Received November 29, 2020, accepted December 6, 2020, date of publication December 17, 2020, date of current version January 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3045424

Contactless Palm Vein Authentication Using Deep Learning With Bayesian Optimization

MARWA ISMAEL OBAYYA[1,2], MOHAMMED EL-GHANDOUR[2], AND FADWA ALROWAIS[3]
[1]Electrical Engineering Department, College of Engineering, Princess Nourah Bint Abdulrahman University, Riyadh 84428, Saudi Arabia
[2]Electronics and Communication Engineering Department, College of Engineering, Mansoura University, Mansoura 35516, Egypt
[3]Computer Sciences Department, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, Riyadh 84428, Saudi Arabia

Corresponding author: Fadwa Alrowais (fmalrowais@pnu.edu.sa)

This work was supported by the Deanship of Scientific Research, Princess Nourah Bint Abdulrahman University, under Grant 41-PRFA-P-25.

ABSTRACT Among various biometric methods, palm vein authentication has taken significant attention because of its uniqueness, stability and non-intrusiveness. In this paper, we propose a palm vein authentication model using convolutional neural networks (CNN), which is the most popular deep learning architecture and Bayesian optimization. First and foremost, region of interest (ROI) of the palm vein is extracted as an image and filtered by Jerman enhancement filter to enhance the gray levels of the vein patterns. The proposed CNN model allows different numbers of convolutional layers to be added to optimize the network structure. Furthermore, the model is trained with training data to extract the highly representative features of the different classes. The training process is performed at every objective function evaluation, each with a different network structure and training options using a Bayesian optimization algorithm to find the optimal network structure and training options in a search space of possible solutions. The CNN model serves as the palm vein template creator or feature extractor for our identification and verification experiments. Receiver operating characteristic (ROC) curve and equal error rate (EER) were plotted for evaluating the performance of the proposed model. Our proposed method attained an average identification accuracy of 99.4 % and average EER of 0.0683%, which outperforms state-of-the-art palm vein authentication approaches.

INDEX TERMS Biometrics, palm vein, Jerman filter, feature extraction, deep learning, convolutional neural networks, Bayesian optimization.

I. INTRODUCTION

In recent years, there has been a high demand for biometric systems, for personal verification and identification. This has led to the evolution of many applications, such as access control systems, biometric time and attendance systems and law enforcement systems. Palm vein authentication technology is a type of biometric authentication based on the characteristics of the palmar hand veins. Compared with conventional technologies based on biological traits, such as fingerprint [1], iris [2], face [3] and palmprint [4] authentication, the characteristic pattern of the veins inside the palm region of the hand has several advantages. Firstly, owing to the complexity of the veins, it forms a unique shape. Even between twins there are differences within the structure of the pattern. Secondly, the vein pattern remains unchanged throughout one's lifetime. Thirdly, it is difficult to forge, since it needs a special camera to capture the vein pattern and the pattern disappears when the person dies, since it relies on continuing blood flow. The last advantage is that the vein pattern lies internally within the body, which makes it difficult to deform or damage. Therefore, the technique of palm vein authentication offers great research potential and wide application prospects. Recently, the majority of studies in this field have employed a large number of feature extraction methods, which can be classified into two main categories, global feature extraction, and local feature extraction. However, the results from these studies have so far not been very satisfactory. Hence, there is much space for subsequent researchers to come up with additional improvements. Numerous studies on palm vein authentication have been undertaken. Shape-based approaches have focused on the segmentation of the blood vessels in the palm before feature extraction using spatial techniques such as adaptive Gabor filter [5], Gabor Wavelet features [6] and maximal intra-neighbor difference (MIND) [7]. The main drawback of such methods lies in

The associate editor coordinating the review of this manuscript and approving it for publication was Thomas Canhao Xu.

the difficulty in obtaining accurate segmentation of the blood vessels, due to the non-uniform illumination and the low contrast ratio, which in turn affects performance. Statistically-based methods represent the textural properties of vein images in the pixel space of the image. Typically, these methods include Local Binary Patterns (LBP) [8]–[10], modified local binary patterns [11], Local Texture Pattern (LTP) [12], Local Tetra Patterns (LTrP) [13], local directional texture pattern (LDTP) [14] and different variants of local derivative pattern (LDP) [15]. Usually, performance is critically affected by the method used, since pixel-to-pixel processing is highly susceptible to noise in the image. Subspace-based techniques are also used for reduction of dimensionality by projecting the data onto a new, lower-dimensional space created on the training dataset. These methods include Linear Discriminant Analysis (LDA) [16], [17], Principal Component Analysis (PCA) [18], [19], and Fisher Linear Discriminant (FLD) [20], while in [21] the identification accuracy was improved using the Radon transform to extract the principal directions of the veins. Other techniques employ local invariant features, e.g. SURF [22], [23], SIFT [24], and RootSIFT [25], as a way to deal with variations in scale, orientation, and translation between images. All these techniques are non-training-based methods. However, training-based methods have also been adopted, such as in [26], [27].

As reported in literature, almost all methods focus on extracting the low-level features, which are the minor details in the image, such as corners, edges, curves, and lines. Higher-level features make use of low-level features to detect objects and larger shapes in the image. Moreover, feature extraction and selection methods are performed manually, which, in turn, can result in low performance if those features are not carefully picked out and, consequently, will not be very representative. Existing approaches that use deep CNN for palm vein verification and identification, including [27]–[30], are very few and they employ fixed structures and training options for their CNN models. Moreover, studies that employ convolutional neural networks for similar applications with Bayesian optimization, such as [31], [32], use fixed structures or pretrained CNN models, and consider optimizing only the training options, such as learning rate and momentum, while ignoring the possibility of optimizing the network structure, such as the number of convolutional layers. To learn the features perfectly by themselves, CNN architectures have several layers. Hyperparameters are significant as they directly control the behavior of a machine learning model and have a crucial impact on the model performance. However, due to the high dimensionality of the data, it is impossible to tune the hyperparameters by human expertise. Thus, relying on arbitrarily chosen or fixed hyperparameters or recommended structures of CNN does not guarantee obtaining an optimum model.

