigmoid (F_C2)
Step 5: **Contextual Reweighting over Spatial Dimensions**
F_s' = ElementWiseMultiply (F_d, α)
return F_s'

3.2.3 SAG

Although the channel attention mechanism emphasizes the importance of each feature channel uniformly, it lacks spatial discrimination. To address this gap, the SAG is proposed to encode location-aware feature importance, enabling the network to focus more attention on salient areas in the spatial domain.

The input to SAG is the contextually enhanced feature map $F_s' \in R^{H'\times W'\times C}$ derived from the previous encoder module. The spatial attention is built through a channel-wise compression approach and a convolutional spatial gating mechanism.

(1). Channel squeezing aggregation

To calculate spatial saliency, a compound summary of feature information is retrieved by performing average pooling and max pooling operations over the channel dimension:

$$F_{Avg} = AvgPool_{channel}(F_s^{'}) \quad (11)$$

$$F_{Max} = MaxPool_{channel}(F_s^{'}) \quad (12)$$

where, F_{Avg} and $F_{Max} \in R^{H'\times W'\times 1}$. Both maps encode different spatial clues from the global average and global max activations.

(2). Spatial attention convolution

The stacked maps are concatenated along the channel axis and sent through a convolutional attention filter using Eqs. (13) and (14), respectively.

$$F_{Concat} = [F_{Avg}; F_{Max}] \in R^{H' \times W' \times 2} \quad (13)$$

$$\beta = \sigma(Conv_{7\times7}(F_{concat})) \in R^{H' \times W'} \quad (14)$$

Here:

• $Conv_{7\times7}$ (.) is a convolution layer with a 7 × 7 kernel, capturing a wider spatial context.

• (·) is the sigmoid activation function that outputs the spatial attention map β.

(3). Attention mechanism

Finally, the spatial attention map is broadcast to all the channels and element-wise multiplied with the input feature map F_S', which gives rise to the spatially attended output: $F_{SA} = R^{H' \times W' \times C}$ computed according to Eq. (15):

$$F_{SA}(i,j,c) = \beta(i,j) F_{SA}(i,j,c) \quad (15)$$

Spatial reweighting in this manner highlights areas in the feature map that are more informative for recognition and dampens less informative areas, thus enhancing the spatial localization ability of the network.

3.3 Feature extraction using hierarchical capsule encoding block

3.3.1 Curve-aware convolution and curvature-guided pooling

Let $X \in R^{H \times W \times C}$ denote the size of the input image patch, where, H, W, and C refer to height, width, and number of channels, respectively. Initially, we introduced a curvature-awareness-oriented convolution operator by computing the local curvature tensor $\kappa(i,j)$ using the Gaussian curvature margin of the Hessian of the input image (I). It can be mathematically formulated in Eq. (16):

$$\kappa(i,j) = \frac{\partial^2 I}{\partial x^2} * \frac{\partial^2 I}{\partial y^2} - \left(\frac{\partial^2 I}{\partial x \partial y}\right)^2 \quad (16)$$

This curvature estimates scales the convolutional weights in a specially designed curve-aware convolution (CAC) operation. The convolution response at the pixel.

$$F_{CAC}(i,j) = \sum_{m,n,c} \kappa(i+m, j+n) * \kappa_{m,n,c} * X(i+m, j+n, c) \quad (17)$$

where, κ is a learned convolution kernel from Eq. (16).

3.3.2 Normalized residual mapping

To maintain the spatial semantics and enable deep feature learning, the CAC output is added back to the input in a residual setup, followed by BN and activation:

$$F_{Res} = ReLU(\mathcal{B}(F_{CAC}(i,j) + X)) \quad (18)$$

where, $\mathcal{B}(.)$ refers to the BN procedure, and ReLU is the rectified linear unit activation function.

3.3.3 Hierarchical capsule encoding block

(1) Primary capsule equation

The preprocessed image, represented as $X \in R^{224*224*3}$ is passed to a sequence of convolutional operations to derive low-level spatial features. The Hierarchical Dense Capsule Network (HDCN) architecture is designed with four capsule layers with progressively coarser spatial resolution. The topmost capsule layer is a grid with 8 × 8 (64 capsules), which ultimately learns low-level handwriting features, such as stroke curvature and edge flow. The next capsule layer is a 4 × 4 grid (16 capsules) that uses the localized activations from the previous layer to learn mid-level structure and continuity in the patterning of writing strokes. The additional capsule layer is 2 × 2 (4 capsules), which learns features that represent higher-level abstractions of the characters in terms of their subcomponents or strokes (typed formation). The fourth and final capsule layer is a 1 × 1 (1 capsule), representing the style of the user's handwriting by combining all features of the spatially distributed capsules into a small, global representation of the handwritten style.

Each capsule produces a pose vector $u_{i,j}^{1} \in R^{d}$, where (i,j) refers to the location of the individual capsule. The pose vectors are mapped to a higher-level feature space by learning transformation matrices, $W_{p,(i,j)}^{1} \in R^{d' \times d}$ realizing the positional vector (Eq. (12)) for the next layers.

$$\hat{u}_{p|i,j}^{(2)} = \hat{W}_{p|i,j}^{(1)} * u_{i,j}^{(1)} \quad (19)$$

These predictions are routed to the second layer through a dynamic routing-by-agreement process.

$$s_{m,n}^{(l+1)} = \sum_{(i,j)} c_{(i,j),(m,n)}^{(l)} \hat{u}_{(m,n)|(i,j)}^{(l+1)} \quad (20)$$

$$v_{(m,n)}^{(l+1)} = \frac{\| s_{m,n}^{(l+1)} \|^2}{1 + \| s_{m,n}^{(l+1)} \|^2} * \frac{s_{m,n}^{(l+1)}}{\left| \left| s_{m,n}^{(l+1)} \right| \right|} \quad (21)$$

The routing coefficients $c_{(i,j),(m,n)}^{(l)}$ are iteratively refined based on the agreement scores between capsules. This process continues hierarchically through all four capsule layers, resulting in a global identity capsule $v^{(4)}$ representing the writer's distinctive handwriting signature. Furthermore, to refine identity cues, $v^{(4)}$ is fed into a secondary spatial capsule network comprising three capsule layers (4 × 4, 2 × 2, 1 × 1). The input capsule grid is initialized as:

$$u_{(i,j)}^{(1')} = ReLU(A_{i,j} * v^{(4)} + b_{i,j}) \quad (22)$$

where, $A_{i,j} \in R^{d' \times d}$ and $b_{i,j}$ are training parameters. To guide dynamic routing more effectively, attention coefficients $\alpha_{(i,j),p}^{1'}$ are introduced to modulate the contribution of each lower capsule based on learned spatial saliency calculated in Eq. (20).

$$\alpha_{(i,j),p}^{1'} = \frac{\exp(score\left(u_{(i,j)}^{(1')}\right), \hat{W}_{p|(i,j)}^{(1')})}{\sum_{q} \left(u_{(i,j)}^{(1')}\right), \hat{W}_{p|(i,j)}^{(1')})} \quad (23)$$

where,

$$score(u, W) = a^T * tanh(W_a u) \quad (24)$$

The final attention-weighted routing input is computed as:

$$s_p^{(2')} = \alpha_{(i,j),\ p}^{(1')} * c_{(i,j),p}^{(1')} \hat{u}_{p|i,j}^{(2')} \tag{25}$$

This attention-guided capsule routing enhances robustness to local distortions and selectively amplifies salient spatial regions crucial for handwriting identity. The final capsule output $v^{(3')}$ encodes a compact, discriminative embedding that merges local and global characteristics for reliable author identification. The detailed architecture of the proposed CA-CapsResNet is shown in Figure 7.

