Computers, Materials & Continua
DOI: 10.32604/cmc.2023.033534
Article

Sailfish Optimizer with Deep Transfer Learning-Enabled Arabic Handwriting Character Recognition

Mohammed Maray[1], Badriyya B. Al-onazi[2], Jaber S. Alzahrani[3], Saeed Masoud Alshahrani[4,*], Najm Alotaibi[5], Sana Alazwari[6], Mahmoud Othman[7] and Manar Ahmed Hamza[8]

[1]Department of Information Systems, College of Computer Science, King Khalid University, Abha, Saudi Arabia
[2]Department of Language Preparation, Arabic Language Teaching Institute, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh, 11671, Saudi Arabia
[3]Department of Industrial Engineering, College of Engineering at Alqunfudah, Umm Al-Qura University, Saudi Arabia
[4]Department of Computer Science, College of Computing and Information Technology, Shaqra University, Shaqra, Saudi Arabia
[5]Prince Saud AlFaisal Institute for Diplomatic Studies, Riyadh, Saudi Arabia
[6]Department of Information Technology, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif, 21944, Saudi Arabia
[7]Department of Computer Science, Faculty of Computers and Information Technology, Future University in Egypt, New Cairo, 11835, Egypt
[8]Department of Computer and Self Development, Preparatory Year Deanship, Prince Sattam bin Abdulaziz University, AlKharj, Saudi Arabia
*Corresponding Author: Saeed Masoud Alshahrani. Email: salshahrani@su.edu.sa
Received: 20 June 2022; Accepted: 29 September 2022

Abstract: The recognition of the Arabic characters is a crucial task in computer vision and Natural Language Processing fields. Some major complications in recognizing handwritten texts include distortion and pattern variabilities. So, the feature extraction process is a significant task in NLP models. If the features are automatically selected, it might result in the unavailability of adequate data for accurately forecasting the character classes. But, many features usually create difficulties due to high dimensionality issues. Against this background, the current study develops a Sailfish Optimizer with Deep Transfer Learning-Enabled Arabic Handwriting Character Recognition (SFODTL-AHCR) model. The projected SFODTL-AHCR model primarily focuses on identifying the handwritten Arabic characters in the input image. The proposed SFODTL-AHCR model pre-processes the input image by following the Histogram Equalization approach to attain this objective. The Inception with ResNet-v2 model examines the pre-processed image to produce the feature vectors. The Deep Wavelet Neural Network (DWNN) model is utilized to recognize the handwritten Arabic characters. At last, the SFO algorithm is utilized for fine-tuning the parameters involved in the DWNN model to attain better performance. The performance of the proposed SFODTL-AHCR model was validated using a series of images. Extensive comparative analyses were conducted. The proposed method achieved a

maximum accuracy of 99.73%. The outcomes inferred the supremacy of the proposed SFODTL-AHCR model over other approaches.

Keywords: Arabic language; handwritten character recognition; deep learning; feature extraction; hyperparameter tuning

1 Introduction

Automatic handwriting detection corresponds to the capability of a mechanism that can identify the handwritten inputs of a human being. The handwritten texts are available in multiple forms such as touch screens, paper documents, images, and other gadgets. The handwritten texts can be retrieved as an input by scanning the content offline or from the online content that is written using a digital pen tip [1]. The handwriting detection process is one of the important challenges in Computer Vision (CV) methodologies. In general, the handwriting of a human being varies from that of the others, while a person's handwriting to changes from time to time [2]. The major complexities in the handwriting detection process involve distortions and paradigm changeability, due to which the feature extraction process becomes an important task. If the structures are selected physically, it might end up in inadequate data. This inadequate data cannot predict the character classes accurately. However, many features usually cause issues due to high dimensionality [3,4].

Recognition is a widely discussed topic in various domains such as the character recognition, face recognition, numerals recognition, fingerprint recognition, image recognition and so on. The handwritten Character Recognition (HDR) mechanism [5] is an intellectual mechanism that can categorize handwritten characters from the perspectives of a human being. Character classification [6] plays a significant role in many CV and image processing methods such as license plate recognition, Optical Character Recognition (OCR) and so on [7]. The categorization of the handwritten characters is a highly-complicated task owing to the distinct handwriting styles of the writers. Recently, the Handwritten Arabic Character Recognition (HACR) domain has attracted substantial interest among researchers. Various authors have contributed important advancements in this domain that reflect the rapid growth of DL techniques. Arabic is a Semitic language [8,9] spoken across the Middle Eastern nations and is also the mother tongue of millions of people across the globe.

The Arabic alphabet has a total of 28 characters. The conventional OCR mechanisms lack efficacy since the character features are hard-coded and are employed to match the characters. Conversely, Neural networks (NN) can study these features by analyzing the datasets. So, the requirement for manual hard-coding of the features gets simplified [10]. During the training procedure, the neural network studies different variables, making it highly flexible in case the handwriting styles get altered. The DL technique, a domain in ML, uses representation learning by stating the input data at different levels in simple depictions [11,12]. Conversely, complex ideas like a 'person' or a 'car' are built via the layers of simple ideas like edges or contours.

The current study develops a Sailfish Optimizer with a Deep Transfer Learning Enabled Arabic Handwriting Character Recognition (SFODTL-AHCR) model. The major aim of the projected SFODTL-AHCR model is to identify the handwritten Arabic characters in the input image. To attain this, the SFODTL-AHCR model pre-processes the input image by following the Histogram Equalization approach. The Inception with ResNet-v2 model examines the pre-processed image to

produce the feature vectors. In order to recognize the handwritten Arabic characters, the Deep Wavelet Neural Network (DWNN) model is utilized. At last, the SFO algorithm is utilized for fine-tuning the DWNN parameters to attain better performance. The performance of the proposed SFODTL-AHCR model was experimentally validated using a series of images.

2 Related Works

Ali et al. [13] discussed the character recognition process for Arabic script in detail. In this study, some established techniques were summarized, whereas a few techniques were also examined to build a powerful HCR mechanism. Further, the authors also involved a certain set of upcoming studies on recognizing handwritten characters. This review examined and discussed about several techniques such as the feature extraction techniques, pre-processing approaches, segmentation approaches and several classification methods for the recognition of the Arabic characters. Alkhateeb [14] presented an efficient technique to recognize the isolated handwritten Arabic characters on the basis of the DL method. Here, the DL approach was conceived based on Convolutional Neural networks (CNNs). The CNN method plays a significant role in each application of the CV field. It can be trained and used for the purpose of Arabic handwritten character recognition in offline mode with the help of three HACR datasets.

