Article

Not peer-reviewed version

Secure CAPTCHA by Genetic Algorithm (GA) and Multi-Layer Perceptron (MLP)

Saman Shojae Chaeikar *, Fatemeh Mirzaei Asl , Saeid Yazdanpanah , Mazdak Zamani *, Touraj Khodadadi

Posted Date: 27 July 2023

doi: 10.20944/preprints202307.1935.v1

Keywords: CAPTCHA; authentication; hand gesture recognition; genetic algorithm; multilayer perceptron; MLP

Article

Secure CAPTCHA by Genetic Algorithm (GA) and Multi-Layer Perceptron (MLP)

Saman Shojae Chaeikar [1,*], Fatemeh Mirzaei Asl [2], Saeid Yazdanpanah [2], Mazdak Zamani [3,*] and Touraj Khodadadi [4]

[1] Australian Institute of Higher Education, Sydney, Australia
[2] Department of Computer Engineering, Khorramabad Branch, Islamic Azad University, Khorramabad, Iran
[3] New York University, USA
[4] Department of Information Technology, Malaysia University of Science and Technology, Selangor, Malaysia
* Correspondence: saman.shojae_chaeikar@mq.edu.au (S.S.C.); zamanim@felician.edu (M.Z.)

Abstract: To achieve an acceptable level of security on the web, Completely Automatic Public Turing test to tell Computer and Human Apart (CAPTCHA) was introduced as a tool to prevent bots from doing destructive actions such as downloading or signing up. Mobile devices have small screens, and therefore, using the common CAPTCHA methods (e.g. text CAPTCHAs) in these devices raises usability issues. To introduce a reliable, secure, and usable CAPTCHA that is suitable for mobile devices, this paper introduces a hand gesture recognition CAPTCHA based on applying Genetic Algorithm (GA) principles on Multi-Layer Perceptron (MLP). The proposed method improves the performance of **MLP-based** hand gesture recognition. It has been trained and evaluated on 2201 videos of IPN Hand dataset, and MSE and RMSE benchmarks report the index values of 0.0018 and 0.0424, respectively. Comparison with the related works shows a minimum of 1.79% fewer errors, and experiments produced a sensitivity of 93.42% and accuracy of 92.27% – 10.25% and 6.65% improvement compared to the MLP implementation.

Keywords: CAPTCHA; authentication; hand gesture recognition; genetic algorithm; multilayer perceptron; MLP

1. Introduction

The Internet has turned into an essential requirement of any modern society, and almost everybody around the world uses it for daily life activities e.g. communication, shopping, research, etc. The growing number of Internet applications, the large volume of the exchanged information, and the diversity of the services attract the attention of people wishing to gain unauthorized access to these resources, e.g. hackers, attackers, and spammers. These attacks highlight the critical role of information security technologies to protect resources and limit access to only authorized users. To this end, various preventive and protective security mechanisms, policies, and technologies are designed [1,2].

Approximately a decade ago, Completely Automatic Public Turing test to tell Computers and Humans Apart (CAPTCHA) was introduced by Ahn et al. [3] as a challenge-response authentication measure. The challenge, as illustrated in Figure 1, is an example of Human Interaction Proofs (HIPs) to differentiate between computer programs and human users. The CAPTCHA methods are often designed based on open and hard Artificial Intelligence (AI) problems that are easily solvable for human users [4].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 July 2023 doi:10.20944/preprints202307.1935.v1

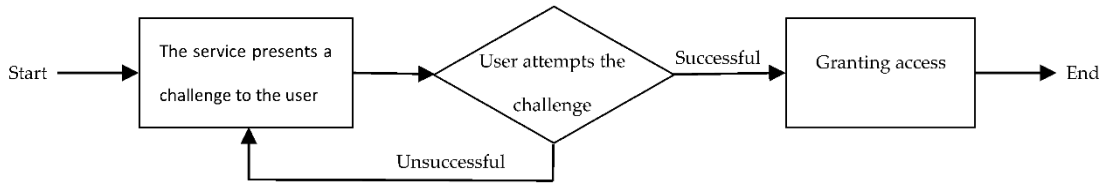

Figure 1. The structure of challenge response authentication mechanism.

As attempts for unauthorized access increase every day, the use of CAPTCHA is needed to prevent the bots disrupt the services such as subscription, registration, and account/password recovery, running attacks like spamming blogs, search engine attacks, dictionary attacks, email worms, and block scrapers. Online polling, survey systems, ticket bookings, and e-commerce platforms are the main targets of the bots [5].

The critical usability issues of initial CAPTCHA methods pushed the cybersecurity researchers to evolve the technology towards alternative solutions that alleviate the intrinsic inconvenience with a better user experience, more interesting designs (i.e. gamification), and support disabled users. The emergence of powerful mobile devices motivated the researchers to design gesture-based CAPTCHAs that are a combination of cognitive, psychomotor, and physical activities within a single task. This class is robust, user-friendly, and suitable for people with disabilities such as hearing and in some cases vision impairment. A gesture-based CAPTCHA works based on the principles of image processing to detect hand motions. It analyzes low-level features such as color and edge to deliver human-level capabilities in analysis, classification, and content recognition.

Recently, outstanding advances have been made in hand gesture recognition. The image processing techniques perform segmentation and feature extraction on the input images and detect the hand pose, fingertips, and hand palm in **real time**. During the past twenty years, the researchers used color or wired gloves equipped with sensors to detect hand pose. However, the skin-color-based analysis does not require a paper cover or sensors in order to detect fingers and works fast, accurately, and in real time.

To detect the hand gesture, the algorithm should first transform the video frames into two-dimensions images, and then apply segmentation and skin filter functions. In this area, the researchers have presented various methods that utilize machine learning techniques such as Support Vector Machines (SVM) [6], Artificial Neural Networks (ANN) [7], fuzzy systems [8], Deep Learning (DL) [9], and metaheuristic methods [10].

2. Background and Related Works

The increasing popularity of the web has made it an attractive ground for hackers and spammers, especially where there are financial transactions or user information [1]. CAPTCHAs are used to prevent bots from accessing the resources designed for human use e.g. login/sign-up procedures, online forms, and even user authentication [11]. The general applications of CAPTCHAs include preventing comment spamming, accessing email addresses in plaintext format, dictionary attacks, and protecting website registration, online polls, e-commerce, and social network interactions [1].

Based on the background technologies, CAPTCHAs can be broadly classified into three classes: visual, non-visual, and miscellaneous [1]. Each class then may divide into several sub-classes, as illustrated in Figure 2.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 July 2023 doi:10.20944/preprints202307.1935.v1

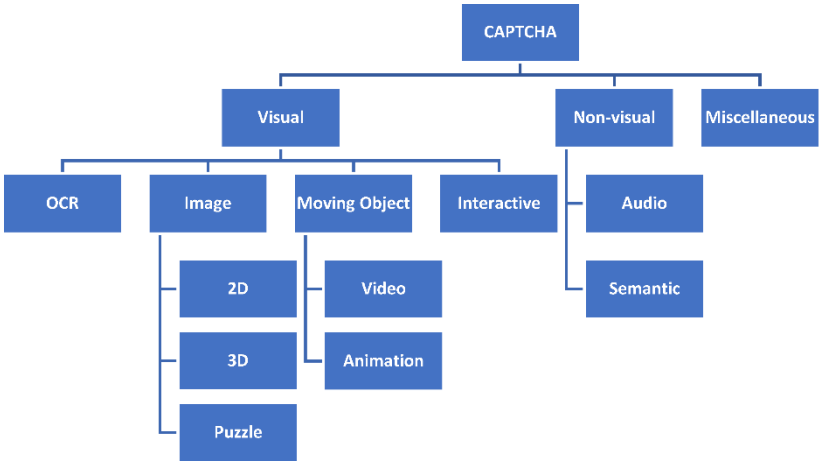

Figure 2. CAPTCHA classification.