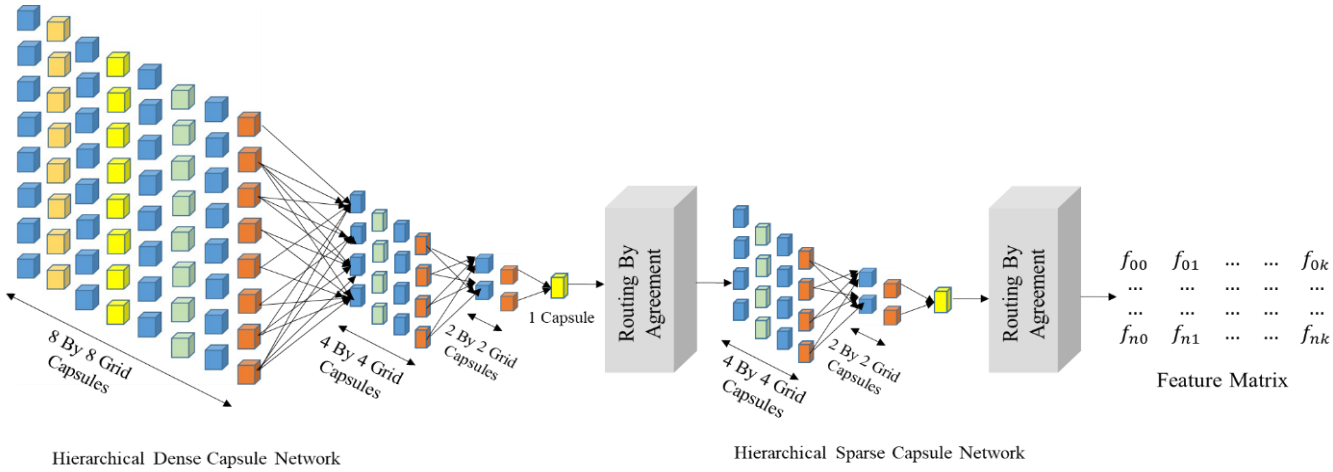

Figure 7. The architecture of the proposed CA-CASResNet

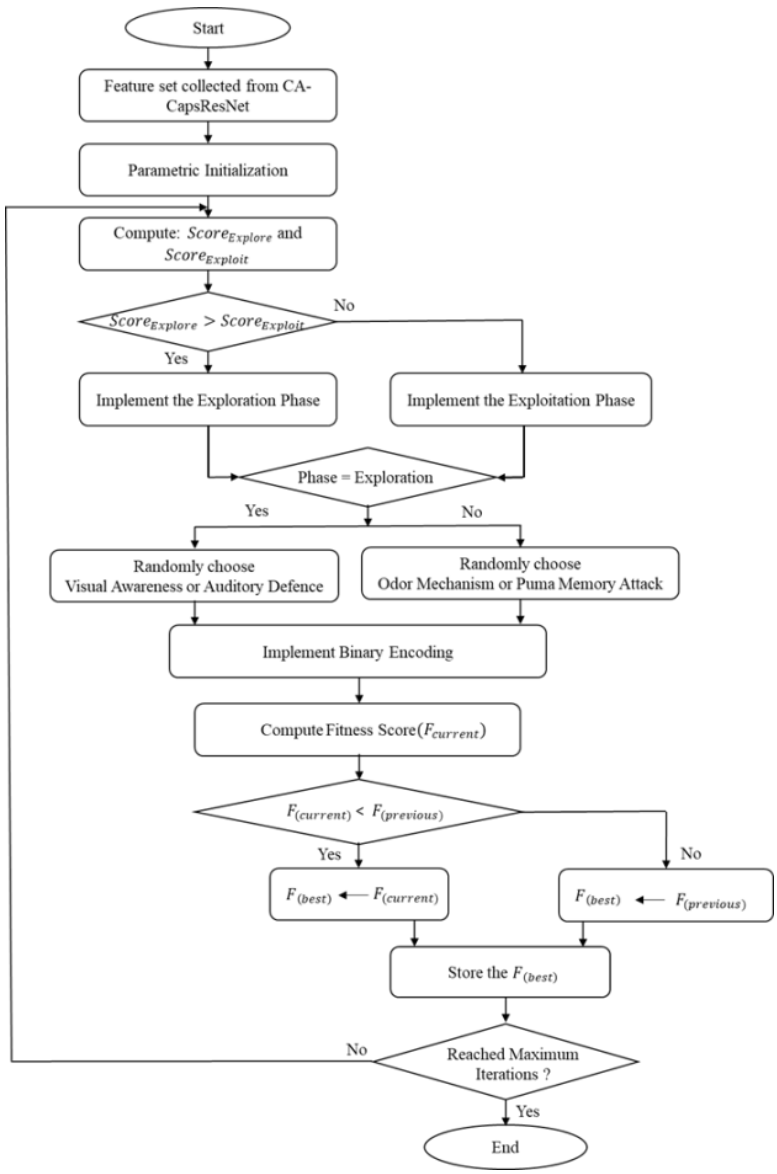

Figure 8. Flowchart of the proposed HCPPO algorithm for feature reduction

3.4 Feature optimization using hybrid puma-crested porcupine optimizer

Feature optimization is crucial for reducing overlapping features while preserving discriminative information. In this paper, we propose a HCPPO to optimize the extracted feature set of Hybrid CA-CapsResNet. This algorithm integrates defence-guided exploration from CPO with intelligence-driven exploitation and adaptive phase transition from PO. The combination of these two strong metaheuristics enables HCPPO to achieve a good balance between exploration and exploitation, thereby determining optimal subsets of features. The flowchart of the proposed algorithm is given in Figure 8. The suggested improvements are discussed in the subsequent steps.

3.4.1 Problem formulation for feature selection

Let $D = \{(x_1, y_1),\ (x_2, y_2), \ldots (x_n, y_n)\}$ be a dataset consisting of n samples and m features where $x_i \in \mathbb{R}^m$ and $y_i \in R$. The objective is to find the best subset of features $S \subseteq (f_1, f_2, \ldots f_n)$ that achieves maximum model performance while minimizing the number of selected features. Define a binary solution vector:

$$X = \{x_1, x_2, \ldots x_n\}, x_i \in 0,1 \tag{26}$$

where, $x_i = 0$ if the feature is rejected and $x_i = 1$ if the feature is selected. The objective function $(f(x))$ for feature optimization is modelled as a weighted sum of classification accuracy error and the relative number of features selected. Symbolically, it can be represented in Eq. (42).

$$f(x) = \alpha.Class_{Error} + (1-\alpha).\frac{No.\ of\ selected\ features}{Original\ features} \tag{27}$$

where, $Class_{Error}$ refers to errors in classification accuracy, and α is a weight balancing the tradeoff between classification accuracy and selected features.

3.4.2 Initialization phase

The algorithm begins by generating a population of N binary agents $X_i \in \{0, 1\}^m$, representing different subsets of features. Let the features generated by CA-CASResNet be defined by

$$X_i^0 = X_1^0,\ X_2^0, \ldots, X_m^0 \tag{28}$$

The fitness score of each feature is calculated by $f(X_i^0)$.

3.4.3 Intelligent phase switching mechanism

The PO-inspired adaptive phase transition adaptively chooses between exploration and exploitation phases based on scoring mechanisms, as outlined in Eqs. (22) and (23), respectively.

$$Score_{Explore} = \lambda_1 f_1^{explore} + \lambda_2 f_2^{explore} \tag{29}$$

$$Score_{Exploit} = \lambda_1 f_1^{exploit} + \lambda_2 f_2^{exploit} \tag{30}$$

$$f_2 = mean\ (\Delta Fitness_{last\ k}) \tag{31}$$

The phase is selected as follows:

$$Phase = \begin{cases} Exploration\ \ if\ Score_{Explore} > Score_{Exploit} \\ Exploitation \qquad\qquad Otherwise \end{cases} \tag{32}$$

Algorithm 3. Pseudocode of proposed HCPPO for feature reduction

Input:
- Extracted feature set
$F = \{(x_1, y_1),\ (x_2, y_2), \ldots (x_n, y_n)\}$ from CA-CapsResNet
- Population size N_{max}, minimum population size N_{min}
- Maximum iterations T_{max}
- Objective function
$f(x) = \alpha.\ Class_{Error} + (1 - \alpha).\frac{\text{No. of selected features}}{\text{Original features}}$
- Convergence parameters: γ (convergence rate), T_f (tradeoff factor)
- Random coefficients r_1, r_2, r_3
- $\lambda 1$, $\lambda 2$ (phase weight factors)

Output:
- Optimized feature subset X_{best}

For t = 1 to T_{max} **do**:

Step 1: Initialization
1.1 Initialize population agent: $X = \{X_1, X_2, \ldots, X_{N_{max}}\}$ randomly
1.2 Evaluate fitness for each candidate solution using f_{max}
1.3. Calculate $Score_{Explore}$ and $Score_{Exploit}$ using Eq. (22), respectively.