Salam et al. [15] suggested a novel structure for Offline Isolated Arabic Handwriting Character Recognition System based on SVM (OIAHCR). This study suggested an Arabic handwriting dataset for training and testing purposes. Though half of the dataset was utilized for training the Support Vector Machine (SVM) model, the rest of the dataset was utilized to test and validate the proposed model. The proposed system attained maximum performance using the least volume of the training dataset. In literature [16], the researchers devised and validated a deep ensemble structure in which the ResNet18 structure was exploited for modelling and classification of the handwritten character images. Specifically, the researchers adapted the ResNet-18 model by totalling a dropout layer. All the convolutional layers were combined through multiple ensemble methods to detect the isolated-handwritten Arabic characters mechanically. Deore et al. [17] suggested a Devanagari Handwritten Character Recognition System (DHCRS) in which the DCNN (Deep CNN) method was applied as a fine-tuning technique in the examination and the classification of the Devanagari Handwritten characters. An open-source novel Devanagari handwritten characters' dataset was developed in this study. A two-stage VGG16 DL method was applied to the data to recognize such characters with the help of two enhanced adaptive-gradient methodologies. A two-stage DL technique was advanced to enhance the overall success of the suggested Devanagari Handwritten Character Recognition System (DHCRS).

3 Design of SFODTL-AHCR Model

In this study, a new SFODTL-AHCR model has been developed to recognize the handwritten Arabic characters in the input image. To attain this, the SFODTL-AHCR model pre-processes the input image by following the Histogram Equalization approach. The Inception with ResNet-v2 model examines the pre-processed image to produce the feature vectors. To recognize the handwritten Arabic characters, the SFO model is utilized with the DWNN model. Fig. 1 illustrates the overall processes involved in the SFODTL-AHCR approach.

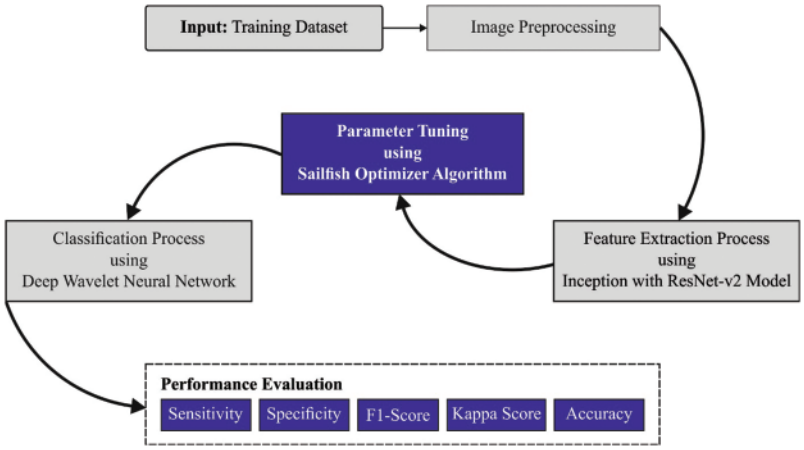

Figure 1: Overall processes involved in the SFODTL-AHCR approach

3.1 Feature Extraction: Inception-ResNetv2 Model

CNN is a widely-applied DL algorithm used in the classification tasks like image classification. Initially, it is characterized by an image whereas the computer vision process is used as the visual cortex. The image sensor is convolved in this model by a series of $d \times d$ kernels. This convolution (i.e., 'Feature Map') is stacked to characterize the distinct features identified by the filter. This phenomenon is explained in the following equation.

$$A_j = f\left(\sum\nolimits_{i=1}^{N} I_i * K_{i,j} + B_j\right) \tag{1}$$

In every matrix, I_i is convolved with the corresponding kernel matrix $K_{i,j}$ and the bias of B_j. Eventually, an activation function is provided for every component. In the CNN training process, both weight and the bias are modified by the feature detection filter before the backpropagation (BP) process. A feature map filter is performed over three channels. To lessen the computation complexity, the CNN method exploits a pooling layer using a single layer in the network to reduce the size of the resultant layer from the input. The pooling process is applied to reduce the outcomes to safeguard the significant features. max pooling model is an established pooling model in which a large activation is carefully chosen from the pooling window. The CNN method makes use of a discriminatory function that employs the BP approach, which is established from a sigmoid or a Rectified Linear Unit (ReLU) function. The last layer has a *softmax* activation function for multiple class problems and a single node with a sigmoid function for dual classification.

$$f(x) = \frac{1}{1+e^{-x}} \in (0,1) \tag{2}$$

$$f(x) = \max(0, x) \tag{3}$$

$$\sigma(z)_j = \frac{e^{z_i}}{\Sigma_{k=1}^{K} e^{z_k}} \tag{4}$$

$$\forall j \in \{1, \ldots, K\}$$

Inception-ResNet-V2 (IRV2), a hybrid of the ResNet and GoogLeNet (Inception) models, was launched by Google, and it serves as a contemporary image classification technique. The Inception model is generally interconnected with its respective layers, as found in the GoogleNet [18]. A tiny convolution kernel is employed to efficiently extract the image features and reduce the module parameters. Next, a large-scale convolution kernel enhances the parameters of the matrices. In this scenario, a small-scale convolution kernel is interchanged correspondingly to limit the parameter's function for comparable receptive fields. To conclude, it is an accurate and an extensive network that can be compared with the Inception module. Recently, Inception v1–v4 has been proposed as a widespread GoogleNet method. The primary goal of the ResNet model is to encompass a direct link of the module in the name of Highway Network. Further, it allows an original input dataset to be communicated directly to the successive layer. On the other hand, the ResNet approach safeguards the privacy of the information through straightforward communication to the output. Adjustments are required between the outputs and the inputs so as to learn about the benefits and disadvantages of the model. For the Residual-Inception module, the Inception model is employed with lower processing difficulty than the original Inception model. Eventually, the residual and non-residual Inception models have a slight modification. In the case of Inception-ResNet model, the Batch Normalization (BN) process is exploited initially for the traditional layers. The exploitation of the best activation size intake demands a huge GPU memory. The maximal Inception module is included through the elimination of the BN layers after the completion of the activation process. In addition, once the filter count exceeds 1000, the residual networks become unreliable whereas the network training model gets ceased.

