International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12– No. 43, March 2024 – www.ijais.org

Analysis of ANN Training Algorithms for Hand Geometry-based Access Control

Kazeem B. Adedeji
Department of Electrical and Electronics Engineering, The Federal University of Technology Akure, Ondo State Nigeria

Apena Waliu O.
Department of Computer Engineering, The Federal University of Technology Akure, Ondo State Nigeria

Adu Michael R.
Department of Electrical and Electronics Engineering, The Federal University of Technology Akure, Ondo State Nigeria

ABSTRACT
Hand geometry-based identification systems are one of the most widely used access control systems due to their simplicity and low cost. Extracted features of hand images are deployed to train a machine learning algorithm. The required hand features include palm size as well as finger length and width. During verification, unauthorized hand images are rejected if their features do not match those already stored in the database. Several machine learning algorithms have been employed for access control, but artificial neural network (ANN) techniques are a major contender due to their robustness and accuracy to parameter changes. The performance of the hand geometry technique is dependent on the training model used to assess the efficacy of the ANN. The study examined four ANN training algorithms: the Levenberg-Marquardt algorithm (LMA), BFGS Quasi-Newton (GFGS-QN), resilient back propagation (RBP), and scaled conjugate gradient (SCG) algorithms. The results revealed through numerical investigation showed that the LMA outperformed the rest of the studied ANN algorithms using mean square error, image gradient coefficient, histogram of errors, regression, accuracy, and precision. LMA ranked first with positive outcomes of the lowest mean square error of 8.8383×10^{-5}, 99.999% regression value, and 99.99% accuracy, respectively. The study complements the performance of LMA with the ANN training algorithm at 13 epochs to reach its best performance and its convergence strength. LMA proves to be the most suitable ANN training algorithm for hand geometry recognition applications.

Keywords
Access control, ANN, hand geometry, LMA, BFGS Quasi-newton, RBP, SCG

1. INTRODUCTION
The fight against insurgency is more prevalent due to increased concerns about national security. Many countries around the world are keen to improve security by controlling access to a particular resource. One way to do this is to require biometric verification from everyone using these resources. An autonomous way to identify people based on physiological traits is offered by biometric-based identification systems [1, 2]. In the past, several frameworks for biometric systems have been developed. Among others, the use of fingerprint, iris, voice, signature, gait, keystrokes, and hand geometry recognitions have been reported [3]. The latter is widespread due to the simplicity of the verification procedure. Some features of the human hand, such as shape, palm size, and finger length and width, are extracted using this system. According to reports [1–8], human hands have recognizable features, and at a certain age, a person's hands don't change significantly. Contrary to previous biometric systems, the verification process' accuracy is not compromised during the dry season, when certain people have dry skin [5]. Hand images can be captured with a cheap webcam or digital camera, unlike other biometric qualities that need specialized scanners. This is one of the reasons why the hand geometry-based system is acceptable for access control. In this system, images of the hand are acquired and used to train a machine learning (ML) algorithm.

Several ML algorithms have been used for access control, yet artificial neural network (ANN) techniques are major participants due to their robustness to parameter variations and disturbances [5, 9-11]. Likewise, ANN is well-known for being accurate and able to extract patterns from data, making it a popular ML technique [12]. ANNs use back-propagation or a similar gradient descent algorithm. It is significant to note that the effectiveness of such a system would be influenced by the learning algorithm employed. Increasing the effectiveness of training and learning for neural network-based algorithms is a current research topic. This paper evaluates the performance of four ANN training algorithms with hand geometry-based recognition applications. The assessed algorithms are the Levenberg-Marquardt algorithm, the BFGS quasi-newton algorithm, resilient back propagation, and scaled conjugate gradient algorithms. The performance of these algorithms is assessed using mean square error, image gradient, error histogram, regression, accuracy, and precision.

2. PAGE SIZE BIOMETRIC ACCESS CONTROL: CONCEPT AND REVIEW OF RELATED STUDIES
A conventional biometric access control system (BACS) is a pattern recognition unit that acquires specific types of biometric data from an individual. Thereafter, it focuses on the relevant features of such data and compares the features to a pre-set group of attributes in its database. Based on the compared attributes, a decision can then be made about whether to grant or deny access to such an individual. Biometric identification that can be used for access control includes Irish, hand geometry, voice, retina, keystroke, and electrocardiogram (ECG), among others. Figure 1 shows the general working principle of a biometric system. The system starts with the enrolment stage, where it captures the biometric feature(s), processes the biometric form, and stores it in a database. The same process goes through the verification stage, and the

system compares both stages to authenticate the user. A standard BACS is composed of a sensory device, feature extraction, feature comparison, a matching unit, and a database, as illustrated in Figure 1. The sensory device is used for data capture (a biometric reader or image scanner).

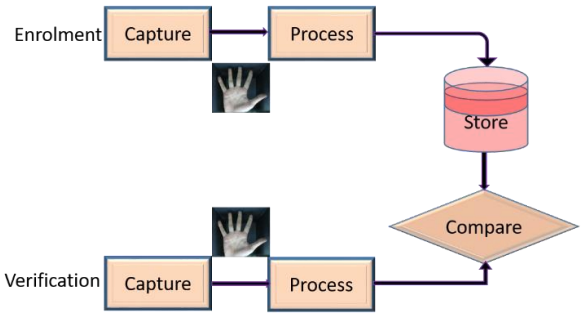

Fig 1: A conventional biometric process.

For hand geometry recognition, the following multi-modal features (fusion of both palm and fingers) of the hand are considered: the finger width, finger length, and sub-length; the palm width and length; the hand length; the hand contour length; and the hand area, as shown in Figure 2. These features are extracted from the captured image during processing and used to train a machine learning algorithm. Thus, the performance of the whole system is dependent on the accuracy of the machine learning algorithm used to train it. As previously mentioned, several ML algorithms have been utilized for BACS. According to Igel and Husken [13], they performed an empirical analysis on an improved Rprop learning algorithm. The study compares the improved Rprop to other variants such as iRprop, Quickprop, and conjugate gradient algorithms. It is obvious from the findings that the improved Rprop performed better than the other algorithms studied.

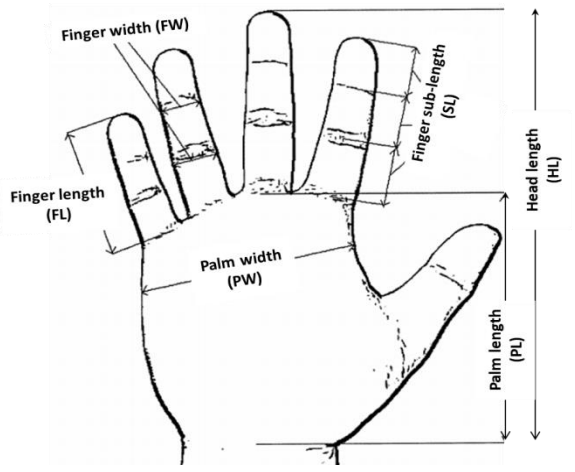

Fig 2: Hand geometry features

Nanni et al. [14], combine biometric matches using ML and statistical approaches with the FVC2006 dataset. The results obtained indicated that the fusion of different fingerprint matches can lead to a significant result when compared to a single match. In Wu et al. [15], pattern identification in finger-vein was developed using principal component analysis and a neural network (NN). Raji et al. [16] proposed an ANN-based biometric prediction system using the respiratory pattern. The proposed system has a classification accuracy of 98% and the network learns past 40 epochs before stagnation occurs when a learning rate of 0.5 is used. In Siswanto et al. [17], an ANN based contactless hand recognition system with a relative geometric parameter was developed. The developed system achieved 87% accuracy with a precision of 86% when tested on the hand image dataset.