Step 2: Phase Selection Process
Select phase based on Eq. (25):
If $Score_{Explore} > Score_{Explore}$ then
Phase ← Exploration
Else
Phase ← Exploitation

Step 3: Decision Step 1
If Phase == Exploration then
Randomly select one of the following phases
- Visual Awareness (Eq. (26)):
- Auditory Defense (Eq. (27)):

Step 4: Decision Step 2
Else if Phase == Exploitation then:
Randomly select one of the following phases
- Odor Mechanism (Eq. (28)):
- Puma Memory Attack (Eq. (29)):

Step 5: Apply Binary Encoding using Eq. (30):
Step 6: Evaluate fitness f (X_i)
If $f(X_i) < f(X_{best})$, then update
$X_{best} \leftarrow X_i$
Else
Do nothing
Step 7: End For
Step 8: Return X_{best}
End

3.4.4 Exploration phase

The CPO model utilizes defensive strategies to diversify the search using two substeps: (1) Visual Awareness and (2) Auditory Defense. The working of both substeps is formulated in Eqs. (33)-(37).

(1) Visual awareness

The visual awareness step in the HCPO algorithm is analogous to the sight defense mechanism of the Crested Porcupine. It facilitates extensive exploratory search through the simulation of far-away threat detection, wherein agents execute random Gaussian walks over the solution space to find new, unexplored areas. This step is formulated in Eq. (33):

$$X_i^{t+1} = X_i^t + \gamma * \mathcal{N}(0,1) \tag{33}$$

where, γ controls the degree of perturbation, and $\mathcal{N}(0,1)$ introduces random Gaussian noise.

(2) Auditory defense

Agents create echo signals based on fitness, affecting others through:

$$X_i^{t+1} = X_i^t + \delta(rand.(X_{best} - X_i^t)) \tag{34}$$

3.4.5 Exploitation phase

At the exploitation phase, intelligent puma strategies are utilized for local intensification:

A. Odor (CPO) mechanism:

$$X_i^{t+1} = X_i^t + \beta.sign\left(X_{best} - X_i^t\right).|(X_{best} - X_i^t)|^n \tag{35}$$

where, $\beta \in (0,1)$ and η controls the learning rate.

B. Puma memory attack: The agent selects the best historical region using the criteria discussed in Eq. (36):

$$X_i^{t+1} = mean\,(X_{best},\ X_{history-best},\ X_i^t) \tag{36}$$

Further, X_i^{t+1} is binary-encoded using Eq. (37):

$$X_i^{t+1}(j) = \begin{cases} 1 & if\ \sigma\left(X_i^{t+1}(j)\right) > r \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

where, $\sigma(.)$ is a sigmoid function. The pseudocode of the proposed HCPPO is given in Algorithm 3.

3.5 Non-linear projection and classification

After the hierarchical capsule encoding and subsequent feature reduction through the HCPPO, the selected feature set $\hat{F}_{Sel} \in \mathbb{R}^{N \times d}$, where, N is the number of samples and d is the optimized feature dimension. In order to increase nonlinearity and enhance the separability of classes in the acquired manifold, we introduce a nonlinear projection module that maps these projected features to a latent discriminative subspace $R \in N^{N \times d'}$ where $d' < d$. Mathematically, we can formulate this step according to Eq. (38):

$$F_{Proj} = Tanh(W_2.ReLU\left(W_1.\hat{F}_{Sel} + b_1\right) + b_2) \tag{38}$$

where, W_1 and W_2 are learnable weight matrices, b_1 and b_2 are bias terms. The projected features are then normalized and passed to a Softmax classifier for final prediction, as shown in Eq. (39):

$$\hat{y} = Softmax\,(W_c.F_{norm} + b_c) \tag{39}$$

where, W_c and b_c are learning parameters of the output layer. The output vector $\hat{y} \in R^{N \times C}$ holds the class probabilities for all samples.

4. EXPERIMENTAL RESULTS AND DISCUSSION

This section provides an overall assessment of the proposed feature extraction using the Capsule Encoding Block and feature optimization using HCPPO from multiple dimensions. Sections 4.1 and 4.2 describe the experimental setup, including the datasets used and the model's implementation details. In Sections 4.3 and 4.4, we compare the performance of the model by evaluating its classification accuracy and parameter efficiency against several state-of-the-art approaches on various image datasets. Additionally, Section 4.5 presents a detailed ablation study to examine the contribution of different architectural components and configurations to the overall performance of the proposed framework.

4.1 Experimental setup

All experiments were conducted on a high-performance workstation equipped with an Intel Core i9-12900K processor at 3.20 GHz and an NVIDIA GeForce RTX 3090 graphics card. Model training and development were performed using PyTorch 1.13.1, a deep learning library that leverages CUDA 11.7 and cuDNN 8.5 to take advantage of GPU acceleration.

4.2 Datasets details

Two benchmark handwriting datasets, the Devnagari Character Dataset and the KHATT dataset, were used in experiments to validate the robustness and generalization ability of the proposed handwriting recognition framework. They were chosen owing to their diversity in script style, linguistic content, and writing complexity. A crisp detail of both datasets is given below:

(1). Devanagari Hindi MNIST dataset

The Devnagari Hindi Character Dataset (DHCD) [28, 29] comprises 92000 scans [32 × 32 pixels] stored in 8-bit grayscale format. It is uniformly distributed over 46 balanced classes, comprising 36-character classes (vowels and consonants) and 10 numeral classes. Each class includes approximately 2,000 samples, ensuring equitable representation and preventing class imbalance during model training and evaluation.

The data set has significant intra-class variability, including variations in handwriting style, skew, and character connectivity, making it an appropriate challenge for testing the spatial sensitivity of our model. The sample images are attached in Figure 9.

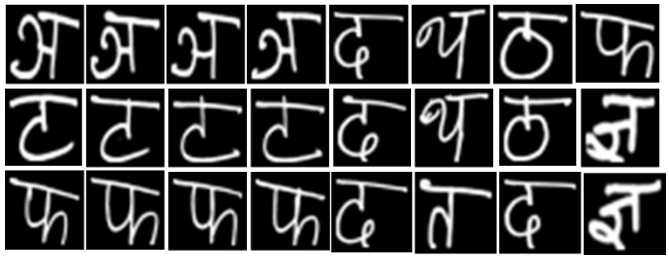

Figure 9. Sample images of the Devanagari Hindi MNIST dataset

(2). KHATT dataset

The KFUPM Handwritten Arabic Text (KHATT) [30] Database is one of the largest databases of unconstrained handwritten Arabic text, comprising 9,850 handwritten forms contributed by 1,000 diverse authors. The dataset comprises 2,000 similar-text paragraph images and 2,000 unique-text paragraph images, uniformly distributed across all 1,000 classes. Each page was scanned at a resolution of 300 dpi and saved as grayscale TIFF files. All the images are followed by manually authenticated ground truth annotations, both in Arabic text and Latin transliteration. The sample images are attached in Figure 10.

4.3 Architectural composition and parametric overview

The proposed handwriting recognition architecture relies on CA-CapsResNet for discriminative and robust feature extraction and HCPPO for effective feature reduction and dimensionality compression. The CA-CapsResNet is designed at the architectural level to encode fine-grained handwriting features by synergistically combining curve-aware adaptive convolutions, residual propagation, and capsule-like spatial encoding, thereby retaining the handwriting's inherent spatial hierarchy and direction variances. Coupling this, HCPPO leverages the search-oriented advantages of Crested Porcupine motion and the Puma optimizer's behavior of exploitation to eliminate redundant features while retaining critical descriptors. Tables 2-4 provide in-depth parametric descriptions of the constituent blocks in the architecture, focusing on kernel sizes, layer-wise compositions, activation modalities, and optimization methods.