The first class is visual CAPTCHA which challenges the user on its ability to recognize the content. Despite the usability issues for visually impaired people, this class is easy to implement and straightforward.

Text-based or Optic Character Recognition (OCR) based CAPTCHA is the first sub-class of visual CAPTCHAs. While the task is easy for a human, it challenges the automatic character recognition programs by distorting the combination of numbers and alphabets. Division, deformation, rotation, color variation, size variation, and distortion are some examples of techniques used for making the text unrecognizable for the machine [12]. BaffleText [12], Pessimal Print [13], GIMPY and Ez—Gimpy [14], and ScatterType [15] are some of the proposed methods in this area. Google reCHAPTCHA v2 [16], Microsoft Office, and Yahoo! Mail are famous examples of industry-designed CAPTCHAs [17]. Bursztein et al. [18] designed a model to measure the security of text-based CAPTCHAs. They identified several areas of improvement and designed a generic solution.

Due to the complexity of object recognition tasks, the second sub-class of visual CAPTCHAs is designed on the principles of semantic gap [19] – people can understand more from a picture than a computer. Among the various designed image-based CAPTCHAs, ESP-Pix [3], Imagination [20], Bongo [3], ARTiFACIAL [21], and Asirra [22] are the more advanced ones. The image-based CAPTCHAs have two variations: 2D and 3D [23]. In fact, the general design of the methods in this sub-class is 2D; however, some methods utilize 3D images (of characters) as a human is better in recognition of 3D images. Another type of image-based CAPTCHA is a puzzle game designed by Gao et al. [24]. The authors used variants in color, orientation, and size, and they could successfully enhance the security. They prove that puzzle-based CAPTCHAs need less time than text-based ones. An extended version of [24] that uses animation effects was later developed in [25].

Moving object CAPTCHA is a relatively new trend that shows a video and asks the users to type what they have perceived or seen [26]. Despite being secure, this method is very complicated and expensive compared to the other solutions.

The last sub-class of visual CAPTCHAs, interactive CAPTCHA, tries to increase the user interactions to enhance security. This method requires more mouse click, drag, and similar activities. Some variations request the user to type the perceived or identified response. Except the users that have some sorts of disabilities, solving the challenges are easy for human but hard for computers. The methods designed by Winter-Hjelm et al. [27] and Shirali-Shahreza et al. [28] are two interactive CAPTCHA examples.

The second CAPTCHA class contains non-visual methods in which the user is assessed based on audio and semantic tests. The audio-based methods are subject to speech recognition attacks while the semantic-based methods are very secure and much harder to be cracked by computers. Moreover, semantic-based methods are relatively easy for users, even the ones with hearing or visual impairment. However, the semantic methods might be very challenging for users with cognitive deficiencies [1].

The first audio-based CAPTCHA method was introduced by Chan [29], and later other researchers such as Holman et al. [30] and Schlaikjer [31] proposed more enhanced methods. A relatively advanced approach in this area is to ask the user to repeat the played sentence. The recorded response is then analyzed by a computer to check if the recorded speech is made by a human or a speech synthesis system [28]. Limitations of this method include requiring equipment such as a microphone and speaker and being difficult for people with hearing and speech disabilities.

Semantic CAPTCHAs are relatively a secure CAPTCHA sub-class as computers are far behind the capabilities of humans in answering semantic questions. However, still they are vulnerable to attacks that use computational knowledge engines, i.e. search engines or Wolfram Alpha. The implementation cost of semantic methods is very low as they are normally presented in a plaintext format [1]. The works by Lupkowski et al. [32] and Yamamoto et al. [33] are examples of a semantic CAPTCHA.

The third CAPTCHA class utilizes diverse technologies, sometimes in conjunction with visual or audio methods, to present techniques with novel ideas and extended features. Google's reCAPTCHA is a free service [16] for safeguarding websites from spam or unauthorized access. The method works based on adaptive challenges and an advanced risk analysis system to prevent bots access web resources. In reCAPTCHA, the user first needs to click on a checkbox. If it fails, the user then needs to select specific images from the set of given images. Google, in 2018, released the third version i.e. reCAPTCHA v3 or NoCAPTCHA [34]. This version monitors the user's activities and reports the probability of being human or robot without needing the user to click the “I'm not a robot” checkbox. However, NoCAPTCHA is vulnerable to some technical issues such as erasing cookies, blocking JavaScript, and using incognito web browsing.

Yadava et al. [35] designed a method that displays the CAPTCHA for a fixed time and refreshes it until the user enters the correct answer. The refreshment only changes the CAPTCHA, not the page, and the defined time makes cracking it harder for the bots.

Wang et al. [36] put forward a graphical password scheme based on CAPTCHAs. The authors claimed it can strengthen security by enhancing the capability to resist brute force attacks, withstand spyware, and reduce the size of password space.

Solved Media presented an advertisement CAPTCHA method in which the user should enter a response to the shown advertisement to pass the challenge. This idea can be extended to different areas, e.g. education or culture [1].

A recent trend is to design personalized CAPTCHAs that are specific to the users based on their conditions and characteristics. In this approach, an important aspect is identifying the cognitive capabilities of the user that must be integrated into the process of designing the CAPTCHA [37]. Another aspect is personalizing the content by considering factors such as geographical location to prevent third-party attacks. As an example, Wei et al. [38] developed a geographic scene image CAPTCHA that combines Google Map information and an image-based CAPTCHA to challenge the user with information that is privately known.

Solving a CAPTCHA on a small screen of a smartphone might be a hassle for the users. Jiang et al. [39] found that wrong touches on the screen of a mobile phone decrease the CAPTCHA performance. Analysis of the user's finger movements in the back end of the application can help in differentiating between a bot and a human [40]. For example, by analyzing the sub-image dragging pattern, their method can detect whether the action is “BOTish” or “HUMANish”. Some similar approaches like [41,42] challenge the user for finding segmentation points in cursive words.

In the area of Hand Gesture Recognition (HGR), the early solutions were using a data glove, hand belt, and camera. Luzhnica *et al.* [43] used a hand belt equipped with a gyroscope, accelerometer, and Bluetooth for HGR. Hung *et al.* [44] acquired the input required data from hand gloves. Another research has used Euclidean distance for analyzing twenty-five different hand gestures and employed an SVM for classification and controlling tasks [45]. In another effort [46], the researchers converted the Red, Green, and Blue (RGB) captured images to grayscale, applied a Gaussian filter for noise reduction, and fed the results to a classifier to detect the hand gesture.

 doi:10.20944/preprints202307.1935.v1

Chaudhary *et al.* [47] used a normal Windows-based webcam for recording user gestures. Their proposed method extracts the Region of Interest (ROI) from frames and applies a Hue, Saturation, Value (HSV)-based skin filter on RGB images in particular illumination conditions. To help the fingertip detection process, the method analyzes hand direction according to pixel density in the image. It also detects the palm hand based on the number of pixels located in a 30×30-pixel mask on the cropped ROI image. The method has been implemented by a supervised neural network based on Levenberg–Marquardt algorithm and uses eight thousand samples for all fingers. The architecture of the method has five input layers for five input finger poses, and five output layers for the bent angle of the fingers. Marium *et al.* [48] extracted the hand gesture, palms, fingers, and fingertips from webcam videos using the functions and connectors in OpenCV.