The use of ANN for signature recognition and verification, Irish recognition, and ECG signals has been reported in [17–21]. Salve and Narote [18] use support vector machines (SVM) and ANN for Irish recognition. In Patro et al. [19], an approach for biometric recognition via ECG was developed. In this study, three ML algorithms (ANN, SVM, and K-nearest neighbour) were used as classifiers. An improved machine learning-based biometric identification system based on RR-interval-framed electrocardiography was proposed by Kim et al. [20]. A fusion of biometric identification traits has also been used for access control, as may be found in [22–24]. In Abozaid et al. [22], a Gaussian mixture model (GMM), ANN, and SVM were used for voice and face recognition. Similarly, Thamaraimanalan et al. [23] use SVM and ANN classifiers for biometric access, while the use of ANN for face and fingerprint for biometric access is reported in [24]. The use of ANN for the implantation of keystroke pressure typing-based BACS may also be found in [25–27]. For instance, Ali and Salami [25] proposed a keystroke pressure-based scheme by hybridizing ANN and ANFIS-based classifiers. When tested on a keystroke pressure-based typing dataset, the system performed relatively better. In Harun et al. [26], the ANN and distance classifier methods were utilized to analyse the effectiveness of a keystroke authentication system. Likewise, in Harun et al. [27], a multilayer perceptron was utilized to assess the effectiveness of a keystroke biometric authentication. Several machine learning algorithms have been used for access control, with ANN techniques playing a significant role.

3. METHODOLOGY

This framework involved examining the performance of some ANN training algorithms for hand geometry recognition in access control systems. The hand geometry verification system is shown in Figure 3. It has three main parts: enrolment, identification, and verification. In the enrolment phase, hand images are pre-processed to generate numerical features, which are then enrolled with an ID in the identification phase and trained using an artificial neural network. The verification phase compares the newly-input data to the data that has already been stored in the database. The photos of hand images served as the input parameters to the system. These photos were obtained from different individuals and stored in the Joint Photographic Experts Group format for processing. The images went through a pre-processing phase, which included conversion to grayscale images, noise removal, and edge detection. The process of converting an image to grayscale preserves the brightness information while greatly reducing the saturation. Background noise was removed to prevent differences between the real and the captured images. The edge detection permits image's edges to extract the hand's geometric features. The edge detection technique is used to locate areas of brightness discontinuities in digital images.

Some of the features extracted from the hand photos are illustrated in Figure 2. The features were used to train four algorithms separately in MATLAB version 2019b environment. The performance of each algorithm was evaluated based on the mean square error, image gradient, error histogram, regression, accuracy and precision. Thus, an overview of the operations involved in some of these algorithms is first presented.

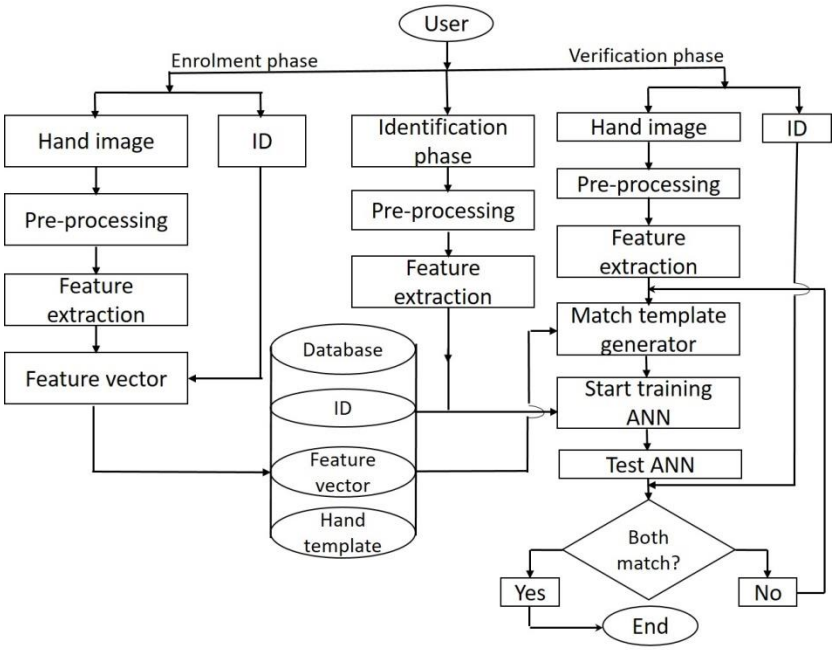

Fig 3: The hand geometry system.

3.1 Overview of the ML Algorithms

This section presents the overview of the ANN training algorithms and the performance metrics. The assessed algorithm is illustrated in Figure 4 are Levenberg-Marquardt algorithm (LMA), BFGS Quasi-newton (GFGS-QN), resilient back propagation (RBP), and the gradient method (GM) using the scaled conjugate gradient (SCG) algorithm.

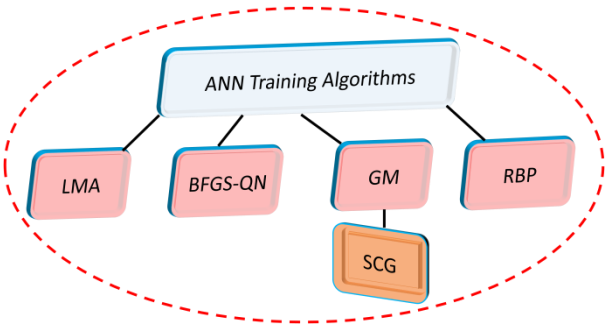

Fig 4: The ANN algorithms investigated

3.1.1 *Levenberg-Marquardt Algorithm (LMA)*

The LMA was developed to work with a loss function [28, 29]. The algorithm works with the computation of the gradient vector of the Jacobian matrix without finding the exact Hessian matrix. Figure 5 gives a summary of the process involved in the LMA. Firstly, the loss gradient and Hessian approximation are estimated, as illustrated in Figure 5. The damping parameter is then changed to reduce losses throughout the iteration.

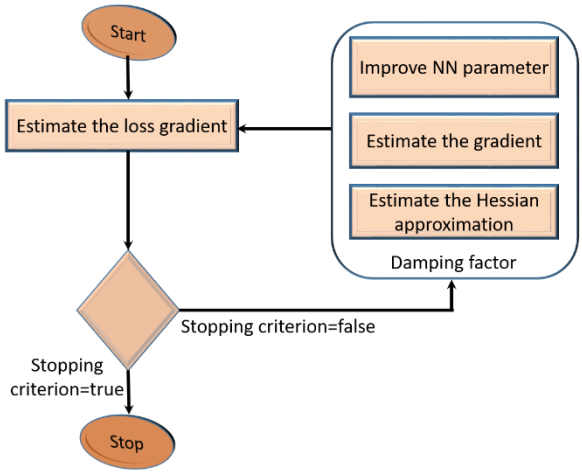

Fig 5: The summary of the process involved in the LMA

Considering a loss function f which takes the form of a squared errors e,

$$f = \sum_{i=1}^{m} e_i^2 \tag{1}$$

where m is the number of training samples. The Jacobian matrix $\boldsymbol{J}$ of f contains the derivatives of e with respect to the weight and biases in the neural network. Thus, $\boldsymbol{J}$ is expressed as

$$J_{i,j} = J \in \Re^{(m \times n)} = \frac{\partial e_i}{\partial w_j} \quad \begin{cases} i = 1,.....,m \\ j = 1,.....,n \end{cases} \tag{2}$$