Figure 10. Sample images of the KHATT dataset

Table 2. Components-wise parametric details used in the CA-CapsResNet model

Module	Key Components	Kernel Size	Parameters Counts	Activation	Normalization
RAB	3 × 3 Conv, 1 × 1 Residual Conv (× 3 layers), ReLU, BN	3 × 3, 1 × 1	~35 K	ReLU	BatchNorm
SCE	Depthwise Separable Conv, GAP, FC × 2 (Attention), ReLU, Sigmoid	3 × 3 DW Conv, FC	~28 K	ReLU	BatchNorm
SAG	Avg & Max Pooling (channel-wise), 7 × 7 Conv, Sigmoid	7 × 7 Conv	~10 K	Sigmoid	-

Table 3. Training configuration and hyperparameter setting of the CA-CapsResNet model

Training Parameter	Value/ Setting
Optimizer	AdamW (Weight Decoupling)
Learning Rate	0.001
Learning Rate Schedule	Cosine Annealing with Warm Restarts (T_0 = 10 epochs, $\eta_{min} = 1e^{-5}$)
Number of Epochs	150
Batch Size	64
Loss Function	Categorical Cross-Entropy with Focal Weighting ($\gamma = 2.0$)
Regularization	DropBlock ($p = 0.1$), L2 weight decay ($\lambda = 1e^{-4}$)
Early Stopping	Patience = 20 epochs based on validation loss
Validation Split	Five-fold (80:20)

4.4 Performance on Devnagari character and KHATT dataset

In this section, classification accuracy and the number of trainable parameters is used as the primary metrics for model evaluation (Table 4). Since the samples in the Devnagari Character Dataset and KHATT datasets are 28 × 28 grayscale images of a single channel, the parameter count is the same for all models. The performance of the proposed framework is compared with six non-capsule network-based handwriting recognition frameworks: (1) R-CSNN [30], (2) WaveMix [31], (3) Threshold-Gabor CNN [32], (4) Fast Keypoints with Harris Corner Detection (FKHCD) [33], (5) Hybrid CNN with SVM [34], (6) Deep Transfer Learning with Random Forest (DTL-RF) [24], and six capsule network-based methods: (1) CapsNet [35], (2) Kernalized Deep Capsule Networks (K-DCN) [16], (3) Deep Hybrid Capsule Networks (DHCN) [36], (4) Dense Capsule Networks (DCN) [37], (5) Deep Multi-prototype Capsule Networks [38], and (6) Modified Part Capsule Auto-encoder (MPCAE) [39]. The experimental outcomes demonstrate a strong performance benefit of the proposed technique over both standard and capsule-based methods on the Devnagari and KHATT datasets (Table 5).

In the case of methods not based on capsules, including R-CSNN, WaveMix, Threshold-Gabor CNN, FKHCD, Hybrid CNN with SVM, and DTL-RF, the accuracy increased by 15–21% in all instances on both datasets when feature reduction was applied. These models, typically characterized by a relatively shallow depth and linear backends for classification, appear to be more susceptible to noise and redundancy in the feature space. For instance, FKHCD increased from 58.75% to 74.40% on the Devnagari dataset, an increase of more than 21%, the best in this set.

Capsule-based models exhibit more complex behavior.

Although they typically begin with higher baseline accuracy due to their built-in routing-by-agreement, they still experience improvements in the range of 11–18% when features are reduced. CapsNet, for example, increases from 70.86% to 86.44% on Devnagari with feature reduction. Interestingly, state-of-the-art capsule networks, such as MPCAE and Deep Multi-Prototype CapsNet, exhibit relatively smaller gains, suggesting that these models already incorporate some implicit feature selection or compression during training. The proposed method consistently performs the best on both datasets, achieving 98.94% and 97.36% accuracy on the Devnagari and KHATT datasets, respectively, when feature reduction is applied. Even without feature reduction, it retains a high performance, beating all other approaches with a substantially reduced parameter number. The reduction in performance in the absence of feature reduction is nominal, at 5.7% and 5.6%, respectively. It proves the inherent strength of the architecture and the reduced reliance on extrinsic preprocessing stages.

Traditional non-capsule networks have a lower parameter count, ranging from 0.7 million to 1.1 million. While they are easy to implement, the models lack the performance of more complex capsule-based methods, especially without feature reduction. Capsule networks, although they offer better performance, are computationally more costly, with parameters ranging from 6.5 million to over 8.2 million. Surprisingly, our proposed approach deviates from this trend by achieving greater accuracy with only 2.87 M parameters, which is significantly lower than those in any capsule-based model.

Overall, we can summarize the performance of our method on both datasets in the following points:

• Key Findings on Devnagri Datasets

(1) The proposed framework achieved 98.94% accuracy with feature reduction and 93.28% without it, marking a 5.7% improvement.

(2) Outperformed all other non-capsule (R-CSNN, WaveMix, FKHCD, etc.) and capsule-based methods (CapsNet, MPCAE, Deep Multi-Prototype CapsNet).

(3) CapsNet improved from 70.86% to 86.44% (18% gain), yet remained significantly below the proposed model.

(4) Traditional non-capsule networks showed 15–21% improvement with feature reduction but achieved notably lower absolute accuracies.

• Key Findings on KHATT Datasets

(1) The proposed framework achieved 97.36% accuracy with feature reduction and 91.91% without it, representing a 5.45% improvement in accuracy.

(2) Consistently outperformed all competing models on the KHATT dataset, maintaining robustness across language scripts.

(3) CapsNet accuracy increased from 68.02% to 83.82% (18.9% gain), while Deep Multi-Prototype CapsNet rose from 75.26% to 88.04% (14.5% gain), but lower than the proposed model.

(4) Non-capsule networks exhibited similar 15–21% relative improvements, but still lower than in the final accuracy levels.

Table 4. Details of optimal hyperparameters used in the feature reduction step using Puma-CPO

Hyperparameter	Symbol	Optimal Value	Description
Population Size (Initial)	N_{max}	Dataset dependent	The initial number of candidate solutions
Minimum Population Size	N_{min}	Dataset dependent	Lower bound for dynamic population reduction
Maximum Iterations	T_{max}	500	Stopping criterion for optimization
Convergence Rate	γ	0.13	Controls the balance between global and local search
Tradeoff Factor	T_f	0.87	Probability of selecting the third or fourth defense mechanisms in CPO
Exploration Factor	r_1	0.19	Adjusts the impact of global best in the Puma search step
Exploitation Factor	r_2	0.81	Controls the refinement of feature selection
Perturbation Strength	S	0.03	Stochastic perturbation for local search in CPO
Adaptive Memory Weight	U_1	0.7	Weight factor for personal best solution updates
Adaptive Step Size	Y_t	0.17	Determines search step variation in CPO
Selection Weight	U_2	0.8	Influences decision-making in feature selection
Objective Function Weight 1	α_1	0.37	Weight for classification accuracy error
Objective Function Weight 2	α_2	0.63	Weight for the number of selected features

Table 5. Accuracy and number of parameters across various methods on Devnagri and KHATT datasets with and without feature reduction

Method	Devnagari Accuracy (%)		KHATT Accuracy (%)		Parameters
	With FR	Without FR	With FR	Without FR	
Non-Capsule Methods					
R-CSNN	78.15	65.42 (-16.4%)	74.80	63.00 (-15.8%)	~0.8M
WaveMix	82.60	68.42 (-17.2%)	80.50	67.44 (-16.3%)	0.7M
Threshold-Gabor CNN	75.88	62.14 (-18.2%)	72.95	61.30 (-16.0%)	~1.0M
FKHCD	74.40	58.75 (-21.0%)	71.23	59.12 (-17.0%)	~0.9M
Hybrid CNN + SVM	81.76	67.33 (-17.6%)	79.01	65.40 (-17.2%)	~0.85M
DTL-RF	85.33	70.61 (-17.2%)	82.19	69.40 (-15.6%)	~1.1M
Capsule-Based Methods					
CapsNet	86.44	70.86 (-18.0%)	83.82	68.02 (-18.9%)	8.2M
K-DCN	87.55	73.06 (-16.6%)	84.90	70.88 (-16.5%)	7.9M
DHCN	88.41	74.02 (-16.2%)	85.75	72.48 (-15.5%)	6.5M
DCN	89.67	75.38 (-15.9%)	86.91	73.63 (-15.3%)	7.2M
Deep Multi-Prototype CapsNet	90.38	78.03 (-13.6%)	88.04	75.26 (-14.5%)	7.6M
MPCAE	91.74	81.03 (-11.7%)	89.66	77.74 (-13.3%)	6.9M
Proposed Method (Ours)	98.94	93.28 (-5.7%)	97.36	91.91 (-5.6%)	2.87M

4.5 Effect of feature selection on classification accuracy

Figure 11 shows the comparison of validation accuracy between various baseline methods and the proposed approach on the Devanagari dataset after applying feature selection. It is clear that the feature selection significantly improves classification performance for all the methods. The proposed algorithm consistently outperforms all baselines, achieving more than 80% validation accuracy in the initial 50 epochs, with performance peaking at approximately 99.5% after 145 epochs. By comparison, the nearest competing approaches, MPCaE and Deep Multi-prototype CapsNet, achieve their highest validation accuracy of approximately 91% and 89%, respectively, which converge at slower rates. Baseline strategies without feature fine-tuning, such as R-CSNN and Hybrid CNN + SVM, plateau at much lower accuracy ranges, from 70% to 85%. The improved performance of the introduced method demonstrates how selecting the most informative features optimizes learning effectiveness, reduces overfitting, and accelerates convergence speed.