Simion *et al.* [49] researched finger pose detection for mouse control. Their method, in the first step, detects the fingertips (except the thumb) and palm hand. Its second step is to calculate the distances between the pointing and middle fingers to the centre of the palm, and the distance between the tips of the pointing and middle fingers. The third step computes the angles between the two fingers, between the pointing finger and the x-axis, and between the middle finger and the x-axis. The researchers used six hand poses for mouse movements including right/left click, double click, and right/left movement.

The main contribution of this paper is the development of a novel hand-gesture-controlled CAPTCHA for mobile devices that is easy to use even for people with visual challenges. Choosing a set of easy-to-imitate hand gestures enhances the usability of the method and minimizes the number of reattempts. The originality of the method is extended by detecting the bending angle of human fingers through applying Genetic Algorithm (GA) principles on a Multi-Layer Perceptron (MLP). The steps include detecting the palm center and fingertips in real time, defining the distance between the palm center and each fingertip, and calculating the bending degree of fingers based on the detected distances. It then detects the bending degree of human fingers and accordingly the hand gesture.

In what followed next, in section 3, we first review the structure of the proposed method including the processes of hand detection based on skin color, detecting hand pose, palm, fingertips and fingers' bend angle. Section 4 covers implementation and experiments. Analysis, comparison, and discussion are located in section 5, and the work is concluded in section 6.

3. The Proposed Method

Gestural CAPTCHA, hereafter called GAPTCHA, shows a set of simple hand poses to the user and then requests the user to imitate these gestures in front of the camera. As illustrated in Figure 3, in this method, the essence of recognizing the gestures is calculating the fingers' bent angles. The method preprocesses and applies a skin filter on the input 2D image. The employed segmentation method extracts the hand pose from an image, even if the background contains skin-colored objects. It focuses only on the extracted areas of interest in the image for faster and more accurate analysis. The method first detects the fingertips and palm in the segmented image and then calculates the distance between the center of the palm and each fingertip. It calculates the bending angle of fingers based on the distances and detects fingers, palm, and angles with no errors. The proposed method can work with a normal quality webcam for capturing the user's hand movements, and no marked gloves, wearable sensors, or even a long sleeve is required. It also has no limitation for the camera angle.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 July 2023 doi:10.20944/preprints202307.1935.v1

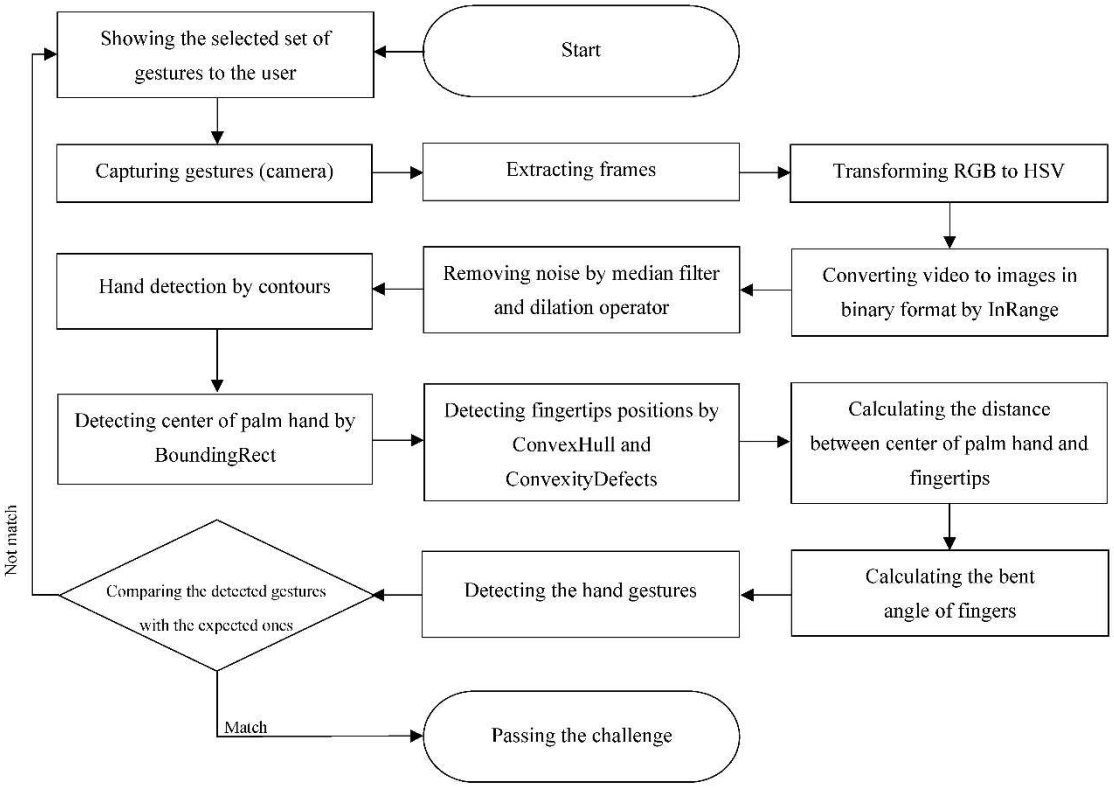

Figure 3. The GAPTCHA process.

The most famous segmentation method in HGR systems is skin color, as it is invariant to the changes like size, rotation, and movements. The captured webcam images are in an RGB color system and the involved parameters in this format are highly correlated and sensitive to illumination. However, the HSV system separates the color and illumination information. Therefore, in skin-color-based HGR systems, the captured RGB images are first converted to the HSV system and then segmented to generate the binary version of the image. Each pixel of a binary image is stored in one bit, and due to the skin-colored pixels in the background, these images normally carry some noise. As shown in Figure 4, this noise creates unwanted spots in the image. A median filter then removes the isolated spots and a dilation operator fills the regions.

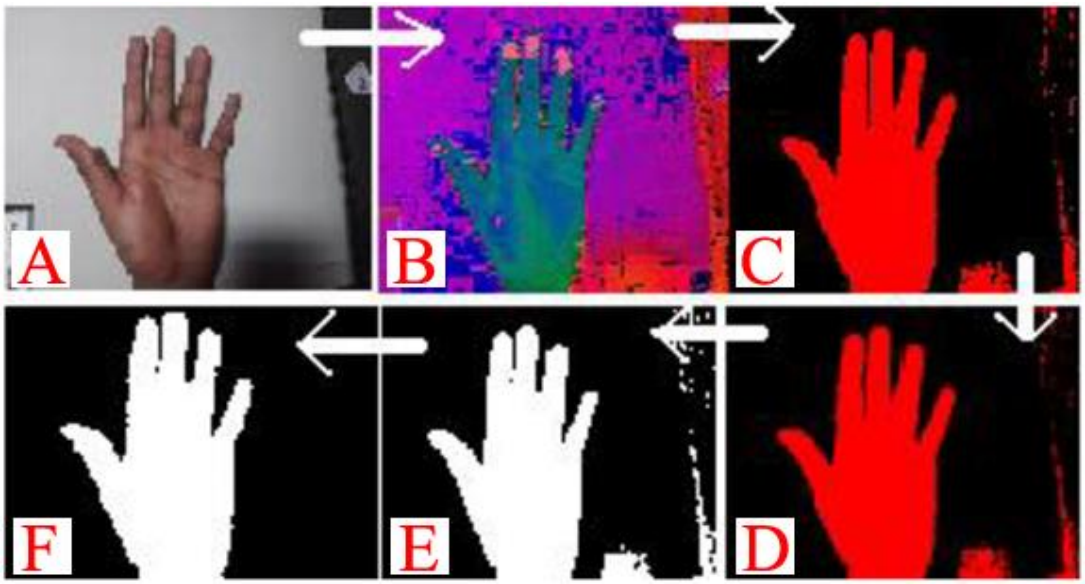

Figure 4. An example of producing a binary image: (**A**) original images, (**B**) HSV conversion, (**C**) filtered image, (**D**) smooth image, (**E**) binary image, and (**F**) noise-free image [50].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 July 2023 doi:10.20944/preprints202307.1935.v1