3.2 Handwritten Character Recognition: DWNN Model

In this study, the DWNN model is utilized to recognize handwritten characters. The general properties of the NN architecture reduce the linear set of an essential function and are heavily employed in the evaluation of the function $\zeta \in L^2(\Re)$. It is a four-layered structure and the layers are as follows; product, input, output and the wavelon states [19]. Though the pre-processed dataset acts as an input, the wavelon state only allows the dataset using wavelet AF. The in-depth features of the dataset are eliminated in this stage using dilated and translated forms of the wavelet function. The result from these processes is previously evaluated at the product state, and the conclusion is provided for the output. Fig. 2 showcases the infrastructure of the WNN model. The n dimensional-biased network creates the result using the m node as given herewith.

$$\zeta = \omega^T \varphi(x, \tau, \sigma) + \mu^T t\phi(X, \tau, \sigma) \tag{5}$$

In Eq. (5), $x = [x_1, x_2, \ldots, x_n]^T \in R^n$ indicates the input vector, $\tau = [\tau_1, \tau_2, \ldots, \tau_m]^T \in R^{mxn}$ and $\sigma = [\sigma_1, \sigma_2, \cdots, \sigma_m]^T \in R^{mxn}$ denote the translation and dilation parameters respectively, $\varphi = [\varphi_1, \varphi_2, \cdots, \varphi_m.]^T \in \Re^m$ indicates the wavelet function and $\phi = [\phi_1, \phi_2, \cdots, \phi_m]^T \in \Re^m$ indicates the corresponding bias function. The bias and the weight functions of the network are established by $\omega = [\omega_1, \ldots, \omega_m]^T \in R^m$ and $\mu = [\mu_1, \ldots, \mu_m]^T \in R^m$, respectively. Optimizing the network parameters is an important feature in the DWNN network as it deals with the network's learning curve. An optimal variable vector attains the optimal computation of a desirable function. The optimization of the network is accomplished by estimating the function as defined herewith.

$$\hat{\zeta}(x(n)) = \hat{\omega}^T \phi(x(n)) + \hat{\mu}^T \phi(x(n)) \tag{6}$$

In Eq. (13), $\hat{\omega}, \hat{\mu}$ denote the optimal assessed values of the parameters, ω^*, μ^* respectively. The optimization problem is reframed by evaluating the error as follows.

$$\widetilde{\zeta}\left(x\left(n\right)\right) = \zeta\left(x\left(n\right)\right) - \hat{\zeta}\left(x\left(n\right)\right) = \left\{\widetilde{\omega}^T\left(n\right) \varnothing\left(x\left(n\right)\right) + \widetilde{\mu}^T\left(n\right) \varphi\left(x\left(n\right)\right) + \varepsilon\left(x\left(n\right)\right)\right\} \tag{7}$$

The aim is to reduce the assessed value of the error $\widetilde{\zeta}$ to an arbitrarily-smaller value by electing the resolution count. An adaptive method is exploited to fine-tune the DWNN architecture's weight and produce the gradient descent algorithm. The corresponding law that tunes to the weight is given herewith.

$$\omega\left(n+1\right) = \omega\left(n\right) + \Delta\omega\left(n\right) = \omega\left(n\right) + \mu\left(-\frac{\partial e\left(n\right)}{\partial\omega\left(n\right)}\right) \tag{8}$$

In Eq. (8), the learning rate and the tuning weights are denoted by μ and ω, respectively. Though the weight gets altered, the approximation error cannot be minimized for fewer values.

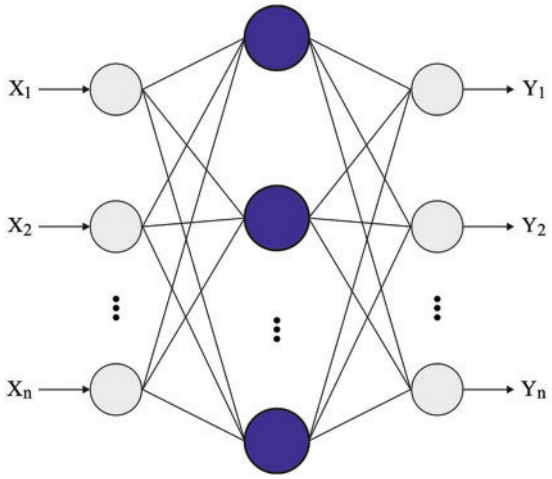

Figure 2: Framework of the WNN model

3.3 Hyperparameter Tuning: SFO Algorithm

In this stage, the SFO algorithm is utilized for fine-tuning the parameters involved in the DWNN model to attain a better performance. Generally, SFO [20] is identified as a population-dependent metaheuristic algorithm that relies upon the attack-alternation principles of a group of hunter sailfishes. A sailfish is disseminated in the searching region, whereas the position of the sardines helps in detecting the optimum solution d in the searching region. It has an optimal fitness measure called 'elite' sailfish, whereas its location at i^{th} iteration is given by $P^i_{SlfBest}$. For sardine, the injured one has the best fitness value and its position in i^{th} iteration is represented by $P^i_{SrdInjured}$. During every iteration, the places of both sailfishes and the sardines are exploited. In *the* $i+1^{th}$ iteration, a novel location i.e., P^{i+1}_{Slf} of the sailfish is upgraded by employing the 'injured' sardine and an 'elite' sailfish as follows.

$$P^{i+1}_{Slf} = P^i_{SlfBest} - \mu_i \times \left(rnd \times \frac{P^i_{Slf\,\mathrm{Best}} + P^i_{SrdInjured}}{2} - P^i_{Slf}\right) \tag{9}$$

In Eq. (9), P^i_{Slf} denotes the previous location of the Slf^{th} sailfish *and* denotes an arbitrary integer between [0, 1] and μ_i indicating a coefficient that is generated as given herewith.

$$\mu_i = 2 \times rnd \times PrD - PrD \tag{10}$$

In Eq. (10), PrD defines the prey density that characterizes the count of the prey during every iteration. Then, the measure of PrD is evaluated as given below, whereas the decreased prey count gets constrained at the time of group hunting.