Figure 12 illustrates the loss for various baselines and the proposed method on the Devanagari dataset. The proposed method exhibits the lowest and most stable validation loss during training, and it rapidly dips below 0.2 at around 70 epochs, while maintaining improved convergence. Baseline methods converge at a higher loss value of about 0.3 to 0.5, indicating inferior learning. The sharp decline of the proposed approach's loss curve also affirms the strength of feature selection in enabling faster and robust optimization.

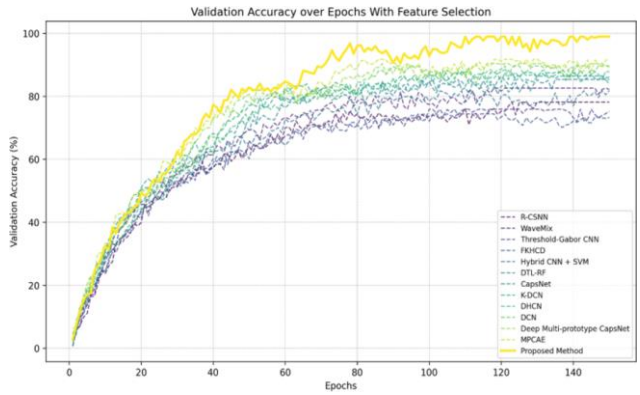

Figure 11. Epoch-wise validation accuracy comparison between various baseline methods and our approach on the Devanagari dataset with feature reduction

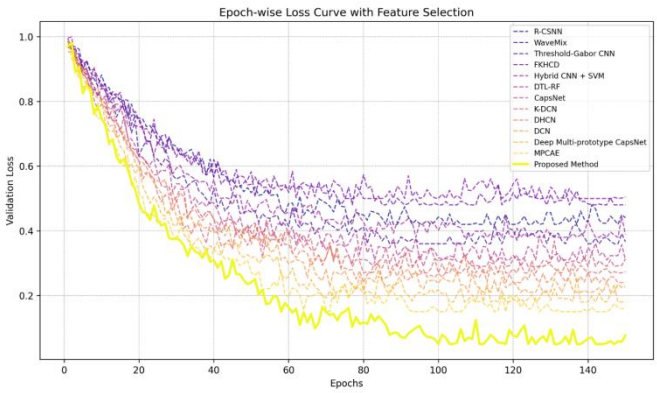

Figure 12. Epoch-wise loss comparison between various baseline methods and our approach on the Devanagari dataset with feature reduction

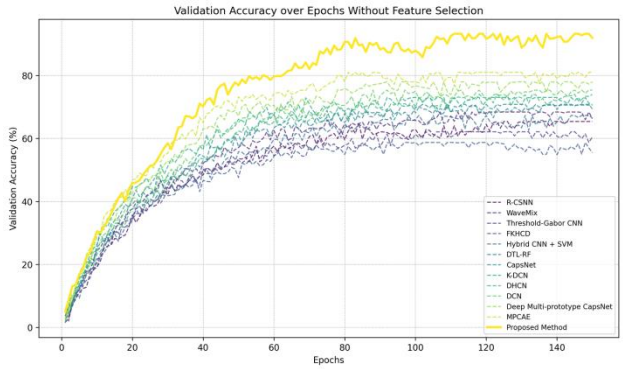

Figure 13. Epoch-wise validation accuracy comparison between various baseline methods and our approach on the Devanagari dataset without feature reduction

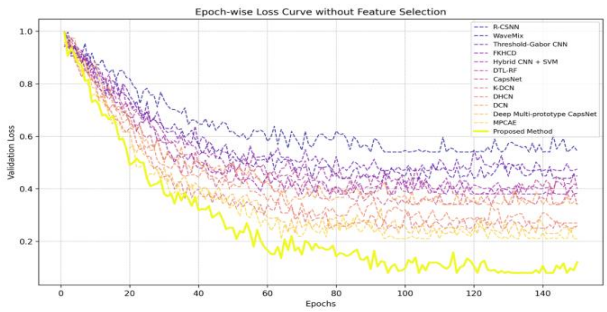

Figure 14. Epoch-wise loss comparison between various baseline methods and our approach on the Devanagari dataset without feature reduction

Figure 13 presents the epoch-by-epoch comparison of validation accuracies for the proposed method and various baselines on the Devanagari dataset, without feature reduction. While the proposed method still far surpasses every other competing methodology, its validation accuracy peak slides down to near 88% compared to ~99.5% when a feature selection criterion is used (Figure 11).

Other alternatives, such as MPCaE and deep multi-prototype CapsNet, boast maximum accuracies of around 82% and 78%, respectively. By contrast, conventional baselines such as R-CSNN and Threshold-Gabor CNN tend to plateau at lower accuracy levels, ranging from 65% to 70%.

Figure 14 illustrates the epoch-wise comparison of validation loss for different baseline models and the proposed approach on the Devanagari dataset, without feature reduction. Even though the suggested approach still has the lowest validation loss among all models, its convergence is slightly less sharp and smooth than when feature selection is used (as shown in Figure 12). The proposed approach converges to a loss of ~0.08, whereas other competing approaches, such as MPCaE and Deep Multi-prototype CapsNet, converge towards losses of ~0.15 and 0.18, respectively. The remaining baseline approaches, such as R-CSNN and Threshold-Gabor CNN, converge at much higher loss values of approximately 0.3–0.45, reflecting poorer generalization performance.

Figure 15 shows the epoch-wise comparison of validation accuracy among different baseline techniques and the suggested technique on the KHATT dataset with feature reduction. The suggested technique outperforms all baselines throughout the epochs, achieving a validation accuracy of around 96–97%, which indicates a prominent margin of 8–12% over the second-best rival, such as MPCaE and Deep

Multi-prototype CapsNet. Other approaches, such as DTL-RF and Hybrid CNN+SVM, plateau at lower accuracies of 80–85%, while traditional baselines, including R-CSNN and Threshold-Gabor CNN, stabilize at accuracies below 75%. The faster and higher convergence of the proposed approach demonstrates the strength of feature selection in retaining the most discriminative patterns and eliminating redundant information, resulting in improved generalization ability and training stability.

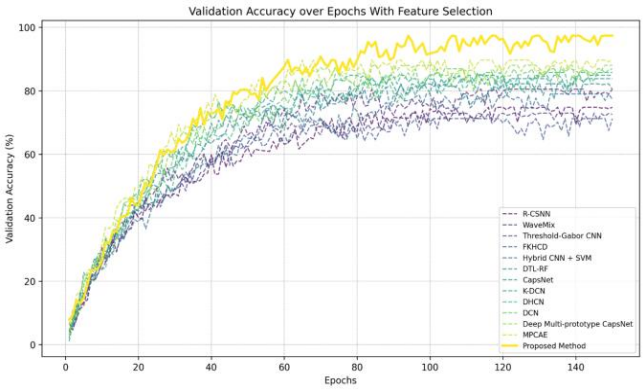

Figure 15. Epoch-wise validation accuracy comparison between various baseline methods and our approach on the KHATT dataset with feature reduction

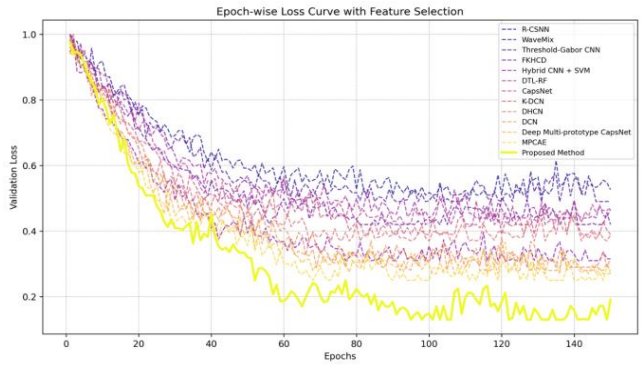

Figure 16. Epoch-wise loss comparison between various baseline methods and our approach on the KHATT dataset with feature reduction

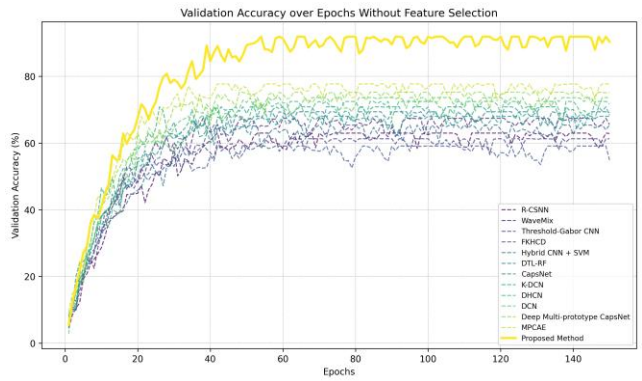

Figure 17. Epoch-wise validation accuracy comparison between various baseline methods and our approach on the KHATT dataset without feature reduction

Figure 16 depicts the validation loss curves for the same experimental configuration on the KHATT dataset with feature reduction. Not only does the proposed method yield the lowest validation loss, which stabilizes at 0.10, but it also has a smoother and quicker convergence than all other models. On the other hand, baselines such as MPCaE and Deep Multi-prototype CapsNet remain at higher losses of 0.20–0.25, whereas conventional approaches, including R-CSNN and Threshold-Gabor CNN, remain above 0.4. The steeper drop and lower final loss values of the proposed approach validate its better learning dynamics, effective optimization, and low overfitting behavior.