The proposed method supports freedom of angle for the camera and user, and the only limitation is to face the palm hand to the camera. The method uses contours to extract the hand image from the binary image. A contour is a list of the points that represent a curve in the image and its main application is in the analysis and detection of shapes. Here, FindContours in OpenCV is used to find the largest contour in the binary image – the hand shape shown in Figure 5 section A. Finding the center of the contour helps in finding the center of the palm hand (Figure 5, section B). To this end, using the BoundingRect function in OpenCV, a bounding box is applied on the hand shape to find the palm hand center.

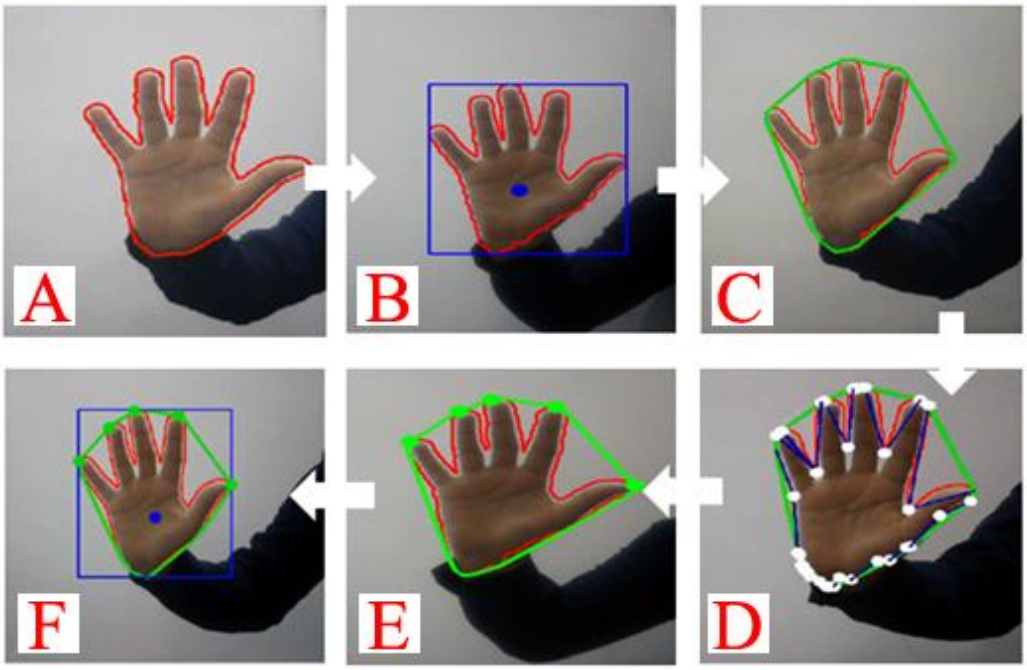

Figure 5. (**A**) hand contour, (**B**) palm hand, (**C**) ConvexHull polygon, (**D**) ConvexityDefects, (**E**) fingertips, (**F**) palm hand and fingertips.

The ConvexHull algorithm is employed to identify fingertips. It returns a set of polygons in which the corners of the largest one represents the fingertips – Figure 5 section C. To automate the process, ConvexityDefects function approximates the gaps between the contour and the polygon by straight lines. The output of this function is multiple records of four fields: (i) the starting defect point, (ii) the ending defect point, (iii) the middle (farthest) defect point that connects starting and ending points, and (iv) the approximate distance to the farthest point. A sample output of this step is shown in Figure 5 section D. Each record results in two lines: a line from starting point to the middle point, and a line from the middle point to the endpoint. However, the function may return more points than the number of fingertips. The steps for filtering the detected points and finding the correct location of fingertips include (i) calculating the internal angle between two defect areas in a certain period, (ii) calculating the angle between the starting point and contour center in a certain period, and (iii) measuring the line length to remain below the defined threshold. The green points in section E of Figure 5 represent the results of this step.

Eq. 1, 2, and 3 are utilized to calculate the length of the produced vectors from start (point p_1), middle (point p_2), and end (point p_3) points, and Eq. 4 computes the angle. $p_{1,2}$, $p_{1,3}$, and $p_{2,3}$ are length of the vectors calculated by convexityDefects function.

$$p_{1,2} = \sqrt{(p_1.x - p_2.x)^2 + (p_1.y - p_2.y)^2} \quad (1)$$

$$p_{1,3} = \sqrt{(p_1.x - p_3.x)^2 + (p_1.y - p_3.y)^2} \quad (2)$$

$$p_{2,3} = \sqrt{(p_2.x - p_3.x)^2 + (p_2.y - p_3.y)^2} \quad (3)$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 July 2023 doi:10.20944/preprints202307.1935.v1

$$\theta = \arccos\left(\frac{{p_{1,2}}^2 + {p_{1,3}}^2 + {p_{2,3}}^2}{2 \times p_{1,2} \times p_{1,3}}\right) \quad (4)$$

Calculating the angle between the first point of defect area and the center of the contour is an essential step for removing the non-fingertips areas. The consequent step is to filter the points to the ones between -30 to +160 degrees. Eq. 5 returns the degree between the first point of the defect area and center of the contour, and Eq. 6 calculates the Euclidean distance – the vector length – between first to middle points. The whole process from finding the hand contour until detecting palm hand and fingertips is illustrated in Figure 5. In this equation, the word *center* represents the center of the contour i.e. center of the palm hand.

$$\theta = arctan\,(center.y - p_2.y, center.x - p_2.x) \quad (5)$$

$$length = \sqrt{(p_2.x - p_1.x)^2 + (p_2.y - p_1.y)^2} \quad (6)$$

As already explained, this research uses a combination of MLP and GA for predicting the bent angle of fingers. The used MLP has a flexible structure in which neurons are located in hidden layers and each neuron can influence its input and produce the desired output. To this end, a weight applies to the input value of each neuron and the result passes into an activation function together with a bias value. Reducing classification errors and correct prediction of the pose in an artificial neural network depends on selecting optimum weight and bias values. Figure 6 illustrates the detailed structure of the proposed method for applying GA principles on MLP to accurately detect the finger's bent angle, and accordingly recognize the hand gesture.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 July 2023 doi:10.20944/preprints202307.1935.v1