In Eq. 2, m denotes the number of samples and n denotes the number of parameters in the neural network. The gradient vector of the loss function ∇f , is estimated as

$$\nabla f = 2J^T e \tag{3}$$

where $\boldsymbol{e}$ is the vector relating to the error terms. Finally, the approximate Hessian matrix $\boldsymbol{H}$ is computed from

$$Hf \approx 2J^T J + \lambda I \tag{4}$$

In Eq. 4, λ is a scaler quantity that denotes the damping factor, and I is an identity matrix. The essence of the damping factor is to ensure that $\boldsymbol{H}$ remain positive. Thus, the parameter improvement process in the LMA is estimated as [30].

$$w^{(i+1)} = w^{(i)} - \left(J^{T(i)}.J^{(i)} + \lambda^{(i)}I\right)^{-1}.\left(2J^{T(i)}e^{(i)}\right) \quad (5)$$

The values of λ can be zero or relatively large. In a situation where λ is zero, the algorithm is the same as the Newton approach, using the approximate Hessian matrix. However, with large λ, the algorithm is the same as the gradient descent approach with a small learning rate. The application of LMA to neural network training may be found in [31]. In MATLAB, the function "trainlm" was used for the LM algorithm.

3.1.2 Resilient Backpropagation (RBP)

RBP is a supervised learning strategy for feedforward artificial neural networks that is mostly utilized for pattern recognition [32]. The RBP uses the derivative's sign to determine the weight update's direction. Suppose w_{ij} denotes the weight via neuron j to i and E represents an arbitrary error measure that can be differentiated with respect to the weights. In backpropagation, the change in weight may be estimated using (6) [33].

$$\Delta w_{ij}(t) = \alpha \times x_i(t) \times \delta_j(t) \quad (6)$$

In Eq. 6, α is the learning rate, $x_i(t)$ represents the inputs propagating back to neuron i under step t, while δ denotes the error gradient. In RBP, Δ_{ij}, is estimated for each connection. This evaluates the size of the weight update. Thus, the computation of Δ_{ij} [12, 33] is valid for;

$$\Delta_{ij} = \begin{cases} \eta^{+} \times \Delta_{ij}^{(t-1)} & if \quad \dfrac{\partial E^{(t-1)}}{\partial w_{ij}} \times \dfrac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ \eta^{-} \times \Delta_{ij}^{(t-1)} & if \quad \dfrac{\partial E^{(t-1)}}{\partial w_{ij}} \times \dfrac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{(t-1)} & Otherwise \end{cases}$$

$$0 < \eta^{-} < 1 < \eta^{+} \quad (7)$$

The value of Δ_{ij} is updated based on the sign of $\partial E^{(t-1)} / \partial w_{ij}$ for previous iteration, and $\partial E^{(t)} / \partial w_{ij}$ for current iteration. Each time the sign of the error gradient of the corresponding w_{ij} changes, the updated value of Δ_{ij} is reduced by η-. A value of 0.5 has been reported [33].

3.1.3 Scaled conjugate gradient

This is not the same as other gradient algorithms that perform line search for each iterations. It is based on conjugate directions; thus, no line search is required during each iteration [32, 34, 35]. In MATLAB, the function "trainscg" can be used to run the SCG algorithm. It is a network training function that modifies the SCG framework's weight and bias settings. With this function, any network may be trained once the derivative functions of its weight and net input exist. In the SCG algorithm, the step size, which can be determined by a variety of approaches, is a function of the quadratic approximation of the error function [32]. The second-order term is calculated as [32].

$$\bar{s}_k = \frac{E^{'}(\overline{w}_k + \sigma_k \bar{p}_k) - E^{'}(\overline{w}_k)}{\sigma_k} + \chi_k \bar{p}_k \quad (8)$$

In Eq. 8, χ_k is a scalar quantity which is adjusted based on σ_k . The step size is estimated using Eq. 9 [32].

$$\alpha_k = \frac{\mu_k}{\sigma_k} = \frac{-\bar{P}_j E^{'}{}_{qw}(\bar{y}_i)}{\bar{P}_j^T E^{''}(\overline{w})\bar{P}_j} \quad (9)$$

where $\overline{w}$ denotes the weight vector in space, $E(\overline{w})$ is the global error function, $E^{'}(\overline{w})$ is used to represent gradient of errors, and $E^{'}_{qw}(\bar{y}_i)$ denotes the quadratic approximation of the error function. If $\bar{p}_1$,....., $\bar{p}_k$ is used to represent a set of non-zero weight vector, λ_k is adjusted [32] such that;

$$\bar{\lambda}_k = 2\left(\lambda_k - \frac{\sigma_k}{|\bar{p}_k|^2}\right) \quad (10)$$

A more detailed description of this algorithm may be found in [32, 34].

3.1.4 BFGS Quasi-Newton (BFGS-QN) Algorithm

The Broyden, Flecther, Goldforth and Shanno (BFGS) algorithm is a local search optimization algorithm. Conventional Newton's method is computationally intensive due to the fact that several operations are required to estimate the Hessian matrix together with its inverse. In the quasi-Newton approach, an approximation to the inverse Hessian matrix is established by determining the first partial derivatives of the loss function [36–38]. Thus, this is achieved by approximating the Hessian matrix with a positive definite matrix B, which is updated during iterations from previous steps. During iteration k, the Hessian approximation B must satisfy the quasi-Newton conditions given as [38, 39].

$$B_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k) \quad (11)$$

This is obtained from first order Taylor expansion of $\nabla f(x_k)$. The BGFS Quasi-newton algorithm [39, 40] is described as:

BFGS Quasi-Newton Algorithm

1. Given an initial guess x_0 and an approximate Hessian matrix B_0,
2. Start
3. Obtain a direction P_k by solving $B_k P_k = -\nabla f(x_k)$
4. Conduct 1-dimensional line search to determine step size α_k in the direction from 3.
   if $\alpha_k == \alpha_k$
   $\alpha_k = \arg\min f(x_k + \alpha P_k)$
   else $\alpha_k \neq\neq \alpha_k$ % perform step 4 again until acceptable step size is found.
5. Set
   $s_k = \alpha_k P_k$ and update
   $x_{k+1} = x_k + s_k$
6. $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
7. $B_{k+1} = B_k + \dfrac{y_k y_k^T}{y_k^T s_k} - \dfrac{B_k s_k s_k^T B_k^T}{s_k^T B_k s_k}$
8. *end if*
9. stop

3.2 Algorithm Evaluation

This section presents the overview of the ANN training algorithms and the performance metrics. The assessed algorithm is illustrated in Figure 4 are Levenberg-Marquardt algorithm (LMA), BFGS Quasi-newton (GFGS-QN), resilient back propagation (RBP), and the gradient method (GM) using the scaled conjugate gradient (SCG) algorithm.

3.2.1 Captured hand image samples

The dataset consists of images of the hand geometry collected from the palm print database of the College of Engineering, Pune. It contains different with a resolution of 1600 by 1200 pixels. The database consists of a total of 1344 hand images pertaining to 168 people, collected over a period of one year. Each algorithm was trained using 70% of the hand images. The remaining 30% was divided into two; 15% was used for testing and another 15% for validation. Figure 6 shows samples of some of the hand images stored in a database.