In Figure 17, the accuracy progression with feature selection for validation clearly shows the better learning ability of the proposed method compared to baseline models. The proposed method exhibits a steep increase in accuracy during the first 40 epochs, reaching above 80% early on, and gradually increases to approximately 95–97% as training continues. By contrast, baseline approaches like R-CSNN, WaveMix, and Threshold-Gabor CNN have slower convergence and plateau at much lower accuracies (~70–85%). Feature reduction has significantly enhanced the discriminative power of the input representations, leading to faster convergence and more stable performance.

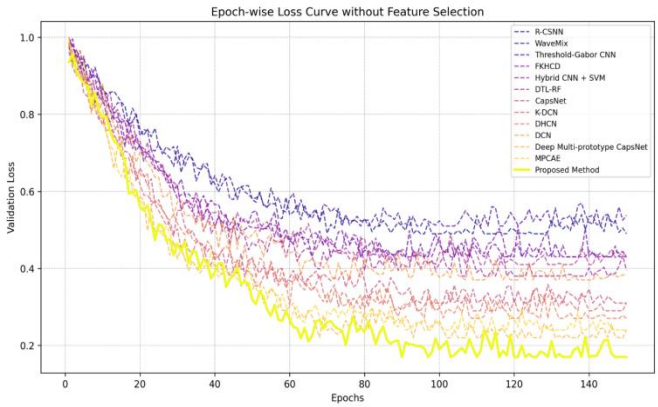

Figure 18. Epoch-wise validation accuracy comparison between various baseline methods and our approach on the KHATT dataset without feature reduction

Figure 18 shows the respective validation loss curves, where the proposed approach records the steepest and most convergent drop in loss. At around 60 epochs, the proposed approach's validation loss is already less than 0.2 and remains low throughout with no significant oscillations for the remainder of training. All other approaches plateau at much higher levels of loss (~0.3–0.5), recording slower optimization and less convergent stability. The gradual and smooth decline in the loss curve for the proposed approach suggests efficient training, quality fit to the data, and no overfitting.

Figure 19 presents a comprehensive comparison of precision, recall, and F1-score between different baseline models and the proposed method, with and without feature selection, on the Devanagari dataset. On all metrics, it is evident that feature selection consistently improves performance across different models. Interestingly, the proposed method (ours) produced the best performance, with a precision of 0.91, a recall of 0.91, and an F1-score of 0.91 when feature selection was used, compared to 0.81, 0.82, and 0.81, respectively, without feature selection. Similarly, conventional techniques such as Thresholder-Gabor CNN and Hybrid CNN + SVM exhibit significant improvements following feature selection, with precision values rising from

0.61 to 0.71 and from 0.63 to 0.71, respectively. In addition, the baselines of deep learning models, such as DTCN and DCN, also improved, with the F1-score of DTCN increasing from 0.72 to 0.83 and that of DCN from 0.74 to 0.85.

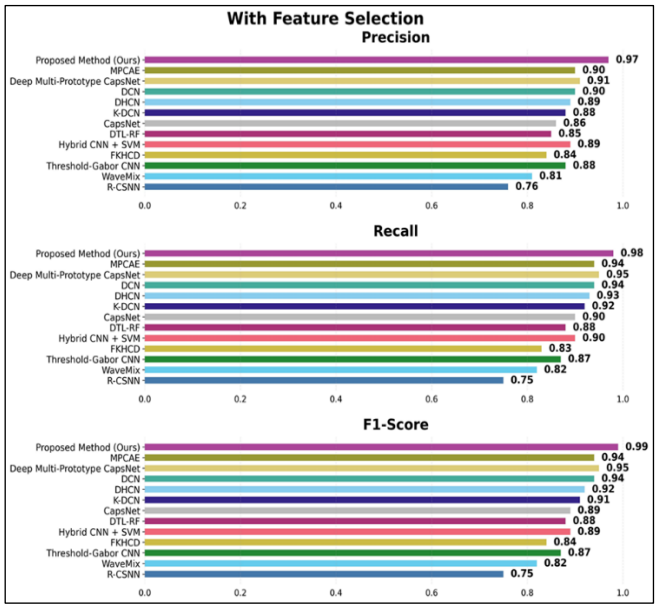

Figure 19. Epoch-wise validation accuracy comparison between various baseline methods and our approach on the Devanagari dataset without feature reduction

Figure 20 compares the precision, recall, and F1-score of some baseline models and the proposed method, tested on the KHATT dataset under both feature selection and non-feature selection conditions. There is a common trend here where feature selection greatly improves the performance of the models in all measures. The proposed approach achieves better performance, with accuracy, recall, and F1-scores of 0.90, 0.90, and 0.90, respectively, when feature selection is used, compared to 0.79, 0.80, and 0.79, respectively, without feature selection. Baseline approaches, such as Thresholded-Gabor CNN and Hybrid CNN + SVM, yield moderate gains; for example, precision values increase from 0.53 to 0.72 and from 0.53 to 0.73, respectively, when feature selection is employed. More complex models, such as DTCN and DCN, also benefited significantly, with the F1-score of DTCN increasing from 0.71 to 0.81 and that of DCN from 0.74 to 0.84 through feature selection. Likewise, MPCFE and Deep Multi-Stage Cascaded Networks showed considerable improvement, achieving F1-scores of 0.86 and 0.85, respectively, which reflects the success of feature selection in deep models.

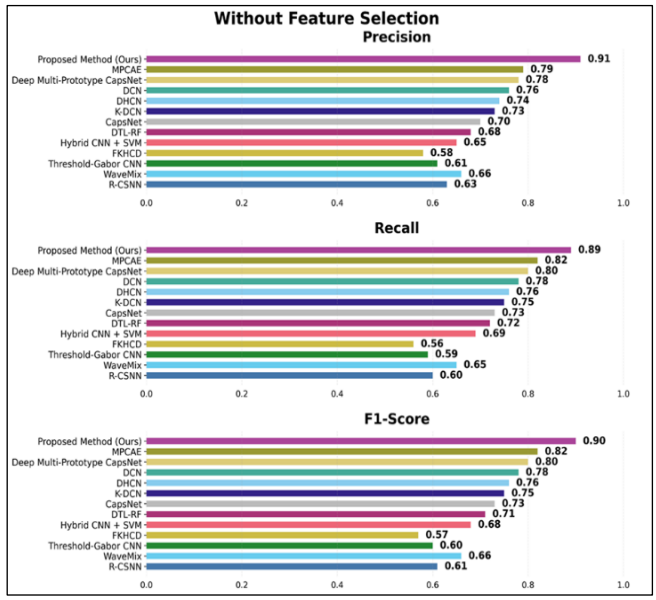

Figure 20. Epoch-wise validation accuracy comparison between various baseline methods and our approach on the KHATT dataset without feature reduction

4.6 Ablation study

In an ablation study, we need to determine how our proposed components contribute to the verification model by comparing the model with each component included to the model without it. Thus, to conduct the ablation analysis, we employed five varying models by excluding each of the suggested components from the original CA-CapsResNet architecture. These models were constructed as follows:

- Model 1: The original proposed CA-CapsResNet without the RAB
- Model 2: The original proposed CA-CapsResNet without the SCE module (Removing spatial attention modeling)
- Model 3: The proposed architecture without the Channel Attention sub-module inside SCE (Only spatial attention is preserved)
- Model 4: The proposed CA-CapsResNet without Capsule Encoding (Only curvature-aware CNN and SCE without the capsule layer)
- Model 5: The original proposed CA-CapsResNet without the Curvature-Aware Convolutions (Using standard convolutions instead).