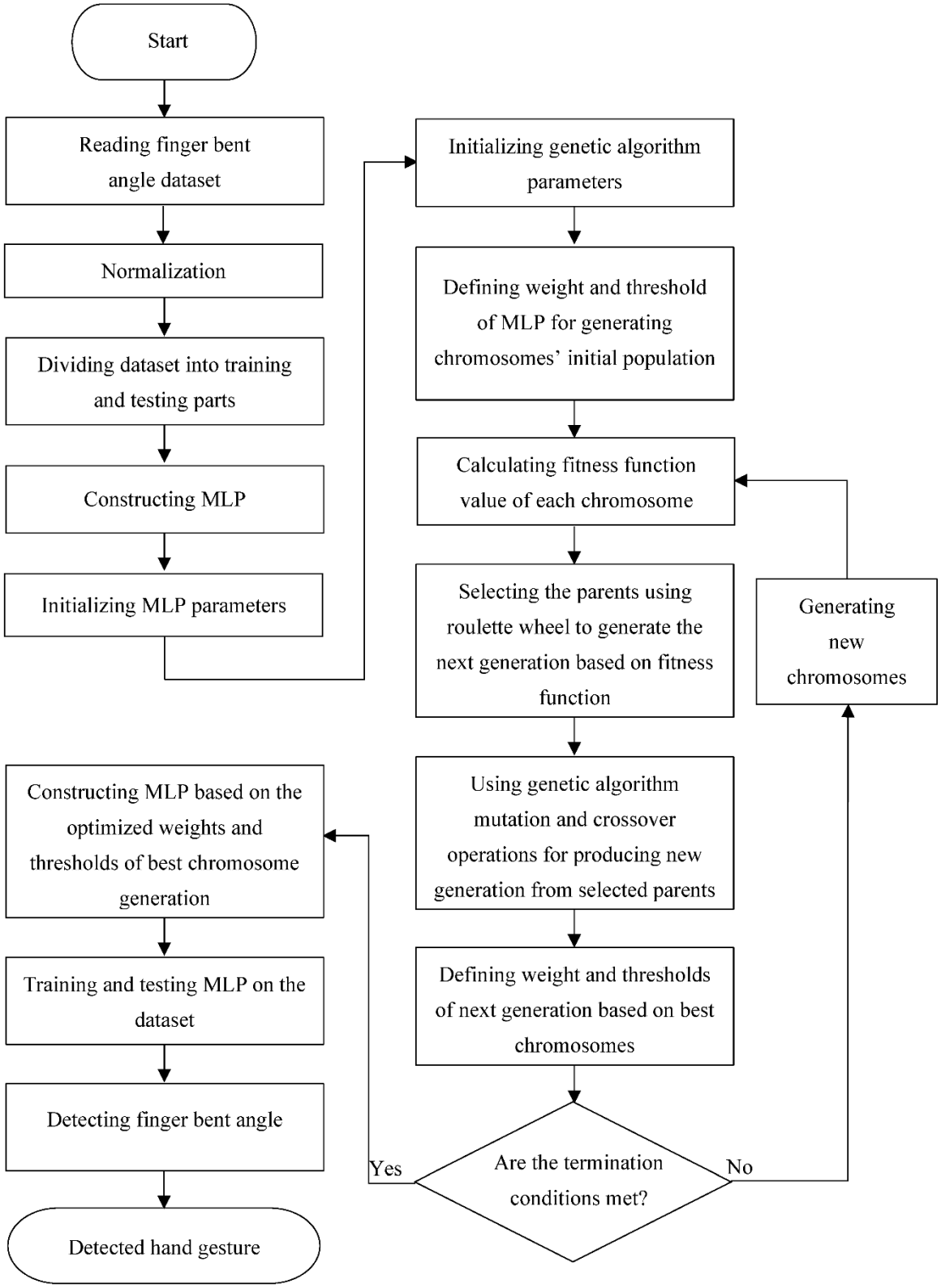

Figure 6. The overall designed process for hand gesture recognition by applying GA principles on MLP.

In GAPTCHA, the structure of each chromosome is its weight and bias in the neural network. This can be shown in the format of $[w_{1,1}, \dots, w_{3,4}, \dots, w_{1,0}, \dots, w_{4,0}, \dots, b_1, b_2]$ in which $w_{i,j}$is weight of the i^{th} entry into the j^{th} neurone of the hidden layer, $w_{j,0}$is the output, and $b_1$and b_2 are the threshold values of the hidden and output layers, respectively. Upon defining the encoding system and the method of converting each answer to a chromosome, the next step is to produce the initial

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 July 2023 doi:10.20944/preprints202307.1935.v1

chromosome population. Normally, generating the initial population is a random process, however, heuristic algorithms can accelerate and optimize it. GAPTCHA uses a roulette wheel mechanism for selecting parents in mutation and crossover processes. In this mechanism, the probability of selecting a chromosome depends on how suitable it is for the evaluation function. In other words, the higher quality of the chromosome, the higher chance to be selected for producing the next generation, and vice versa. Eq. 7 calculates the chance of selecting a chromosome in the roulette wheel.

$$p = \frac{fi}{\sum_{j=1}^{n} fj} \tag{7}$$

In the above equation, the probability (p) of selecting the i^{th} chromosome is the proportion of the fitness function (f) value of chromosome i to the sum of fitness function values of all chromosomes. Single-point crossover is used to apply crossover in this method. It chooses a random point in the chromosomes and swaps the information in the remaining parts. The steps to apply single-point crossover are:

i. Choose a random chromosome value between 0 and 1,
ii. Go to the next step and mutate if the number is bigger than the mutation threshold (a value between 0 and 1), otherwise skip mutation,
iii. Choose a random number that indicates one of the chromosome genes and makes a numerical mutation.

4. Implementation and Experiments

This research uses OpenCV functions and C++ for preprocessing, segmentation, and feature extraction from video. The functions help to detect the center of the palm hand, fingertips, and bent angle of fingers. Matlab is the chosen platform due to having rich programming interface and metaheuristic libraries. Applying GA principles to MLP helps the method to enhance predicting fingers' bent angles. The followings are the steps taken for implementing the proposed method:

i. Defining the initial parameters such as population, number of iterations, mutation rate, and crossover rate of GA,
ii. Preprocessing and normalizing part of input data to reduce learning errors and enhance performance in hand movement detection,
iii. Dividing the dataset into training and evaluation parts,
iv. Producing chromosomes for MLP and applying mutation and crossover on the chromosomes to find optimum weight and bias and reduce the error rate,
v. Evaluating the method according to the metrics in section 4.2.

4.1. Dataset

The selected dataset for training and evaluation of the proposed method is "IPN Hand", produced by Benitez et al [51], that consists 5649 RGB videos recorded in the resolution of **640×480 at 30 fps**. The dataset has two parts: 1431 non-gestural and 4218 gestural videos. Since the gestural section contains gesture categories that are not easy to imitate, considering the usability requirement of GAPTCHA, some video categories removed from the selected dataset.

After filtering the videos, the resulting subset contains 200 click with one finger, 200 click with two fingers, 200 throw up, 201 throw down, 200 throw left, 200 throw right, 200 open twice, 200 double click with one finger, 200 double click with two fingers, 200 zoom in, and 200 zoom out videos – a total of 2201.

4.2. Evaluation Metrics

In the proposed method, each hand pose is considered as a class and is defined by a set of features. Therefore, this research challenges a multiclass optimization problem. The proposed method classifies each given sample in its corresponding class and tries to minimize the average learning and classification errors. The Mean Square Error (MSE) and Root Mean Square Error (RMSE)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 July 2023 doi:10.20944/preprints202307.1935.v1

are the chosen goal test measures and can be calculated using Eq. 8 and 9. In these equations, n represents the number of the samples, and real and predicted values of an instance, e.g. index i, are shown by o_i and $\hat{o_i}$, respectively.

$$MSE = \frac{\sum_{i=1}^{n}(\hat{o_i} - o_i)^2}{n} \tag{8}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{o_i} - o_i)^2}{n}} \tag{9}$$

Two other important parameters for evaluating the applicability of the proposed algorithm are sensitivity and accuracy. Sensitivity is the proportion of true positive detections to the sum of true positives and false negatives (Eq. 10), and accuracy is the ratio of truly detected positives and negatives to all detections (Eq. 11).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{o_i} - o_i)^2}{n}} \tag{10}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

4.3. Error Analysis by Increasing Population and Iterations

The experiments were conducted by two populations of 5 and 10. The number of iterations was set to 20, and mutation and crossover ratios were defined as 0.15 and 0.5, respectively. A chromosome is a two-layer MLP in which each layer has four neurons. MSE prediction errors of both populations are reported in Figures 7 and 8.