Fig 6: Some of the acquired hand images.

ANN learning is obtained by updating the weights along the network in consecutive iterations of the feed-forward and back propagation processes. Training was done by back-propagating the error in the weights and biases using the four algorithms. The effectiveness of these algorithms was analysed using the metric discussed in the next section.

3.2.2 Performance metrics

(i) *Mean square error* (*MSE*): In statistical modelling, the *MSE* measures the average of the square of the errors. It is used to determine the degree to which a model fits the data well. A small *MSE* will produce the best fit. If a vector of *n* predictions is obtained from a sample of *n* data points, and Y_i and $\hat{Y}_i$ denote the vectors of the observed and predicted values, respectively, the *MSE* is computed as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{12}$$

The *MSE* may be written in matrix form as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(e_i\right)^2 = \frac{1}{n}e^T e \tag{13}$$

where e_i is $\left(Y_i - \hat{Y}_i\right)^2$ and $e \in \Re^{(n\times 1)}$ is a single row error matrix.

(ii) *Image gradient:* This is a directional change in the intensity of an image. Conventionally, using the Sobel filter method, an original image may be convolved with a filter to generate an image gradient. The gradient of an image can be described as a vector of its partials [41, 42].

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{14}$$

where $\frac{\partial f}{\partial x}$ is the derivate with respect to *x* (that is, gradient in the *x*-direction) and $\frac{\partial f}{\partial y}$ is that in the *y*-direction. Applying a 1-dimensional convolution to image *A*,

$$\frac{\partial f}{\partial y} = \begin{bmatrix} -1 \\ +1 \end{bmatrix} * A \tag{15}$$

where * is used to represent the 1-dimensional convolution operation. The expressions in (15) [40, 41] is a (2×1) filter which shift the image by half a pixel. The image gradient was computed using Sobel filter method.

(iii) *Histogram of errors*: This shows how the errors from the NN on the testing instance are distributed. The application of the error histogram plot for analysing the performance of algorithms may be found in [43].

(iv) *Regression*: The regression plot gives an idea of how close the output from the model is to the actual target values. The regression values were computed during the training, validation, and testing, respectively. Also, this value was also computed when the training, validation, and testing where considered together.

(v) *Accuracy*: This is a measure of the percentage of accurate predictions across all the instances analysed. It can be estimated as

$$p = \frac{T_p + T_N}{T_p + F_p + T_N + F_N} \tag{16}$$

In Eq. 16, T_N stands for true negative, F_N denotes false negative, T_P signifies the true positives while F_P represents the false positives.

(vi) *Precision*: The precision (*p*) is an estimate of the percentage of positive patterns in a positive class that are correctly predicted. It is estimated using Eq. 17.

$$p = \frac{T_p}{T_p + F_p} \tag{17}$$

4. EMPIRICAL RESULTS

4.1 MSE Analysis

Figure 7 illustrates how the *MSE* changes by epoch for the algorithm training, validation, and testing.

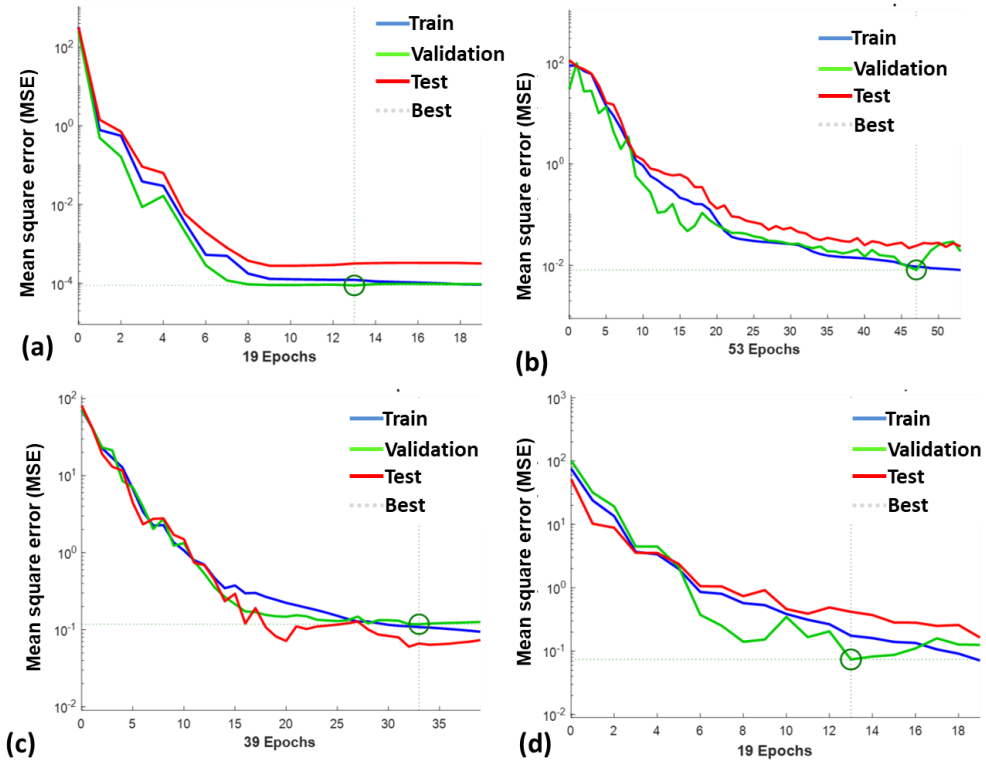

Fig 7: Mean square error plot for (a) LMA (b) BFGS-QN (c) RBP and (d) SCG algorithms.

This figure shows the *MSE* by epoch due to the LMA (Figure 7(a)), BFGS-QN (Figure 7(b)), RBP (Figure 7(c)), and SCG (Figure 7(d)) algorithms. For the training phase, a *MSE* of 1.25×10^{-4} was achieved at 10 iterations as shown in Figure 7(a). An excellent validation performance of 8.8383×10^{-5} was observed at 13 epochs. The training is terminated in the default setup after six successive upswings in validation error; as a result, the top performance is chosen from the epoch with the minimum validation error. The validation error is minimum at the 13th epoch. After this phase, the *MSE* becomes at 9.25×10^{-5} at 19 epochs, with the top validation performance of 8.8383×10^{-5} at 13 epochs. Considering Figure 7(b), for the BFGS-QN algorithm, a *MSE* of 4.67×10^{-1} was obtained at 35 iterations during the training phase, with the top performance of 8.1728×10^{-3} at epoch 47 where its minimum validation error occurs. Beyond the training phase, the *MSE* becomes 8.11×10^{-3} at epoch 53 epochs with top performance of 8.1728×10^{-3} at epoch 47. Unlike the LMA, its minimum validation error occurs at the epoch 47, as opposed to the 13 epochs for the LMA. Considering Figure 7(c), for the RBP algorithm, a *MSE* of 1.87×10^{-1} was obtained at 25 iterations during the training phase, with the top performance of 1.1789×10^{-3} at 33 epochs where its minimum validation error occurs. Unlike LMA and BFGS-QN algorithms, the best performance (with the lowest validation error) occurs at epoch 33. It took the LMA and BFGS-QN 19 and 47 epochs, respectively, to achieve the minimum validation error. After the training, the *MSE* becomes 2.45×10^{-1} at 39 epochs, with a best performance of 1.1789×10^{-3} at epoch 33. In Figure 7(d), for the SCG algorithm, a MSE of 1.567×10^{-1} was achieved after 10 iterations during the training phase, with the best validation performance of 7.3919×10^{-2} at 13 epochs. Like the LMA, this algorithm has the minimum validation error at epoch 13. Unlike the LMA and BFGS-QN algorithms, the best performance (with the minimum validation error) occurs at epoch 33. It took the LMA and BFGS-QN 19 and 47 epochs, respectively, to achieve the minimum validation error. After the training, the *MSE* becomes 7.12×10^{-2} at 19 epochs, with the best performance of 7.3919×10^{-2} at 13 epochs. In comparison with the other three algorithms, it took the SCG algorithm 13 epochs to achieve the minimum validation error (best performance). While this performance is closely similar to that of the LMA, the LMA, however, has a relatively smaller *MSE* at 13 epochs. For the four algorithms, the *MSE* variations with epoch follow a similar trend. The *MSE* reduces as the number of epochs rises up to a point where the validation error becomes lowest. The validation graph tends to become more error-prone when the epoch is raised above the point of best performance, which causes the data to become over fitted. Comparing the potential of the algorithms, the results revealed that the LMA and SCG algorithm converge faster than the other algorithms. However, in terms of the *MSE*, the LMA has superior performance, followed by the BFGS-QN algorithm, while the RBP algorithm has the least performance. The overall assessment of these algorithms shows the superiority of LMA, considering both speed and error level.