Table 6. Comparison of the proposed method with the five models used in the ablation study

Model	Devnagari Accuracy (%)				KHATT Accuracy (%)			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Method 1	92.80%	89.10%	87.32%	88.20%	89.67%	86.44%	84.00%	85.20%
Method 2	91.45%	88.05%	85.13%	86.56%	88.40%	84.20%	82.75%	83.47%
Method 3	93.92%	91.11%	88.79%	89.94%	91.02%	87.58%	85.41%	86.48%
Method 4	89.63%	86.23%	84.56%	85.38%	87.24%	83.12%	80.90%	81.99%
Method 5	90.71%	87.32%	85.07%	86.18%	88.56%	84.77%	82.50%	83.62%
Full Model	98.94%	97.66%	96.33%	97.00%	97.36%	94.80%	93.44%	94.05%

Model 1 is also very strong on both datasets, achieving accuracies of 92.80% for Devnagari and 89.67% for KHATT. The precision, recall, and F1-score values all indicate similar strengths, with the highest precision and recall being 89.10% and 87.32%, respectively, for Devnagari. For KHATT, the corresponding precision and recall values are 86.44% and 84.00%. Even as a strong runner-up, Method 1 is still behind the full model on all metrics. Model 2 has slightly weaker

performance than Method 1, with accuracies of 91.45% (Devnagari) and 88.40% (KHATT). Precision and recall scores also follow the same trend of decrease, with Devnagari precision at 88.05% and recall at 85.13%, whereas KHATT scores fall to 84.20% for precision and 82.75% for recall. The F1-scores overall also decrease, which means that Method 2's performance is slightly weaker than Method 1. The Model 3 is the most accurate method in terms of both accuracy and overall performance. It achieves the best accuracy of 93.92% for Devnagari and 91.02% for KHATT, surpassing both Method 1 and Method 2. It also has the best improvements in precision, recall, and F1-score, especially for Devnagari (91.11%, 88.79%, and 89.94%, respectively) and for KHATT (87.58%, 85.41%, and 86.48%, respectively). These findings indicate that Method 3 achieves a more optimal balance between recall and precision than the earlier methods (Table 6).

4.7 Research limitations

Although our proposed model for handwriting recognition has demonstrated promising results, several limitations need to be addressed. A list of five key limitations and their potential solutions is discussed below:

1). The model can be challenged by extreme handwriting changes, specifically inconsistent handwriting styles, which can be detrimental to feature extraction and compromise performance. This problem can be addressed by utilizing adaptive normalization for domain alignment, which standardizes handwriting styles during the training procedure to improve style robustness and reduce variation between writers.

2). The hybrid CA-CapsResNet exhibits high computational requirements, making real-time or resource-constrained deployment challenging. To mitigate this issue, developers can investigate model compression methods, such as pruning, quantization, and knowledge distillation, aiming to reduce the computational burden while maintaining acceptable recognition performance.

3). The model's stability can be compromised when exposed to noisy or low-quality images, as its ability to extract significant features can be adversely affected. In the future, this can be reduced by using noise-robust preprocessing modules, such as denoising autoencoders and contrastive regularization, to extract stable features under challenging imaging conditions.

4). Limited generalization ability may occur, particularly when training on datasets with insufficient diversity in handwriting samples, resulting in a decline in performance when applied to unseen data. Future work may focus on increasing data variability through synthetic augmentation, cross-domain adaptation, and style transfer to enhance model generalization to unseen scripts and writing conditions.

5). Handwriting types with distinctive or excessively ornate characteristics, such as calligraphy or stylized cursive, may cause difficulties for the model, as their distinctive traits may not align well with the feature extraction approaches, thereby reducing accuracy. This challenge can be alleviated by utilizing multi-scale spatial transformer networks or semantic regularisation mechanisms that can dynamically respond to structural deformations and the utilization of complex stroke structures.

5. CONCLUSION AND FUTURE SCOPE

In this paper, a new handwriting recognition system is introduced that integrates RABs, SCE, SAG, hierarchical capsule encoding, and a hybrid puma-crested porcupine optimizer for feature reduction. This hybrid model can better maintain hierarchies of features in both local and global contexts while also improving the spatial integration of those feature structures and learning hierarchical representations. Quantitatively, the framework achieved an accuracy of 98.94% and 97.36% (with feature reduction) on the Devanagari and KHATT datasets, respectively, compared with an accuracy of 93.28% and 91.91% without feature reduction, representing improvements of 5.66% and 5.45% (Table 5). These results differ significantly from two traditional capsule-based models, where CapsNet improved by almost 18% or Multi-Prototype CapsNet by 13 to 14%, but with reduced accuracy, as their accuracies remained in the 86 to 90% range.

Therefore, the described framework represents a new state-of-the-art for handwriting recognition tasks, combining high accuracy, parameter efficiency, and stability through feature reduction. Nonetheless, several potential avenues exist for extending this work. One of these is to investigate cross-lingual handwriting recognition, where the model can be adapted and fine-tuned to recognize handwriting in various languages and scripts, including Chinese and Japanese.

Future research can aim to extend this model to cross-lingual handwriting recognition, enabling the model to generalize to different writing systems (e.g., Arabic, Chinese, or Japanese) through transfer learning or domain adaptation. Another avenue is to optimize this model for real-time implementation on edge and mobile devices, which would reduce latency and the computational burden on educational, document authentication, and banking digitization applications. In addition, the feature selection based on hybrid optimization could be enhanced through adaptive or dynamic feature selection, which would enable the neural network to dynamically focus on a smaller subset of important features depending on the characteristics of the input. Ultimately, this system is adaptable for commercial and forensic applications, including automated document verification, writer identification, and handwriting-based behavioral analysis.

REFERENCES

[1] Humayun, M., Siddiqi, R., Uddin, M., Kandhro, I.A., Abdelhaq, M., Alsaqour, R. (2024). A novel methodology for offline English handwritten character recognition using ELBP-based sequential (CNN). Neural Computing and Applications, 36(30): 19139-19156. https://doi.org/10.1007/s00521-024-10206-1

[2] Khan, A.A., Shaikh, A.A., Laghari, A.A., Dootio, M.A., Rind, M.M., Awan, S.A. (2022). Digital forensics and cyber forensics investigation: Security challenges, limitations, open issues, and future direction. International Journal of Electronic Security and Digital Forensics, 14(2): 124-150. https://doi.org/10.1504/IJESDF.2022.121174

[3] Damayanti, F., Suzanti, I.O., Jauhari, A., Herawati, S. (2024). Restoration of Javanese characters based on Wasserstein Generative Adversarial Network-Gradient Penalty. Mathematical Modelling of Engineering