$$PrD = 1 - \frac{Num_{Slf}}{Num_{Slf} + Num_{Srd}} \tag{11}$$

In Eq. (11), Num_{Slf} and Num_{Srd} represent the values of the sardines and the sailfishes correspondingly.

$$Num_{Slf} = Num_{Srd} \times Prcnt \tag{12}$$

In Eq. (12), $Print$ represents the ratio of the sardine population that improves a primary sailfish population. The count of the sardines is better than the count of the sailfishes. The location of the sardines gets upgraded during every iteration as given below.

$$P_{Srd}^{i+1} = rnd\,(0,\ 1) \times \left(P_{SLfBest}^{i} - P_{Srd}^{i} + ATK\right) \tag{13}$$

$$ATK = A \times (1 - (2 \times itr \times \kappa)) \tag{14}$$

In this expression, P_{Srd}^{i} and P_{Srd}^{i+1} indicate the current and the former positions of the sardine, respectively, whereas ATK denotes the attacking efficacy of the sailfishes at itr iteration. Next, the sardine count upgrades the position and displacement count based on ATK. The reduction of the ATK assists in the convergence of the searching agent. In the application of the parameter ATK, the sardine count upgrades the location (γ), whereas the variable count (δ) is described as follows.

$$\gamma = Num_{Srd} \times ATK \tag{15}$$

$$\delta = v \times ATK \tag{16}$$

Let v be the variable value and Num_{Srd} refers to the count of the sardines. After the sardines are simplified compared to the sailfishes, the sailfishes exploit the place, whereas the sardines get detached from the population. A random selection of the sardines and the sailfishes safeguards the detection of the searching region. Because the attack capacity of the sailfish gets reduced during every iteration. It allows the sardines to escape from the sailfish, which in turn assists in the exploitation phase. The ATK parameter deals with the identification of a trade-off between exploitation and the exploration phases.

4 Results and Discussion

The current section inspects and validates the performance of the proposed SFODTL-AHCR model under a different number of epochs. Table 1 details the experimental outcomes of the proposed SFODTL-AHCR model during training phase with 1,000 epochs. Fig. 3 depicts the extensive analytical outcomes achieved by the proposed SFODTL-AHCR model during training phase with 1000 epochs. The results confirmed that the proposed SFODTL-AHCR model produced effectual outputs during each run. For instance, on run-1, the SFODTL-AHCR model achieved $accu_{racy}$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.14%, 99.78%, 99.36%, 99.32% and 99.27% respectively. Also, on run-2, the proposed SFODTL-AHCR method rendered $accu_{racy}$, $sens_y$, $spec_y$, $F1_{score}$ and kappa

values such as 99.53%, 99.33%, 99.82%, 99.15% and 99.42% correspondingly. Meanwhile, on run-3, the proposed SFODTL-AHCR algorithm offered $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.40%, 99%, 99.40%, 99.54% and 99.14% correspondingly. Eventually, on run-4, the presented SFODTL-AHCR technique accomplished $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.75%, 99.43%, 99.25%, 99.37% and 99.07% correspondingly. At last, on run-5, the proposed SFODTL-AHCR method yielded $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.18%, 99.56%, 99.67%, 99.21% and 99.16% correspondingly.

Table 1: Analytical results of the SFODTL-AHCR approach on epoch-1000 under different measures

Epoch-1000					
Methods	Accuracy	Sensitivity	Specificity	F1-Score	Kappa Score
Training phase					
Run-1	99.14	99.78	99.36	99.32	99.27
Run-2	99.53	99.33	99.82	99.15	99.42
Run-3	99.40	99.00	99.40	99.54	99.14
Run-4	99.75	99.43	99.25	99.37	99.07
Run-5	99.18	99.56	99.67	99.21	99.16
Average	99.40	99.42	99.50	99.32	99.21
Testing phase					
Run-1	99.07	99.29	99.48	99.85	99.37
Run-2	99.58	99.52	99.61	99.52	99.18
Run-3	99.76	99.31	99.53	99.08	99.85
Run-4	99.51	99.6	99.38	99.34	99.57
Run-5	99.45	99.58	99.68	99.31	99.39
Average	99.47	99.46	99.54	99.42	99.47

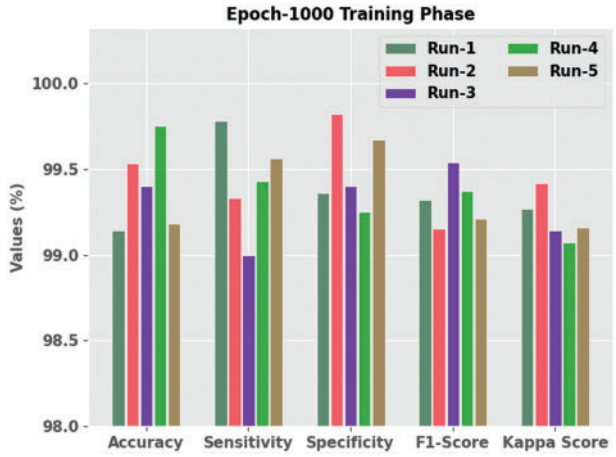

Figure 3: Analytical results of the SFODTL-AHCR approach during training phase with 1000 epochs

Fig. 4 portrays the extensive outcomes of the proposed SFODTL-AHCR method during testing phase with 1,000 epochs. The results portray that the proposed SFODTL-AHCR model produced effectual outputs for each run. For instance, on run-1, the proposed SFODTL-AHCR algorithm achieved $accu_{racy}$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.07%, 99.29%, 99.48%, 99.85% and 99.37% correspondingly. Also, on run-2, the proposed SFODTL-AHCR technique achieved $accu_{racy}$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.58%, 99.52%, 99.61%, 99.52% and 99.18% correspondingly. Likewise, on run-3, the presented SFODTL-AHCR approach attained $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.76%, 99.31%, 99.53%, 99.08% and 99.85% correspondingly. Eventually, on run-4, the proposed SFODTL-AHCR method accomplished $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.51%, 99.6%, 99.38%, 99.34% and 99.57% correspondingly. At last, on run-5, the presented SFODTL-AHCR model yielded $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.45%, 99.58%, 99.68%, 99.31% and 99.39% correspondingly.