4.2 Analysis of the Histogram of Errors

In Figure 8, the histogram of errors for the four algorithms was presented. As can be seen, "bins" refers to the number of vertical bars on the graph. The total error range is divided into 20 smaller bins in this instance. The y-axis shows how many samples from the dataset fit into each bin, and the zero-error line lines up with the zero-error value on the x-axis. As shown in Figure 8(a) for the LMA, the highest error instances of 1.907×10^{-3} were obtained during the testing phase, with 9.187×10^{-3} error instance during validation.

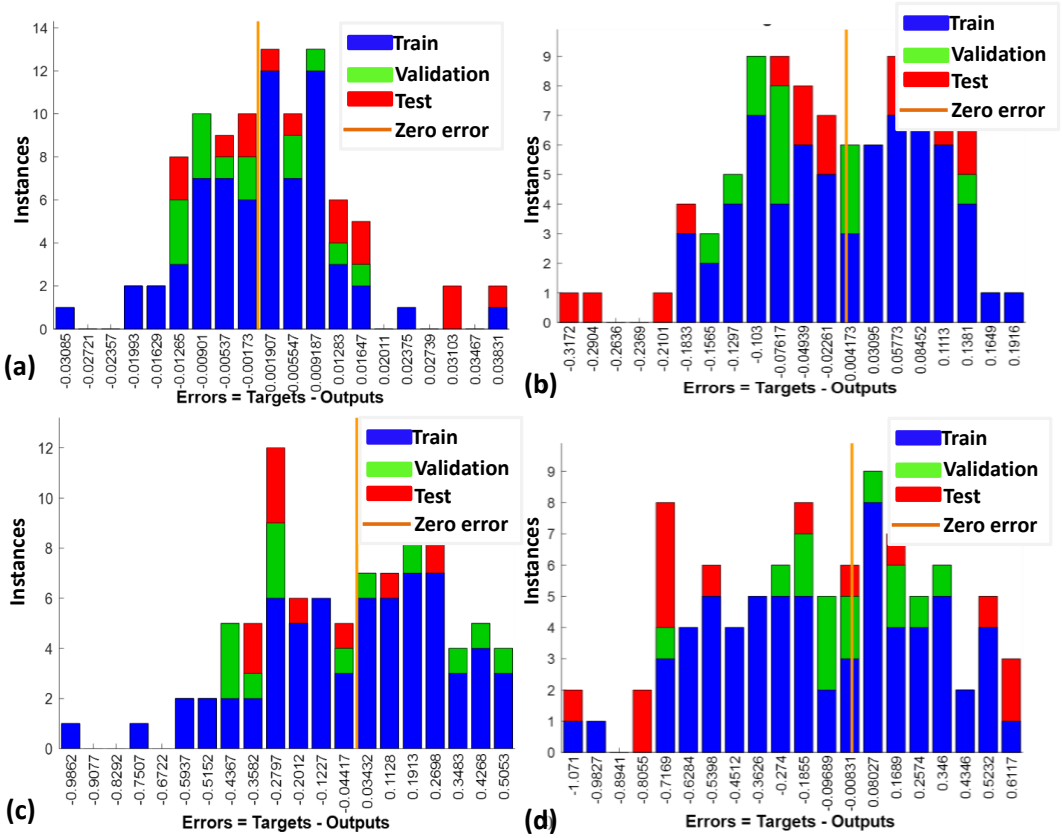

Fig 8: Error histogram plot with 20 bins for (a) LMA (b) BFGS-QN (c) RBP and (d) SCG algorithms.

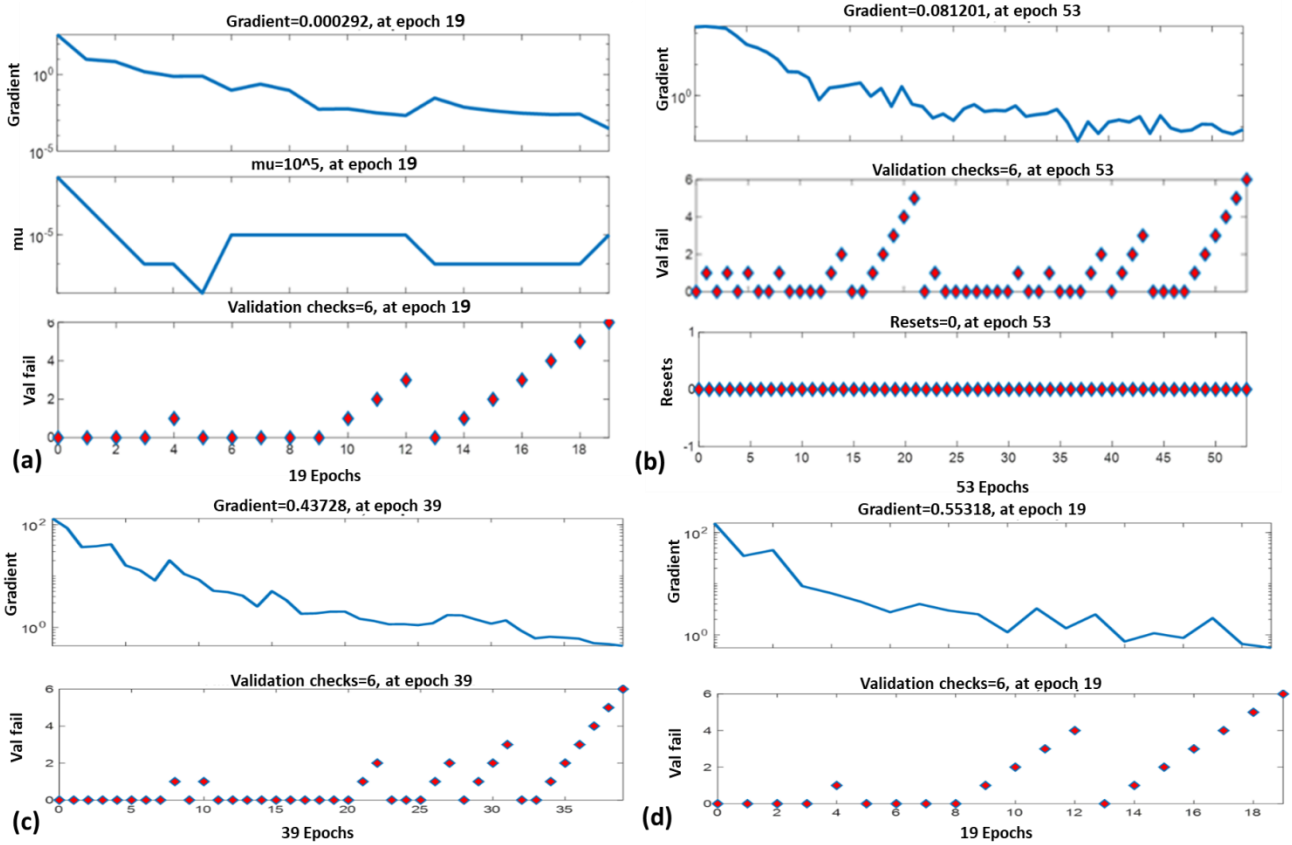

Fig 9: Image gradient plot for (a) LMA (b) BFGS-QN (c) RBP and (d) SCG algorithms.