Problems, 11(12): 3447-3457. https://doi.org/10.18280/mmep.111223
[4] Tripathi, K.M., Kamat, P., Patil, S., Jayaswal, R., Ahirrao, S., Kotecha, K. (2023). Gesture-to-text translation using SURF for Indian sign language. Applied System Innovation, 6(2): 35. https://doi.org/10.3390/asi6020035
[5] Alghyaline, S. (2023). Arabic optical character recognition: A review. Computer Modeling in Engineering & Sciences, 135(3): 1825-1861. https://doi.org/10.32604/cmes.2022.024555
[6] Kumar, M., Jindal, M.K., Kumar, M. (2022). Distortion, rotation and scale invariant recognition of hollow Hindi characters. Sādhanā, 47(2): 92. https://doi.org/10.1007/s12046-022-01847-w
[7] Lee, S.H., Yu, W.F., Yang, C.S. (2022). ILBPSDNet: Based on improved local binary pattern shallow deep convolutional neural network for character recognition. IET Image Processing, 16(3): 669-680. https://doi.org/10.1049/ipr2.12226
[8] Sharma, N.K., Rahamatkar, S., Rathore, A.S. (2024). Comprehensive analysis to detect optimal vehicle position for roadside traffic surveillance using lightweight contour-based CNN. International Journal of Transport Development and Integration, 8(1): 197-213. https://doi.org/10.18280/ijtdi.080119
[9] Mahadevappa, M., Aradhya, V.N.M., Basavaraju, H.T., Shivarudraswamy, S. (2024). CNN-DEdge: Multilingual scene text detection and extraction. Mathematical Modelling of Engineering Problems, 11(11): 3152-3160. https://doi.org/10.18280/mmep.111125
[10] Noor, M.H.M., Ige, A.O. (2025). A survey on state-of-the-art deep learning applications and challenges. Engineering Applications of Artificial Intelligence, 159: 111225. https://doi.org/10.1016/j.engappai.2025.111225
[11] Truong, T.N., Nguyen, C.T., Zanibbi, R., Mouchère, H., Nakagawa, M. (2024). A survey on handwritten mathematical expression recognition: The rise of encoder-decoder and GNN models. Pattern Recognition, 153: 110531. https://doi.org/10.1016/j.patcog.2024.110531
[12] Younesi, A., Ansari, M., Fazli, M., Ejlali, A., Shafique, M., Henkel, J. (2024). A comprehensive survey of convolutions in deep learning: Applications, challenges, and future trends. IEEE Access, 12: 41180-41218. https://doi.org/10.1109/ACCESS.2024.3376441
[13] Xi, E., Bing, S., Jin, Y. (2017). Capsule network performance on complex data. arXiv preprint arXiv:1712.03480. https://doi.org/10.48550/arXiv.1712.03480
[14] Abdel-Basset, M., Mohamed, R., Abouhawwash, M. (2024). Crested porcupine optimizer: A new nature-inspired metaheuristic. Knowledge-Based Systems, 284: 111257. https://doi.org/10.1016/j.knosys.2023.111257
[15] Abdollahzadeh, B., Khodadadi, N., Barshandeh, S., Trojovský, P., Gharehchopogh, F.S., El-kenawy, E.S.M., Mirjalili, S. (2024). Puma optimizer (PO): A novel metaheuristic optimization algorithm and its application in machine learning. Cluster Computing, 27(4): 5235-5283. https://doi.org/10.1007/s10586-023-04221-5
[16] Yao, H., Tan, Y., Xu, C., Yu, J., Bai, X. (2021). Deep capsule network for recognition and separation of fully overlapping handwritten digits. Computers & Electrical Engineering, 91: 107028. https://doi.org/10.1016/j.compeleceng.2021.107028
[17] Parcham, E., Ilbeygi, M., Amini, M. (2021). CBCapsNet: A novel writer-independent offline signature verification model using a CNN-based architecture and capsule neural networks. Expert Systems with Applications, 185: 115649. https://doi.org/10.1016/j.eswa.2021.115649
[18] Moudgil, A., Singh, S., Rani, S., Shabaz, M., Alsubai, S. (2024). Deep learning for ancient scripts recognition: A CapsNet-LSTM based approach. Alexandria Engineering Journal, 103: 169-179. https://doi.org/10.1016/j.aej.2024.01.017
[19] Abd Elaziz, M., Lu, S., He, S. (2021). A multi-leader whale optimization algorithm for global optimization and image segmentation. Expert Systems with Applications, 175: 114841. https://doi.org/10.1016/j.eswa.2021.114841
[20] Al-Saffar, A., Awang, S., Al-Saiagh, W., Al-Khaleefa, A.S., Abed, S.A. (2021). A sequential handwriting recognition model based on a dynamically configurable CRNN. Sensors, 21(21): 7306. https://doi.org/10.3390/s21217306
[21] Altwaijry, N., Al-Turaiki, I. (2021). Arabic handwriting recognition system using convolutional neural network. Neural Computing and Applications, 33(7): 2249-2261. https://doi.org/10.1007/s00521-020-05107-0
[22] Kavitha, B.R., Srimathi, C.B. (2022). Benchmarking on offline handwritten tamil character recognition using convolutional neural networks. Journal of King Saud University - Computer and Information Sciences, 34(4): 1183-1190. https://doi.org/10.1016/j.jksuci.2018.09.004
[23] Assael, Y., Sommerschield, T., Shillingford, B., Bordbar, M., Pavlopoulos, J., Chatzipanagiotou, M., De Freitas, N. (2022). Restoring and attributing ancient texts using deep neural networks. Nature, 603(7900): 280-283. https://doi.org/10.1038/s41586-022-04448-z
[24] Das, S., Imtiaz, M.S., Neom, N.H., Siddique, N., Wang, H. (2023). A hybrid approach for Bangla sign language recognition using deep transfer learning model with random forest classifier. Expert Systems with Applications, 213: 118914. https://doi.org/10.1016/j.eswa.2022.118914
[25] Chauhan, V.K., Singh, S., Sharma, A. (2024). HCR-Net: A deep learning based script independent handwritten character recognition network. Multimedia Tools and Applications, 83(32): 78433-78467. https://doi.org/10.1007/s11042-024-19263-8
[26] Hadadi, S., Arabani, S.P. (2024). A novel approach for Parkinson's disease diagnosis using deep learning and Harris Hawks optimization algorithm with handwritten samples. Multimedia Tools and Applications, 83(34): 81491-81510. https://doi.org/10.1007/s11042-024-18584-3
[27] Mohamad, M.A., Ahmad, M.A., Mahmood, J. (2024). Feature extraction algorithm based metaheuristic optimization for handwritten character recognition. Journal of Telecommunication, Electronic and Computer Engineering, 16(2): 27-30. https://doi.org/10.54554/jtec.2024.16.02.004
[28] Acharya, S., Pant, A.K., Gyawali, P.K. (2015). Deep learning based large scale handwritten Devanagari character recognition. In 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Kathmandu, Nepal, pp. 1-6.

https://doi.org/10.1109/SKIMA.2015.7400041
[29] Mahmoud, S.A., Ahmad, I., Alshayeb, M., Al-Khatib, W.G., Parvez, M.T., Fink, G.A., El Abed, H. (2012). Khatt: Arabic offline handwritten text database. In 2012 International Conference on Frontiers in Handwriting Recognition, Bari, Italy, pp. 449-454. https://doi.org/10.1109/ICFHR.2012.224
[30] Mirsadeghi, M., Shalchian, M., Kheradpisheh, S.R., Masquelier, T. (2023). Spike time displacement-based error backpropagation in convolutional spiking neural networks. Neural Computing and Applications, 35(21): 15891-15906. https://doi.org/10.1007/s00521-023-08567-0
[31] Viswanathan, K., Sethi, A. WaveMix-Lite: A Resource-efficient neural network for image analysis. arXiv pre-print arXiv.2205.14375. https://doi.org/10.48550/arXiv.2205.14375
[32] Mridha, M.F., Ohi, A.Q., Shin, J., Kabir, M.M., Monowar, M.M., Hamid, M.A. (2021). A thresholded Gabor-CNN based writer identification system for Indic scripts. IEEE Access, 9: 132329-132341. https://doi.org/10.1109/ACCESS.2021.3114799
[33] Semma, A., Hannad, Y., Siddiqi, I., Djeddi, C., El Kettani, M.E.Y. (2021). Writer identification using deep learning with fast keypoints and Harris corner detector. Expert Systems with Applications, 184: 115473. https://doi.org/10.1016/j.eswa.2021.115473
[34] Ali, A.A.A., Mallaiah, S. (2022). Intelligent handwritten recognition using hybrid CNN architectures based-SVM classifier with dropout. Journal of King Saud University-Computer and Information Sciences, 34(6): 3294-3300. https://doi.org/10.1016/j.jksuci.2021.01.012
[35] Moudgil, A., Singh, S., Gautam, V., Rani, S., Shah, S.H. (2023). Handwritten Devanagari manuscript characters recognition using CapsNet. International Journal of Cognitive Computing in Engineering, 4: 47-54. https://doi.org/10.1016/j.ijcce.2023.02.001
[36] Tan, Y., Yao, H. (2021). Deep capsule network handwritten digit recognition. International Journal of Advanced Network, Monitoring and Controls, 5(4): 1-8. https://doi.org/10.21307/ijanmc-2020-031
[37] Rani, N.S., BR, P. (2022). Robust recognition technique for handwritten Kannada character recognition using capsule networks. International Journal of Electrical & Computer Engineering, 12(1): 383-391. https://doi.org/10.11591/ijece.v12i1.pp383-391
[38] Abbassi, S., Ghiasi-Shirazi, K., Harati, A. (2024). Deep multi-prototype capsule networks. arXiv preprint arXiv:2404.15445. https://doi.org/10.1007/s11063-023-11155-x
[39] Wu, X.J., Ao, X., Zhang, R.S., Liu, C.L. (2023). Structural recognition of handwritten Chinese characters using a modified part capsule auto-encoder. In CAAI International Conference on Artificial Intelligence, pp. 478-490. https://doi.org/10.1007/978-981-99-8850-1_39