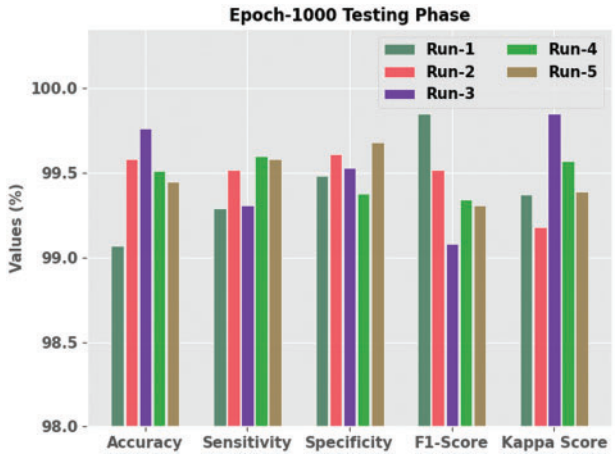

Figure 4: Analytical results of the SFODTL-AHCR approach under testing phase with 1,000 epochs

Table 2 details about the experimental outcomes of the presented SFODTL-AHCR model during the training phase with 2,000 epochs. Fig. 5 shows the extensive analytical results achieved by the SFODTL-AHCR model during training phase with 2,000 epochs. The results expose that the proposed SFODTL-AHCR model produced effectual outputs during each run. For instance, on run-1, the SFODTL-AHCR method offered $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.69%, 99.38%, 99.39%, 99.32% and 99.33% correspondingly. Additionally, on run-2, the proposed SFODTL-AHCR model attained $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99%, 99.69%, 99.76%, 99.71% and 99.68% correspondingly. Likewise, on run-3, the proposed SFODTL-AHCR model accomplished $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.63%, 99.68%, 99.11%, 99.17% and 99.07% respectively. Eventually, on run-4, the proposed SFODTL-AHCR method rendered $accu_{racy}$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.54%, 99.07%, 99.02%, 99.53% and 99.78% correspondingly. At last, on run-5, the presented SFODTL-AHCR model achieved $accu_{racy}$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.05%, 99.52%, 99.49%, 99.38% and 99.36% respectively.

Table 2: Analytical results of the SFODTL-AHCR approach under different measures on epoch-2000

Epoch-2000					
Methods	Accuracy	Sensitivity	Specificity	F1-Score	Kappa Score
Training phase					
Run-1	99.69	99.38	99.39	99.32	99.33
Run-2	99.00	99.69	99.76	99.71	99.68
Run-3	99.63	99.68	99.11	99.17	99.07
Run-4	99.54	99.07	99.02	99.53	99.78
Run-5	99.05	99.52	99.49	99.38	99.36
Average	99.38	99.47	99.35	99.42	99.44
Testing phase					
Run-1	99.89	99.70	99.31	99.03	99.04
Run-2	99.77	99.67	99.04	99.47	99.65
Run-3	99.57	99.22	99.72	99.32	99.75
Run-4	99.64	99.13	99.74	98.99	99.82
Run-5	99.80	99.85	99.33	99.36	99.68
Average	99.73	99.51	99.43	99.23	99.59

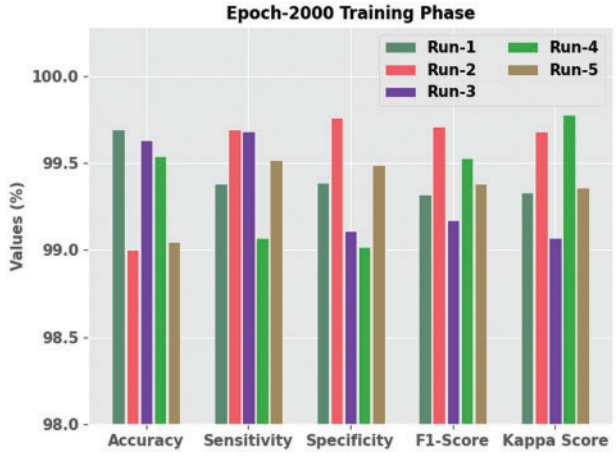

Figure 5: Analytical results of the SFODTL-AHCR approach during training phase with 2,000 epochs

Fig. 6 depicts the extensive analytical outcomes achieved by the proposed SFODTL-AHCR model during testing phase with 2,000 epochs. The results portray that the proposed SFODTL-AHCR model produced effectual outputs during each run. For example, on run-1, the presented SFODTL-AHCR model achieved $accu_{racy}$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.89%, 99.70%, 99.31%, 99.03% and 99.04% correspondingly. Besides, on run-2, the SFODTL-AHCR model yielded $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.77%, 99.67%, 99.04%, 99.47% and

99.65% correspondingly. Meanwhile, on run-3, the proposed SFODTL-AHCR model provided $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.57%, 99.22%, 99.72%, 99.32% and 99.75% respectively. Eventually, on run-4, the SFODTL-AHCR model accomplished $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.64%, 99.13%, 99.74%, 98.99%, and 99.82% respectively. At last, on run-5, the presented SFODTL-AHCR method attained $accu_y$, $sens_y$, $spec_y$, $F1_{score}$ and kappa values such as 99.80%, 99.85%, 99.33%, 99.36% and 99.68% respectively.

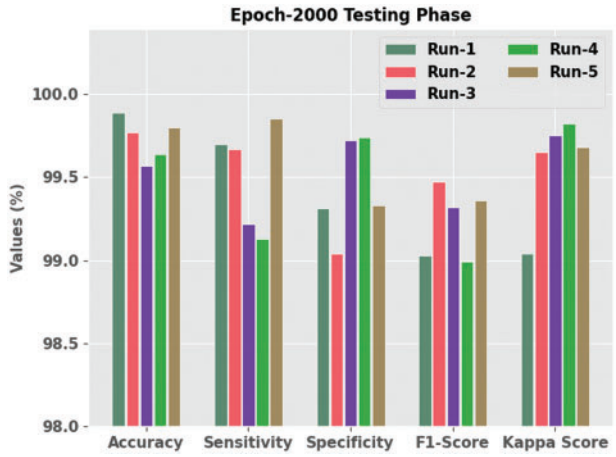

Figure 6: Analytical results of the SFODTL-AHCR approach during testing phase with 2,000 epochs

Both Training Accuracy (TA) and Validation Accuracy (VA) values, acquired by the proposed SFODTL-AHCR method on test dataset, are demonstrated in Fig. 7. The experimental outcomes infer that the proposed SFODTL-AHCR technique achieved the maximum TA and VA values whereas the VA values were higher than the TA values.