Considering the BFGS-QN algorithm (Figure 8(b)), the testing stage has the highest error around -0.0761 and 0.05773, the validation stage has the highest instances at -0.103, and the training stage has the highest instances at error -0.103, 0.05773 and 0.08452. For the RBP algorithm (Figure 8(c)), it is observed that the testing stage has the highest error around -0.2792 and highest number of instances at an error of 0.2689. The validation stage has the highest instances of error at -0.4367, while the training stage has the highest instances of error at 0.2698 and 0.1913. For the SCG algorithm (Figure 8(d)), the testing stage has the highest error around -0.2792 and has the highest number of instances at the error of 0.2689, the validation stage has the highest instances at error -0.4367, and the training stage has the highest instances at error 0.2698 and 0.1913. The overall assessment indicates that LMA has the lowest error among the other algorithms, which supports the earlier claim.

4.3 Image Gradient Analysis

The plot of the image gradient illustrated in Figure 9 shows the variations of the gradient coefficient with the number of epochs. Considering the LMA (Figure 9(a)), at epoch 19, the final value of the gradient coefficient is 2.82×10^{-4}, which is close to zero. It is important to mention that the algorithm has better performance when the gradient coefficient is relatively low. The gradient value decreases as the number of epochs increases in the figure; however, this is not the case for validation. It was observed that as the period reached 19, the error began to rise due to over fitting. In Figure 9(b), for the BFGS-QN algorithm, at epoch 53, the final value of the gradient coefficient is 8.12×10^{-2}, the validation check is 6 at epoch 53, and resets are equal to 0 at epoch 53. For the RBP algorithm (Figure 9(c)), the final value of the gradient coefficient is 4.3728×10^{-1} at epoch 39, and the validation check is equal to 6 after epoch 39. Moreover, for the SCG algorithm (Figure 9(d)), the final value of the gradient coefficient is 5.55318×10^{-1} at epoch 19, while the validation check is equal to 6 after epoch 19 (similar to that of the LM algorithm). For the LMA and SCG algorithms, it takes only 19 epochs to reach a validation check of 6, whereas 39 and 53 epochs are required to achieve the same validation check for the RBP and BFGS-QN algorithms, respectively. In line with the results presented in Figure 7 and Figure 8, the overall assessment still shows the superiority of the LMA using this metric.

4.4 Accuracy and Precision Analysis

Figure 10 shows the accuracy and precision results for the four algorithms and also confirms the superiority of LMA over the other algorithms investigated.

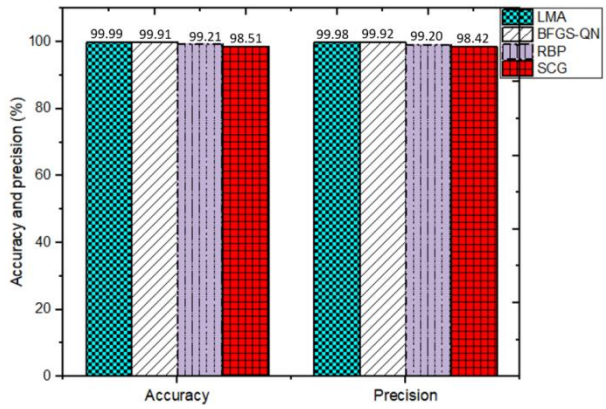

Fig 10: Accuracy and precision results for the four algorithms

The LMA is the most accurate model with an accuracy of 99.99%, while the BFGS-QN is second on the list with 99.91% accuracy. Out of the four algorithms, SCG records the lowest performance. In terms of the precision results, the LMA also recorded an exceptional result with a precision of 99.98%, while the SCG had the lowest precision result. As a result of the fundamental performance metrics in comparison to other algorithms, the LMA appears to be the best, while BFGS-QN appears to be the second best.

4.5 Regression Analysis

Figure 11 shows the linear regression plots for the algorithms. This plot is used to find the relationship between the predicted and actual values. The "target" values in the plot are the actual dataset, whereas the "output" is the predicted values obtained from each of the algorithms. Conventionally, the linear regression value ranges between 0 and 1. A value of 0 indicates a terrible model, while a value of 1 denotes a perfect model. As can be observed in Figure 10, there is a good agreement between the predicted values and the actual ones, which indicates that the algorithms provide a performance where the predicted values perfectly match the target values. Analyzing the regression plot for the LMA (Figure 11(a)), during the training phase, the regression value of 0.9999 was obtained, while the values for validation and testing are 1 and 0.99998, respectively. Considering the three processes (training, testing, and validation), the regression value amounts to 0.99999. This gives an average regression percentage of 99.99%. This value is extremely close to the 100% benchmark for excellent performance that any algorithm could achieve in terms of the linear regression metric. For the BFGS-QN algorithm (Figure 11(b)), the training dataset achieved a regression value of 0.99947, 0.99921 during validation, 0.99891 during testing, and 0.99932 for the whole process, which amounts to an average regression percentage of 99.93%. Considering the regression analysis of the RBP algorithm (Figure 11(c)), it may be observed that during the training process, a regression value of about 0.99408 is obtained. Also, values of 0.99172 and 0.99493 were obtained during the validation and testing, respectively. For the entire process, the regression value is 0.99384, which gives an average regression percentage of 99.38%. For the SCG algorithm (Figure 11(d)), this algorithm gives the least performance in terms of regression. Unlike the other three algorithms, a regression percentage of 98.93% is observed. Comparing these algorithms, LMA is the perfect model among the ANN training algorithms investigated.

In Table 1, a summary of the performance of the four ANN training algorithms is presented. The LMA has the lowest epoch for training and the best performance of 8.8383×10^{-5}. It also has the least *MSE*. The LMA outperformed the other three algorithms used in training the ANN. Also, both the LMA and SCG algorithm require the fewest number of epochs to attain their best performance, making them relatively faster than the RBP and BFGS-QN algorithms. In terms of error performance, BFGS-QN performed better than RBP and SCG and next to LMA. All the algorithms have regression values closer to 1, but LMA provides the best fit. As a result, LMA is the best fit when choosing ANN for hand geometry recognition applications.