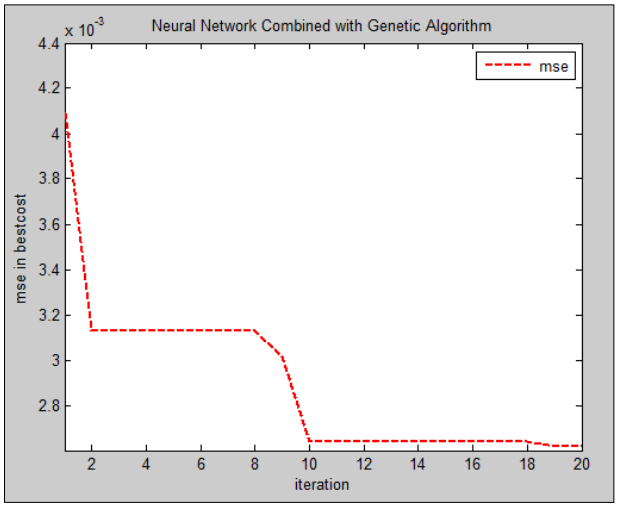

Figure 7. Prediction error rate with 5 chromosomes in the proposed method.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 July 2023 doi:10.20944/preprints202307.1935.v1

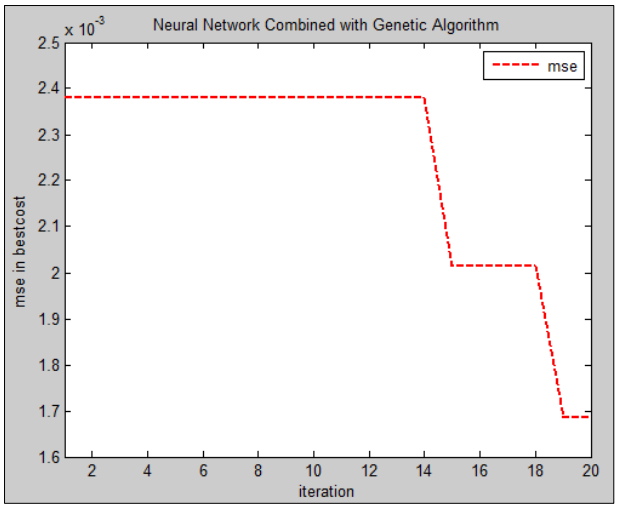

Figure 8. Prediction error rate with 10 chromosomes in the proposed method.

The reported values in Figures 7 and 8 prove that increasing the number of chromosomes results in reducing MSE for hand pose detection. It happens due to the following reasons:

I. Increasing the number of chromosomes increases the number of neural networks for prediction, and results in a more accurate classification,
II. Increasing population produces more and diverse test ratios in GA, and accordingly increases the chance of finding a final and accurate answer,
III. Increasing chromosomes in GA increases problem search space that normally results in reaching optimum answers and reducing error,
IV. Increasing the population increases the number of elite members and enhances the probability of mutation and crossover. This leads to increasing the chance of having a more accurate MLP for hand pose prediction.

In addition to the population, the number of iterations can help in reducing hand gesture detection errors. Behavioral analysis of Figures 8 and 9 reveals that MSE and the number of iterations have a reverse correlation. In other words, increasing iterations give more chance to the chromosomes to choose optimum ratios in the MLP that leads to lower MSE.

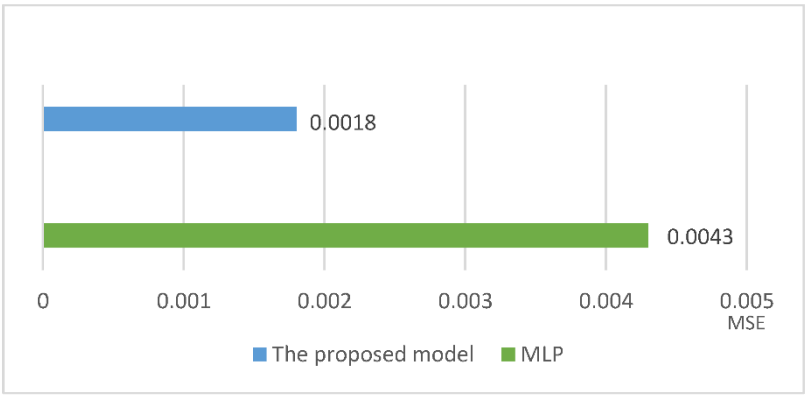

Figure 9. Comparing MSE of the proposed method and MLP implementation.

5. Comparison and Discussion

To analyze and compare, the proposed GA-based MLP method has been compared with a standard MLP implementation. They both have four hidden neurons, a sigmoid activation function, a population of 15, 30 iterations, 0.15 mutation, and 0.5 crossover. Also, 75% of the dataset is dedicated to training and 25% to testing. Figure 9 demonstrates the MSE difference between the proposed method and the standard MLP, and Figure 10 compares them in terms of the RMSE benchmark.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 July 2023 doi:10.20944/preprints202307.1935.v1

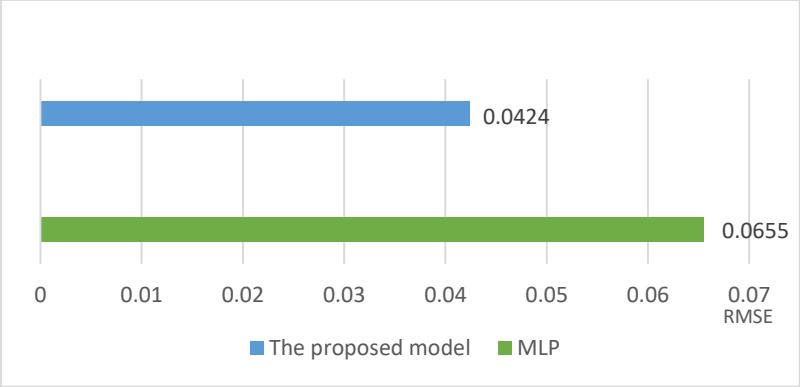

Figure 10. Comparing RMSE of the proposed method and MLP implementation.

The experiments and analysis of MSE and RMSE results demonstrate that the proposed method has a higher accuracy compared to MLP, as the index values of the proposed method are 0.0018 and 0.0424 while for MLP they increase to 0.0043 and 0.0655, respectively. In other words, applying GA in the proposed method reduces MSE by 58.13% and RMSE by 35.26%. These lower error rates prove that using GA decreases the hand gesture detection errors and boosts the learning process.

To evaluate sensitivity and accuracy, 50 experiments were conducted with the aforementioned configurations. As demonstrated in Figure 11, the sensitivity of the proposed method and MLP are 93.42% and 83.17%, respectively. In terms of accuracy, the proposed method has a 6.65% higher performance – 92.27% compared to 85.62%. These results prove that the proposed combination of GA and MLP delivers more reliable results for detecting human hand gestures.