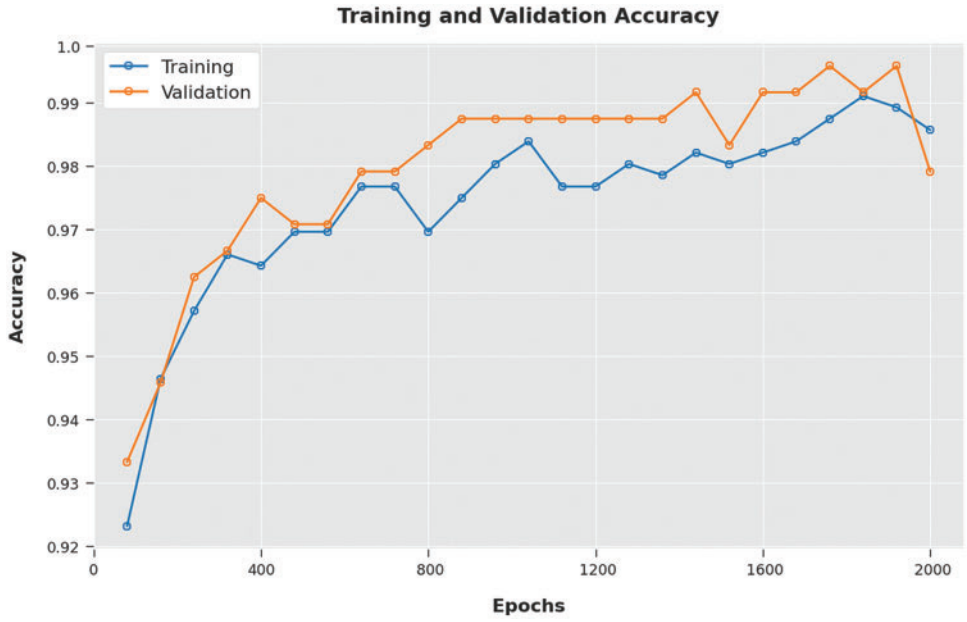

Figure 7: TA and VA analyses results of the SFODTL-AHCR approach

Both Training Loss (TL) and Validation Loss (VL) values, achieved by the proposed SFODTL-AHCR technique on the test dataset, are portrayed in Fig. 8. The experimental outcomes imply that the SFODTL-AHCR algorithm accomplished the least TL and VL values whereas the VL values were lesser than the TL values.

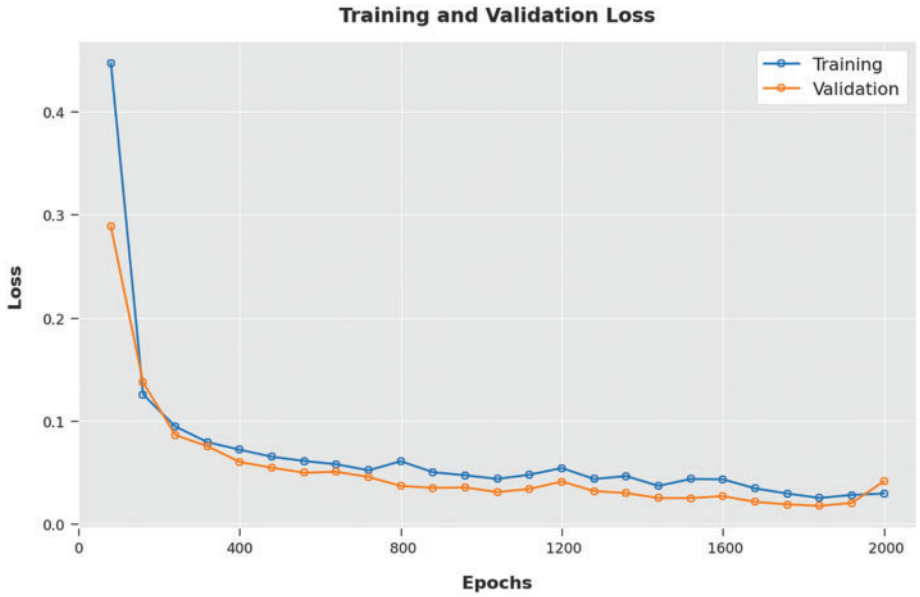

Figure 8: TL and VL analyses results of the SFODTL-AHCR approach

A clear precision-recall analysis was conducted for the SFODTL-AHCR method using the test dataset and the results are shown in Fig. 9. The figure indicates that the proposed SFODTL-AHCR method produced enhanced precision-recall values under all the classes.

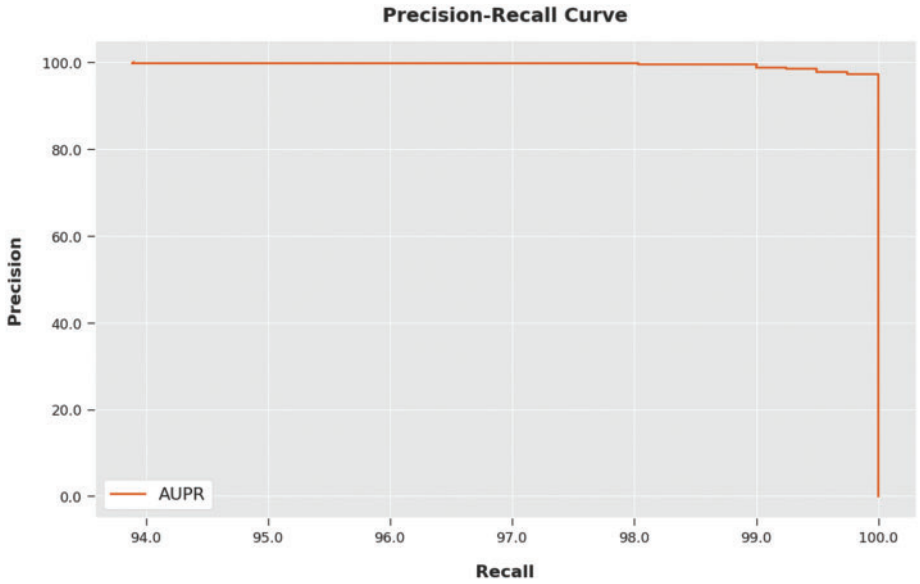

Figure 9: Precision-recall curve analysis results of the SFODTL-AHCR approach

A brief ROC analysis was conducted upon the SFODTL-AHCR method using the test dataset and the results are portrayed in Fig. 10. The results indicate that the proposed SFODTL-AHCR approach established its ability in categorizing the UCF-Sports Action dataset under distinct classes.