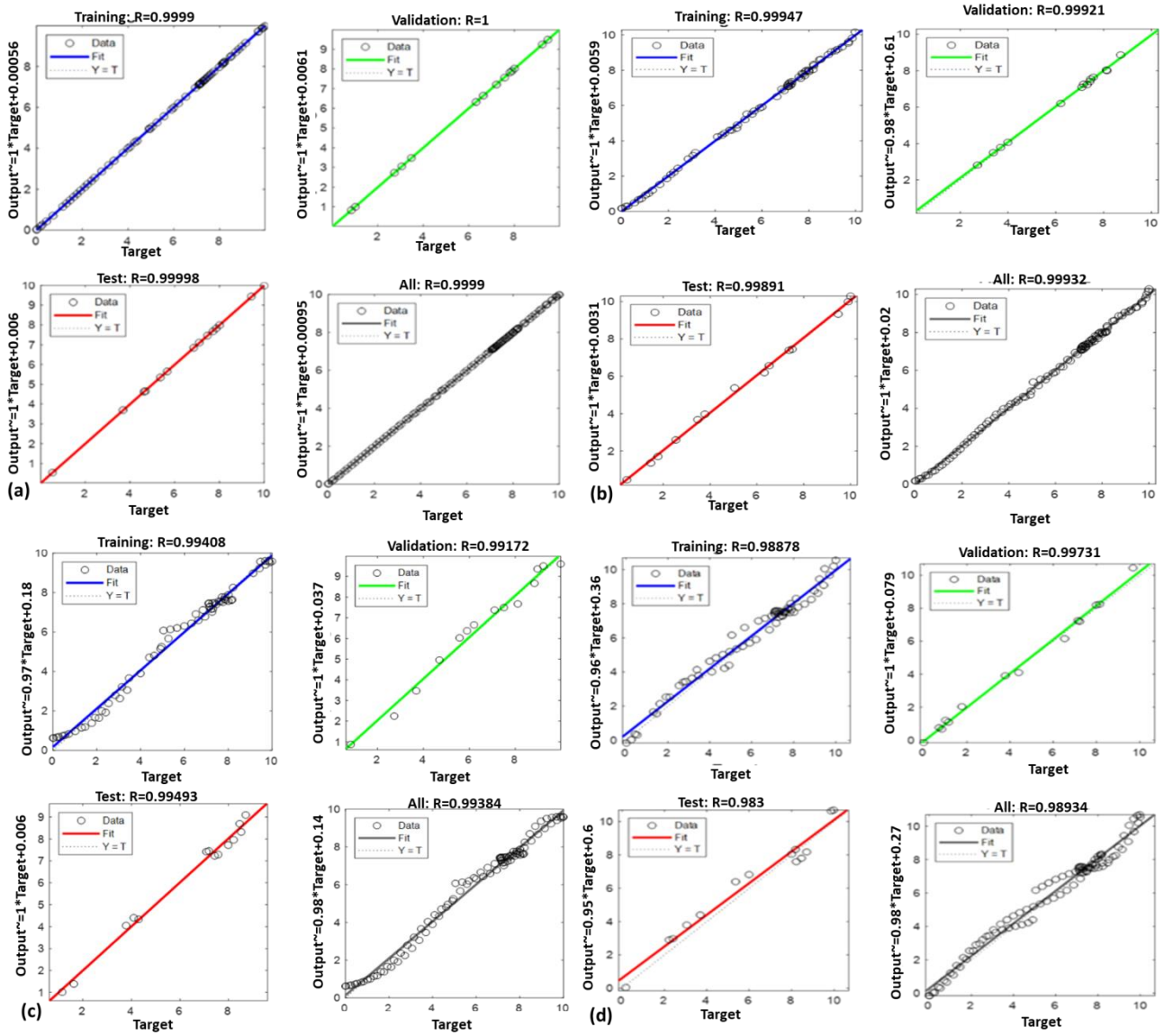

Fig 11: Linear regression plot for (a) LMA (b) BFGS-QN (c) RBP and (d) SCG algorithms.

Table 1. Compared performance of the studied algorithms.

Algorithms	Total epochs	MSE	MSE at best performance	Number of Epoch for best performance	Image gradient	Regression percentage
LMA	19	9.25×10^{-5}	8.8383×10^{-5}	13	2.82×10^{-4}	99.999
BFGS-QN	53	8.11×10^{-3}	8.1728×10^{-3}	47	8.1201×10^{-2}	99.93
RBP	39	2.458×10^{-1}	1.1789×10^{-1}	33	4.3728×10^{-1}	99.38
SCG	19	7.12×10^{-2}	7.39191×10^{-2}	13	5.55318×10^{-1}	98.93

5. CONCLUSION AND FUTURE WORK

The paper investigates the performance of four different ANN training algorithms for hand geometry recognition-based access control applications. In this study, a total of 1344 hand images pertaining to 168 people were used. These photos were obtained from different individuals and stored in the Joint Photographic Experts Group format for processing. The hand images are pre-processed to generate numerical features which are extracted to train a machine learning algorithm. The extracted hand features pertain to the palm size as well as finger length and width. The performance of the system is evaluated

based on the MSE, image gradient, histogram of errors, regression, accuracy and precision. The results obtained showed that the Levenberg-Marquardt algorithm performed better than the other three algorithms chosen. In general, the LMA has the least mean square error and the lowest number of epochs to attain the best performance. The LMA seems to be the fastest algorithm when training medium-sized feed-forward neural networks. All the algorithms have regression values closer to 1, but the LMA provides the best fit and is thus more accurate than the three other algorithms. Therefore, LMA proves to be the most suitable ANN training algorithm for hand geometry recognition applications. In this study, it can be observed that right hand images of the individual were scanned and used for processing. Future study will be to verify the use of a combination of right and left hand images on the system performance. Also, the performance of the system will be evaluated using various dataset or scenarios where the palm structure is affected due to injuries.

6. ACKNOWLEDGMENTS

The authors wish to appreciate the management of the Federal University of Technology Akure for the opportunity given to conduct this research.

7. REFERENCES

[1] Dutagaci, H., Sankur, S., and Yörük, E. 2008. A comparative analysis of global hand appearance-based person recognition. Journal of Electronic Imaging, 17(1), 011018.

[2] Pereira, T.M., Coneicao, R.C., Sencadas, V., and Sebastiao, R. 2023. Biometric recognition: A systematic review on electrocardiogram data acquisition methods. Sensors, 23(3), 1-31.

[3] Adedeji, K.B., and Esan, O.A. 2022. A GUI-based peg-free hand geometry recognition for biometric access control using artificial neural network. Journal of Engineering Advancements, 3(04), 131-143.

[4] Angadi, S., and Hatture, S. 2018. Hand geometry based user identification using minimal edge connected hand image graph. IET Computer Vision, 12, 744-752.

[5] Dhole, S.A., and Patil, V.H. 2012. Person identification using peg free hand geometry measurement. International Journal of Engineering Science and Technology, 4, 2942-2949.

[6] Sahoo, J.P., Sahoo, S.P., Ari, S., and Patra, S.K. 2023. DeReFNet: Dual-stream dense residual fusion network for static hand gesture recognition. Displays, 77, 102388.

[7] Miah, A.S.M., Hasan, M.A.M., and Shin, J. 2023. Dynamic hand gesture recognition using multi-branch attention based graph and general deep learning model. IEEE Access, 11, 4703-4716.

[8] Byberi, A., Ravan, M., and Amineh, R.K.. 2023. GloveSense: A hand gesture recognition system based on inductive sensing. IEEE Sensor Journal, 23(9), 9210-9219.

[9] Abiodun, O.I., Jantan, A., A.E. Omolara, K.V., Dada, N., Mohamed, N., and Arshad, H. 2018. State-of-the-art in artificial neural network applications: A survey. Heliyon, 4, e00938.