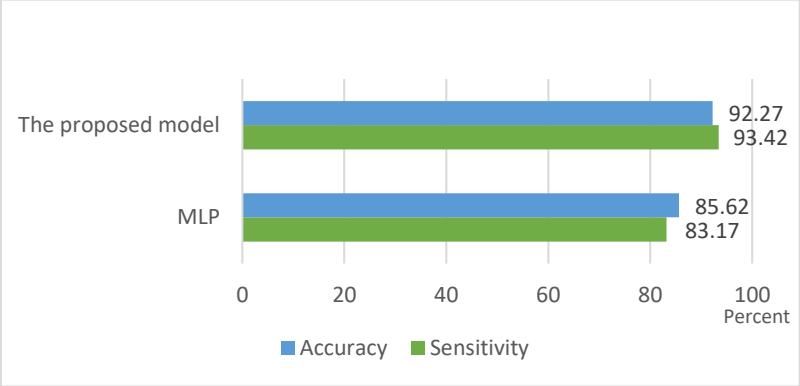

Figure 11. Comparing sensitivity and accuracy of the proposed method and MLP implementationTo the best of our knowledge, there is no other video CAPTCHA algorithm based on hand gesture detection. Therefore, to compare our work with the related ones, the hand gesture detection section of GAPTCHA is compared with the similar algorithms in [52] and [53]. Referring to the RMSE results in Table 1, GAPTCHA outperforms the Redmon et al. [52] work by a big distance. Compared to Li et al. [53] work, our algorithm produces less 1.79% errors and is more efficient.

Table 1. RMSE comparison in hand gesture detection.

Algorithm	RMSE
Redmon et al. (YOLOv3) [52]	0.3894
Li et al. [53]	0.0603
GAPTCHA	0.0424

6. Conclusion

The expansion the Internet created a wider surface for the malicious automated programs (i.e. bots) to run attacks such as Denial of Service (DoS) on the web services. Completely Automated Public

Turing test to tell Computers and Humans Apart (CAPTCHA) in an authentication mechanism that is able to distinguish human users from malicious computer programs.

Various types of CAPTCHAs are designed for different applications; for example, helping people with disabilities, or using on specific platforms. Using the common types of CAPTCHAs (e.g. text-based) may not be a convenient choice for mobile devices due to using them in mobility and small size of screen. In this situation, a hand gesture recognition CAPTCHA can facilitate the use of the device to pass a human authentication challenge.

This research proposes a novel hand gesture recognition CAPTCHA that applies genetic algorithm principles on MLP to develop a reliable method called Gestural CAPTCHA i.e. GAPTCHA. The method shows a set of hand gestures to the user for imitating. GAPTCHA then recognizes the user hand gestures utilizing a set of hand pose features based on the distances between fingertips and center of the hand palm. Genetic algorithm principles such as concentrating on elite population, mutation, and crossover have been used to reduce the detection error rates. The experiments show less 58.13% MSE and 35.26% RMSE hand gesture recognition errors compared to the standard MLP implementation. It also outdoes the related compared algorithms by at least 1.79%. In terms of sensitivity and accuracy, the proposed method reports the values of 93.42% and 92.27%, while for MLP these values are 83.17% and 85.62%, respectively.

References

1. Moradi, M. and Keyvanpour, M., 2015. CAPTCHA and its Alternatives: A Review. *Security and Communication Networks*, *8*(12), pp.2135-2156.
2. Chaeikar, S.S., Alizadeh, M., Tadayon, M.H. and Jolfaei, A., 2021. An intelligent cryptographic key management model for secure communications in distributed industrial intelligent systems. *International Journal of Intelligent Systems*.
3. Von Ahn, L., Blum, M. and Langford, J., 2004. Telling humans and computers apart automatically. *Communications of the ACM*, *47*(2), pp.56-60.
4. Chellapilla, K., Larson, K., Simard, P. and Czerwinski, M., 2005, April. Designing human friendly human interaction proofs (HIPs). In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 711-720).
5. Chaeikar, S.S., Jolfaei, A., Mohammad, N. and Ostovari, P., 2021, October. Security principles and challenges in electronic voting. In *2021 IEEE 25th International Enterprise Distributed Object Computing Workshop (EDOCW)* (pp. 38-45). IEEE.
6. Arteaga, M.V., Castiblanco, J.C., Mondragon, I.F., Colorado, J.D. and Alvarado-Rojas, C., 2020. EMG-driven hand model based on the classification of individual finger movements. *Biomedical Signal Processing and Control*, *58*, p.101834.
7. Lupinetti, K., Ranieri, A., Giannini, F. and Monti, M., 2020, September. 3d dynamic hand gestures recognition using the leap motion sensor and convolutional neural networks. In *International Conference on Augmented Reality, Virtual Reality and Computer Graphics* (pp. 420-439). Springer, Cham.
8. Cobos-Guzman, S., Verdú, E., Herrera-Viedma, E. and Crespo, R.G., 2020. Fuzzy logic expert system for selecting robotic hands using kinematic parameters. *Journal of Ambient Intelligence and Humanized Computing*, *11*(4), pp.1553-1564.
9. Martinelli, Dieisson, Alex Luiz Sousa, Mario Ezequiel Augusto, Vivian Cremer Kalempa, Andre Schneider de Oliveira, Ronnier Frates Rohrich, and Marco Antonio Teixeira. "Remote control for mobile robots using gestures captured by the rgb camera and recognized by deep learning techniques." In *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, pp. 98-103. IEEE, 2019.
10. Chaeikar, S.S., Ahmadi, A., Karamizadeh, S. and Chaeikar, N.S., 2022. SIKM–a smart cryptographic key management framework. *Open Computer Science*, *12*(1), pp.17-26.
11. Khodadadi, T., Javadianasl, Y., Rabiei, F., Alizadeh, M., Zamani, M. and Chaeikar, S.S., 2021, December. A novel graphical password authentication scheme with improved usability. In *2021 4th International symposium on advanced electrical and communication technologies (ISAECT)* (pp. 01-04). IEEE.
12. Chew, M. and Baird, H.S., 2003, January. Baffletext: A human interactive proof. In *Document Recognition and Retrieval X* (Vol. 5010, pp. 305-316). SPIE.