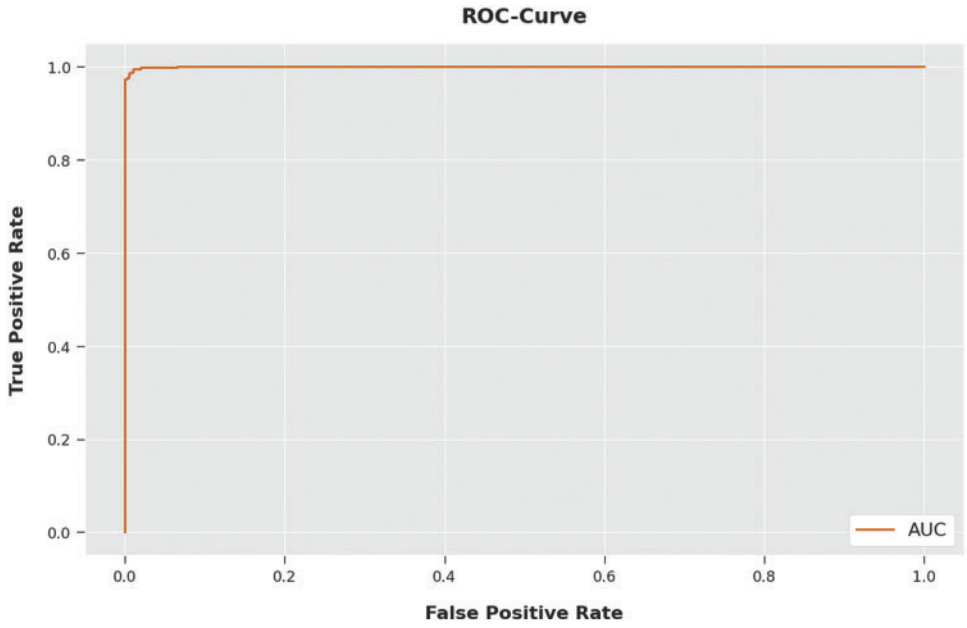

Figure 10: ROC curve analysis results of the SFODTL-AHCR approach

Finally, a brief comparison study was conducted between the proposed SFODTL-AHCR model and other existing models and the results are shown in Table 3. Fig. 11 illustrates the $accu_y$ analysis results achieved by the SFODTL-AHCR model and other existing models on the training dataset. The figure implies that the KNN, RF and the NB models reported the least values of $accu_y$. Followed by, the DT-LDA model attained a somewhat improved $accu_{racy}$ of 97.97%. At the same time, the CNN and the SVM models reported reasonable $accu_{racy}$ values such as 99.60% and 99.56% respectively. However, the proposed SFODTL-AHCR model achieved effectual outcomes with a maximum $accu_{racy}$ of 99.73%.

Table 3: Comparative analysis results of the SFODTL-AHCR approach and other existing methodologies in terms of accuracy

	Accuracy	
Methods	Training Phase	Testing Phase
SFODTL-AHCR	99.73	99.38
CNN Model	99.60	97.51
SVM Model	99.56	99.04
NB Classifier	89.50	93.17
KNN Model	80.28	80.21
RF Model	83.91	84.72
DT-LDA	97.97	75.18

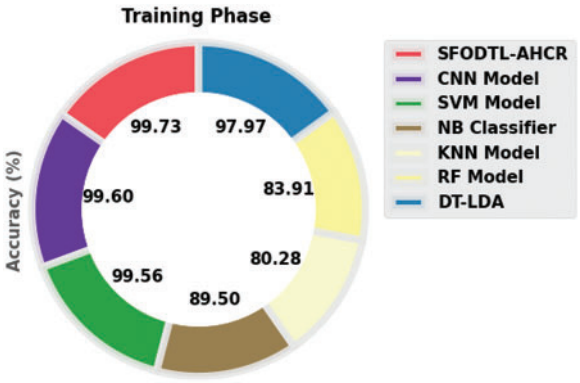

Figure 11: $Accu_y$ analysis results of the SFODTL-AHCR approach on the training dataset

Fig. 12 demonstrates the $accu_y$ analysis results of the proposed SFODTL-AHCR method and other existing models on the testing dataset. The figure denotes that the KNN, RF and the NB models achieved the least values of $accu_y$. Followed by, the DT-LDA model attained a somewhat improved $accu_{racy}$ of 75.18%. Meanwhile, the CNN and the SVM models achieved reasonable $accu_{racy}$ values such as 97.51% and 99.04% correspondingly. But, the proposed SFODTL-AHCR model achieved effectual outcomes with a maximum $accu_{racy}$ of 99.38%.

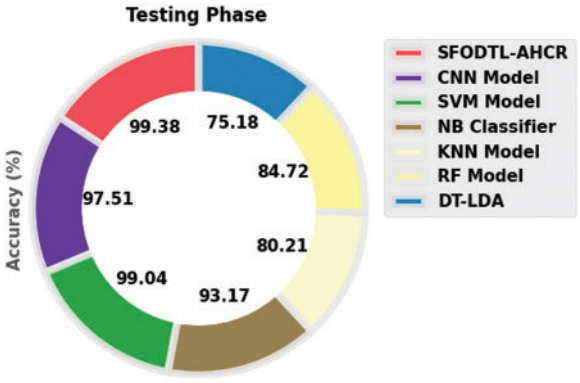

Figure 12: $Accu_y$ analysis results of the SFODTL-AHCR approach on the testing dataset

Table 4 illustrates the CT analysis results achieved by the SFODTL-AHCR model and other existing models [3]. The figure implies that the SVM model achieved a maximum CT of 8.034 s. Then, the CNN, NB, and the KNN models reported slightly low CT values such as 5.039, 6.406 and 6.365 s respectively. Likewise, the RF and the DT-LDA models accomplished reasonable CT values such as 4.072 and 3.498 s respectively. But, the proposed SFODTL-AHCR model produced superior results with the least CT of 3.162 s.