[10] Erkaymaz, O. 2020. Resilient back-propagation approach in small-word feed-forward neural network topology based on Newman-Watts algorithm. Neural Computing and Applications, 30, 16279-16289.

[11] He, C., Ma, M., and Wang, P. 2020. Extract interpretability-accuracy balanced rules from artificial neural networks: A review, Neurocomputing, 387, 346-358.

[12] Robles-Velasco, A., Munuzuri, J., Onieva, L., and Rodriquez-Palero, M. 2021. Trends and application of machine learning in water supply network management. Journal of Industrial Engineering and Management, 14, 45-54.

[13] Igel, C., and Husken, M. 2003. Empirical evaluation of the improved Rprop learning algorithms. Neurocomputing, 50, 105-123.

[14] Nanni, L., Lumini, A., Ferrara, M., and Capelli, R. 2015. Combining biometric matches by means of machine learning and statistical approaches. Neurocomputing, 149, 526-535.

[15] Wu, J.D., and Liu, C.T. 2011. Finger-vein pattern identification using principal component analysis and the neural network technique. Expert Systems with Applications, 38, 5423-5427.

[16] Raji, R.K., Adjeisah, M., Miao, X., and Wan, A. 2020. A novel respiration pattern biometric prediction system based on artificial neural network. Sensor Review, 40, 8-16.

[17] Siswanto, A., Tarigan, P., and Fahmi, F. 2013. Design of contactless hand biometric system with relative geometric parameters. In: Proceedings of the IEEE 3rd International Conference on Instrumentation, Communications, Information Technology and Biomedical Engineering, 7-8 November, Bandung, Indonesia, pp. 199-203.

[18] Salve, S.S., and Narote, S.P. 2016. Iris recognition using SVM and ANN. In: Proceedings of the IEEE International Conference on Wireless Communications, Signal Processing and Networking, 23-25 March, Chennai, India, pp. 474-478.

[19] Patro, K.K., and Kumar, P.R. 2017. Machine learning classification approaches for biometric recognition system using ECG signals. Journal of Engineering Science and Technology Review, 10, 1-8.

[20] Kim, S., Yeun, C.Y., and Yoo, P.D. 2019. Enhanced machine learning based biometric authentication system using RR-interval framed electrocardiograph. IEEE Access, 7, 168669- 168674.

[21] Patro, K.K., Prakash, J.A., Rao, J.M., and Kumar, R.P. 2022. An efficient optimized feature selection with machine learning approach for ECG biometric recognition. IETE Journal of Research, 68(4), 2743-2754.

[22] Abozaid, A., Haggag, A., Kasban, H., and Eltokhy, M. 2019. Multimodal biometric scheme for human authentication technique based on voice and face recognition fusion. Multimedia Tools and Applications, 78, 16345-16361.

[23] Thamaraimanalan, T., Pranavakumar, S., Lingeswaran,

R.A, Kabileshar, R.M., and Sarankarthick, M. 2021. Multi biometric authentication using SVM and ANN classifiers. Irish Interdisciplinary Journal of Science & Research, 5, 118-130.

[24] Patterh, M.S. 2017. A biometric fusion based on face and fingerprint recognition using ANN. International Journal on Recent and Innovation Trends in Computing and Communication, 5, 88-92.

[25] Ali, H., and Salami, M.J. 2009. Keystroke pressure based typing biometrics authentication system by combining ANN and ANFIS-based classifiers. In: Proceedings of the IEEE 5th International Colloquium on Signal Processing & Its Applications, 6-8 March, Kuala Lumpur, Malaysia, pp. 198-203.

[26] Harun, N., Woo, W.L., and Dlay, S.S. 2010. Performance of keystroke biometrics authentication system using artificial neural network (ANN) and distance classifier method. In: Proceedings of the IEEE International Conference on Computer and Communication Engineering, 11-12 May, Kuala Lumpur, Malaysia, pp. 1-6.

[27] Harun, N., Dlay, S.S., and Woo, W.L. 2010. Performance of keystroke biometrics authentication system using Multilayer Perceptron neural network (MLP NN). In: Proceedings of the IEEE 7th International Symposium on Communication Systems, Networks & Digital Signal Processing, 21-23 July, pp. 711-714.

[28] Ranganathan, A. 2004. The Levenberg-Marquardt algorithm. Tutorial on LM Algorithm, 11, 101-110.

[29] Lourakis, M.I. 2005. A brief description of the Levenberg-Marquardt algorithm implemented by levmar. Foundation of Research and Technology, 4, 1-6.

[30] Amini, K., and Rostami, F. 2016. Three-steps modified Levenberg–Marquardt method with a new line search for systems of nonlinear equations. Journal of Computational and Applied Mathematics, 300, 30-42

[31] De Rubio, J.J. 2020. Stability analysis of the modified Levenberg-Marquardt algorithm for the artificial neural network training. IEEE Transactions on Neural Networks and Learning Systems, 32(8), 3510-3524.

[32] Babani, L., Jadhav, S., and Chaudhari, B. 2016. Scaled conjugate gradient based adaptive ANN control for SVM-DTC induction motor drive. In: Proceedings of the 12th IFIP International Conference on Artificial Intelligence Applications and Innovations, Sepetember, Thessaloniki, Greece. pp.384-395.

[33] Prasad, N., Singh, R., and Lal, S.P. 2013. Comparison of back propagation and resilient propagation algorithm for spam classification. In: Proceedings of the IEEE 5th International Conference on Computational Intelligence, Modelling and Simulation, pp. 29-34.

[34] Andrei, N. 2007. Scaled conjugate gradient algorithms for unconstraint optimization. Computational Optimization and Applications, 38, 416-416.

[35] Wang, S., and Guan, H. 2013. A scaled conjugate gradient method for solving monotone nonlinear equations with convex constraints. Journal of Applied Mathematics, 2013, 1-7.

[36] Dai, Y.H. 2002. Convergence properties of the BFGS algorithm. SIAM Journal of Optimization, 13, 693-701.

[37] Yang, T., Li, P., and Wang, X. 2020. Convergence analysis of an improved BFGS method and its application in the Muskingum model. Mathematical Problems in Engineering, 2020, 1-9.

[38] Mishra, S.K., Panda, G., Chakraborty, S.K., Samei, M.E., and Ram, B. 2020. On q-BFGS algorithm for unconstrained optimization problems. Advances in Difference Equations, 2020(1), 1-24.

[39] Li, D., and Fukushima, M. 2001. A modified BFGS method and its global convergence in noncovex minimization. Journal of Computational and Applied Mathematics, 29, 15-35.

[40] Lai, K.K., Mishra, S.K., Panda, G., Chakraborty, S.K., Samei, M.E., and Ram, B. 2020. A limited memory q-BFGS algorithm for unconstraint optimization problems. Journal of Applied Mathematics and Computing, 66, 183-202.

[41] Fariq, H., and Simoncelli, E.P. 2004. Differentiation of discrete multi-dimensional signals. IEEE Transactions on Image Processing, 13, 496-508.

[42] Hast, A., Simple filter design for first and second order derivatives by a double filtering approach. Pattern Recognition Letters, vol. 42, 2014, pp. 65-71.

[43] Khosravi, M.R., & Yazdi, M., (2018). A lossless data hiding scheme for medical images using a hybrid solution based on IBEW error histogram computation and quartered interpolation with greedy weights. Neural Computing and Applications, 30, 2017-2028.