13. Baird, H.S., Coates, A.L. and Fateman, R.J., 2003. Pessimalprint: a reverse turing test. *International Journal on Document Analysis and Recognition*, *5*(2), pp.158-163.
14. Mori, G. and Malik, J., 2003, June. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* (Vol. 1, pp. I-I). IEEE.
15. Baird, H.S., Moll, M.A. and Wang, S.Y., 2005, August. ScatterType: A legible but hard-to-segment CAPTCHA. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)* (pp. 935-939). IEEE.
16. Wang, D., Moh, M. and Moh, T.S., 2020, January. Using Deep Learning to Solve Google reCAPTCHA v2's Image Challenges. In *2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM)* (pp. 1-5). IEEE.
17. Singh, V.P. and Pal, P., 2014. Survey of different types of CAPTCHA. *International Journal of computer science and information technologies*, *5*(2), pp.2242-2245.
18. Bursztein, E., Aigrain, J., Moscicki, A. and Mitchell, J.C., 2014. The End is Nigh: Generic Solving of Text-based {CAPTCHAs}. In *8th USENIX Workshop on Offensive Technologies (WOOT 14)*.
19. Obimbo, C., Halligan, A. and De Freitas, P., 2013. CaptchAll: an improvement on the modern text-based CAPTCHA. *procedia computer science*, *20*, pp.496-501.
20. Datta, R., Li, J. and Wang, J.Z., 2005, November. Imagination: a robust image-based captcha generation system. In *Proceedings of the 13th annual ACM international conference on Multimedia* (pp. 331-334).
21. Rui, Y. and Liu, Z., 2003, November. Artifacial: Automated reverse turing test using facial features. In *Proceedings of the eleventh ACM international conference on Multimedia* (pp. 295-298).
22. Elson, J., Douceur, J.R., Howell, J. and Saul, J., 2007. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. *CCS*, *7*, pp.366-374.
23. Hoque, M.E., Russomanno, D.J. and Yeasin, M., 2006, March. 2d captchas from 3d models. In *Proceedings of the IEEE SoutheastCon 2006* (pp. 165-170). IEEE.
24. Gao, H., Yao, D., Liu, H., Liu, X. and Wang, L., 2010, December. A novel image based CAPTCHA using jigsaw puzzle. In *2010 13th IEEE International Conference on Computational Science and Engineering* (pp. 351-356). IEEE.
25. Gao, S., Mohamed, M., Saxena, N. and Zhang, C., 2015, December. Emerging image game CAPTCHAs for resisting automated and human-solver relay attacks. In *Proceedings of the 31st Annual Computer Security Applications Conference* (pp. 11-20).
26. Cui, J.S., Mei, J.T., Wang, X., Zhang, D. and Zhang, W.Z., 2009, November. A captcha implementation based on 3d animation. In *2009 International Conference on Multimedia Information Networking and Security* (Vol. 2, pp. 179-182). IEEE.
27. Winter-Hjelm, C., Kleming, M. and Bakken, R., 2009. An interactive 3D CAPTCHA with semantic information. In *Proc. Norwegian Artificial Intelligence Symp* (pp. 157-160).
28. Shirali-Shahreza, S., Ganjali, Y. and Balakrishnan, R., 2011. Verifying human users in speech-based interactions. In *Twelfth Annual Conference of the International Speech Communication Association*.
29. Chan, N., 2002. Sound oriented CAPTCHA. In First Workshop on Human Interactive Proofs (HIP), abstract available at http://www. aladdin. cs. cmu. edu/hips/events/abs/nancy_ abstract. pdf.
30. Holman, J., Lazar, J., Feng, J.H. and D'Arcy, J., 2007, October. Developing usable CAPTCHAs for blind users. In *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility* (pp. 245-246).
31. Schlaikjer, A., 2007. A dual-use speech CAPTCHA: Aiding visually impaired web users while providing transcriptions of Audio Streams. *LTI-CMU Technical Report*, pp.07-014.
32. Lupkowski, P. and Urbanski, M., 2008, October. SemCAPTCHA—user-friendly alternative for OCR-based CAPTCHA systems. In *2008 international multiconference on computer science and information technology* (pp. 325-329). IEEE.
33. Yamamoto, T., Tygar, J.D. and Nishigaki, M., 2010, April. Captcha using strangeness in machine translation. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications* (pp. 430-437). IEEE.
34. Gaggi, O., 2022. A study on Accessibility of Google ReCAPTCHA Systems. In *Open Challenges in Online Social Networks* (pp. 25-30).

35. Yadava, P., Sahu, C. and Shukla, S., 2011. Time-variant Captcha: generating strong Captcha Security by reducing time to automated computer programs. *Journal of Emerging Trends in Computing and Information Sciences*, *2*(12), pp.701-704.
36. Wang, L., Chang, X., Ren, Z., Gao, H., Liu, X. and Aickelin, U., 2010, April. Against spyware using CAPTCHA in graphical password scheme. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications* (pp. 760-767). IEEE.
37. Belk, M., Fidas, C., Germanakos, P. and Samaras, G., 2012. Do cognitive styles of users affect preference and performance related to CAPTCHA challenges?. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems* (pp. 1487-1492).
38. Wei, T.E., Jeng, A.B. and Lee, H.M., 2012, December. GeoCAPTCHA—A novel personalized CAPTCHA using geographic concept to defend against 3 rd Party Human Attack. In *2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC)* (pp. 392-399). IEEE.
39. Jiang, N. and Dogan, H., 2015, July. A gesture-based captcha design supporting mobile devices. In *Proceedings of the 2015 British HCI Conference* (pp. 202-207).
40. Pritom, A.I., Chowdhury, M.Z., Protim, J., Roy, S., Rahman, M.R. and Promi, S.M., 2020, February. Combining movement model with finger-stroke level model towards designing a security enhancing mobile friendly captcha. In *Proceedings of the 2020 9th international conference on software and computer applications* (pp. 351-356).
41. Parvez, M.T. and Alsuhibany, S.A., 2020. Segmentation-validation based handwritten Arabic CAPTCHA generation. *Computers & Security*, *95*, p.101829.
42. Shah, A.R., Banday, M.T. and Sheikh, S.A., 2021. Design of a drag and touch multilingual universal captcha challenge. In *Advances in Computational Intelligence and Communication Technology* (pp. 381-393). Springer, Singapore.
43. Luzhnica, G., Simon, J., Lex, E. and Pammer, V., 2016, March. A sliding window approach to natural hand gesture recognition using a custom data glove. In *2016 IEEE Symposium on 3D User Interfaces (3DUI)* (pp. 81-90). IEEE.
44. Hung, C.H., Bai, Y.W. and Wu, H.Y., 2016, January. Home outlet and LED array lamp controlled by a smartphone with a hand gesture recognition. In *2016 IEEE international conference on consumer electronics (ICCE)* (pp. 5-6). IEEE.
45. Chen, Y., Ding, Z., Chen, Y.L. and Wu, X., 2015, August. Rapid recognition of dynamic hand gestures using leap motion. In *2015 IEEE International Conference on Information and Automation* (pp. 1419-1424). IEEE.
46. Panwar, M. and Mehra, P.S., 2011, November. Hand gesture recognition for human computer interaction. In *2011 International Conference on Image Information Processing* (pp. 1-7). IEEE.
47. Chaudhary, A. and Raheja, J.L., 2013. Bent fingers' angle calculation using supervised ANN to control electro-mechanical robotic hand. *Computers & Electrical Engineering*, *39*(2), pp.560-570.
48. Marium, A., Rao, D., Crasta, D.R., Acharya, K. and D'Souza, R., 2017. Hand gesture recognition using webcam. *American Journal of Intelligent Systems*, *7*(3), pp.90-94.
49. Simion G, David C, G Simion, G., David, C., Gui, V. and Caleanu, C.D., 2016. Fingertip-based real time tracking and gesture recognition for natural user interfaces. *Acta Polytechnica Hungarica*, *13*(5), pp.189-204. ui V, Căleanu CD. Fingertip-based real time tracking and gesture recognition for natural user interfaces. Acta Polytechnica Hungarica. 2016 Jan 1;13(5):189-204.
50. Chaudhary, A., Raheja, J.L. and Raheja, S., 2012. A vision based geometrical method to find fingers positions in real time hand gesture recognition. *J. Softw.*, *7*(4), pp.861-869.
51. Benitez-Garcia, G., Olivares-Mercado, J., Sanchez-Perez, G. and Yanai, K., 2021, January. IPN hand: A video dataset and benchmark for real-time continuous hand gesture recognition. In *2020 25th International Conference on Pattern Recognition (ICPR)* (pp. 4340-4347). IEEE.
52. Li, Y.M., Lee, T.H., Kim, J.S. and Lee, H.J., 2021, June. Cnn-based real-time hand and fingertip recognition for the design of a virtual keyboard. In *2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)* (pp. 1-3). IEEE.
53. Redmon, J. and Farhadi, A., 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.