Table 4: CT analysis results of the SFODTL-AHCR approach and other existing approaches

Methods	Computational Time (sec)
SFODTL-AHCR	3.162
CNN Model	5.039
SVM Model	8.034
NB Classifier	6.406
KNN Model	6.365
RF Model	4.072
DT-LDA	3.498

5 Conclusion

In this study, a new SFODTL-AHCR model has been developed for the recognition of the handwritten Arabic characters in the input image. To attain this, the proposed SFODTL-AHCR model pre-processes the input image through the Histogram Equalization approach. Followed by, the Inception with ResNet-v2 model examines the pre-processed image to produce the feature vectors. In order to recognize the Arabic handwritten characters, the DWNN model is utilized. At last, the SFO algorithm is utilized for fine-tuning the parameters of the DWNN model to attain a better performance. The performance of the proposed SFODTL-AHCR model was validated using a series of images. Extensive comparative analyses outcomes confirmed the supremacy of the proposed SFODTL-AHCR model over other approaches. In future, the segmentation approaches can be included in the SFODTL-AHCR model to boost the overall performance of the model.

Funding Statement: The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through Large Groups Project under grant number (168/43). Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R263), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. The authors would like to thank the Deanship of Scientific Research at Umm Al-Qura University for supporting this work by Grant Code: (22UQU4340237DSR32). The author would like to thank the Deanship of Scientific Research at Shaqra University for supporting this work.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] H. M. Balaha, H. A. Ali, M. Saraya and M. Badawy, "A new Arabic handwritten character recognition deep learning system (AHCR-DLS)," *Neural Computing and Applications*, vol. 33, no. 11, pp. 6325–6367, 2021.

[2] H. M. Balaha, H. A. Ali, E. K. Youssef, A. E. Elsayed, R. A. Samak *et al.,* "Recognizing Arabic handwritten characters using deep learning and genetic algorithms," *Multimedia Tools and Applications*, vol. 80, no. 21–23, pp. 32473–32509, 2021.

[3] A. A. A. Ali and M. Suresha, "A new design based-fusion of features to recognize Arabic handwritten characters," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 5, pp. 2570–257, 2019.
[4] N. Altwaijry and I. Al-Turaiki, "Arabic handwriting recognition system using convolutional neural network," *Neural Computing and Applications*, vol. 33, no. 7, pp. 2249–2261, 2021.
[5] F. N. Al-Wasabi, "Proposing high-smart approach for content authentication and tampering detection of Arabic text transmitted via internet," *IEICE Transactions on Information and Systems*, vol. E103.D, no. 10, pp. 2104–2112, 2020.
[6] N. Lamghari, M. E. H. Charaf and S. Raghay, "Hybrid feature vector for the recognition of Arabic handwritten characters using feed-forward neural network," *Arabian Journal for Science and Engineering*, vol. 43, no. 12, pp. 7031–7039, 2018.
[7] F. N. Al-Wasabi, "Entropy-based watermarking approach for sensitive tamper detection of Arabic text," *Computers, Materials & Continua*, vol. 67, no. 3, pp. 3635–3648, 2021.
[8] B. H. Nayef, S. N. H. S. Abdullah, R. Sulaiman and Z. A. A. Alyasseri, "Optimized leaky ReLU for handwritten Arabic character recognition using convolution neural networks," *Multimedia Tools and Applications*, vol. 81, no. 2, pp. 2065–2094, 2022.
[9] B. Rajyagor and R. Rakhlia, "Handwritten character recognition using deep learning," *International Journal of Recent Technology and Engineering*, vol. 8, no. 6, pp. 5815–5819, 2020.
[10] M. S. Amin, S. M. Yasir and H. Ahn, "Recognition of pashto handwritten characters based on deep learning," *Sensors*, vol. 20, no. 20, pp. 5884, 2020.
[11] H. Ali, A. Ullah, T. Iqbal and S. Khattak, "Pioneer dataset and automatic recognition of Urdu handwritten characters using a deep autoencoder and convolutional neural network," *SN Applied Sciences*, vol. 2, no. 2, pp. 152, 2020.
[12] T. Ghosh, M. M. H. Z. Abedin, S. M. Chowdhury, Z. Tasnim, T. Karim *et al.,* "Bangla handwritten character recognition using MobileNet V1 architecture," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 6, pp. 2547–2554, 2020.
[13] A. A. A. Ali, M. Suresha and H. A. M. Ahmed, "A survey on Arabic handwritten character recognition," *SN Computer Science*, vol. 1, no. 3, pp. 152, 2020.
[14] J. H. Alkhateeb, "An effective deep learning approach for improving off-line Arabic handwritten character recognition," *International Journal of Software Engineering and Computer Systems*, vol. 6, no. 2, pp. 53–61, 2020.
[15] M. Salam and A. A. Hassan, "Offline isolated Arabic handwriting character recognition system based on SVM," *The International Arab Journal of Information Technology*, vol. 16, no. 3, pp. 467–472, 2019.
[16] H. Alyahya, M. M. B. Ismail and A. Al-Salman, "Deep ensemble neural networks for recognizing isolated Arabic handwritten characters," *ACCENTS Transactions on Image Processing and Computer Vision*, vol. 6, no. 21, pp. 68–79, 2020.
[17] S. P. Deore and A. Pravin, "Devanagari handwritten character recognition using fine-tuned deep convolutional neural network on trivial dataset," *Sādhanā*, vol. 45, no. 1, pp. 243, 2020.
[18] M. S. Nair and J. Mohan, "Static video summarization using multi-CNN with sparse autoencoder and random forest classifier," *Signal, Image and Video Processing*, vol. 15, no. 4, pp. 735–742, 2021.
[19] L. Yang and H. Chen, "Fault diagnosis of gearbox based on RBF-PF and particle swarm optimization wavelet neural network," *Neural Computing and Applications*, vol. 31, no. 9, pp. 4463–4478, 2019.
[20] Y. Zhang and Y. Mo, "Dynamic optimization of chemical processes based on modified sailfish optimizer combined with an equal division method," *Processes*, vol. 9, no. 10, pp. 1806, 2021.