In this paper, we automate the process of hyperparameter selection using Bayesian optimization and focus on optimizing the number of convolutional layers as well as the training options to obtain the best CNN model, in the hope that the final model will improve the matching results of palm vein authentication. Our motivation and contributions can be summarized as follows:

1. The existing feature-based approaches of palm vein authentication rely on manual feature extraction and selection, which is a time consuming and error-prone process. Here, we automate the feature extraction process using an optimized structure of a deep convolutional neural network.
2. There are only a few deep learning-based feature extraction approaches for palm vein authentication and these approaches depend on fixed structures and training options for their CNN models. Using fixed structures and training options is an error-prone procedure, since it is impossible to predict the correct hyperparameters using human expertise. This is due to the high dimensionality of the hyperparameters and the wide range of each hyperparameter, which in turn will have a negative impact on the performance of the authentication process. To avoid this, we present a new methodology for palm vein authentication using CNN with both the structure and training options optimized by a Bayesian optimization algorithm. The use of Bayesian optimization to find the optimal structure also helps to obtain a model that brings better performance in terms of generalizing the test data. It manages to do this by taking into consideration information on the hyperparameter combinations it has seen so far when selecting the next set of hyperparameters.
3. In the experiments, it was found that the optimum CNN model obtained by Bayesian optimization significantly improves both identification and verification results compared to state-of-the-art methods.

This paper is organized as follows. Section 2 introduces the dataset used in this work. Section 3 describes in detail the main preprocessing steps. Section 4 introduces the deep learning-based automated feature extraction method. Section 5 presents an overview of Bayesian optimization. In section 6, the proposed CNN structure as well as the hyperparameters involved in the optimization process with Bayesian optimization are discussed. Section 7 presents and discusses the experimental results of this work. Section 8 is devoted to the discussion of our approach and main results. Finally, conclusions are drawn in section 9.

II. DATA COLLECTION

To validate the performance of the proposed method, 1200 left-palm vein images from the CASIA Multi-Spectral Palmprint Image Database [33] were used in our experiment. These images were captured in two sessions separated by an interval of more than a month from 100 different volunteers, using self-designed multiple spectral imaging devices operating at wavelengths of 850 and 940 nm. All images in this database were the same size (768×576) and were 8-bit gray level in the JPEG format. In order to simulate an actual application, a certain degree of hand posture variation is allowed during image capturing to create variance among intra-class samples.

III. PALM VEIN PREPROCESSING

Palm vein pre-processing involves two main procedures. The first procedure is to set up a coordinate system based on the key points between fingers to align palm vein images and segment the central part of the palm vein image for feature extraction.

The second procedure aims to highlight the low contrast vein pattern. Since the vein pattern we are interested in is located in the central part of the hand, the palm vein images have to undergo a segmentation process to extract this ROI. Here, the segmentation process involves five main steps: binarizing the palm vein images, locating the centroid and the contour of the hand, detecting key points, establishing a coordinate system, and, finally, extracting the ROI. Before initializing the actual segmentation of the palm, the wrist part is located on the right-hand side of the palm vein image. Fig. 1 (a) should be excluded due to the irregular illumination in this area. Thus, this area is zero-padded, and the resulting image is converted into binary form using the Otsu thresholding method, as shown in Fig. 1 (b).

After locating the centroid (C) and the boundary points of the hand, as shown in Fig. 1(c), the Euclidean distance between C and every boundary point is estimated to obtain the radial density function, RDF, as shown in Fig. 1 (d).

The 4 maxima and 3 minima correspond to the 4 fingertips and the 3 valleys, respectively, as shown in Fig 1 (e). Let θ be the angle between the line $P\bar{1}P_2$ and the vertical line L, as shown in Fig. 1 (f). The hand image needs to be rotationally normalized to vertically align the 2 points: P1 and P2. This is done by rotating the palm to the left by angle θ, as shown in Fig. 1 (g).

$$\theta = \tan^{-1}\left(\left(Y_{P_2} - Y_{P_1}\right) / \left(X_{P_2} - X_{P_1}\right)\right), \tag{1}$$

where $\left(X_{P_1}, Y_{P_1}\right)$ and $\left(X_{P_2}, Y_{P_2}\right)$ are the coordinates of points P_1 and P_2, respectively. For ROI extraction, some methods rely only on points P1 and P2 to define the boundaries of the ROI. However, these methods do not take into consideration the additional information existing along the borders of the palm. Thus, we need to define a new set of points: P3, P4, P5, and P6 to extract a larger ROI encompassing the additional information-rich area at the boundaries of the palm. On the little finger, we define a point P6, which is located behind P8 at a boundary distance equal to that between P2 and P8. Similarly, P3 is located behind P7 at a boundary distance equal to that between P1 and P7. Next, we need to determine P4 and P5 to define the boundaries of the ROI. The X and Y coordinates of P4 are located in the midpoint between those of P1 and P3, respectively. Similarly, P5 coordinates are located at the midpoint between those of P2 and P6. Finally, The ROI is defined by the square outlined in yellow, as shown in Fig. 1 (h), whose side length is equal to the vertical distance

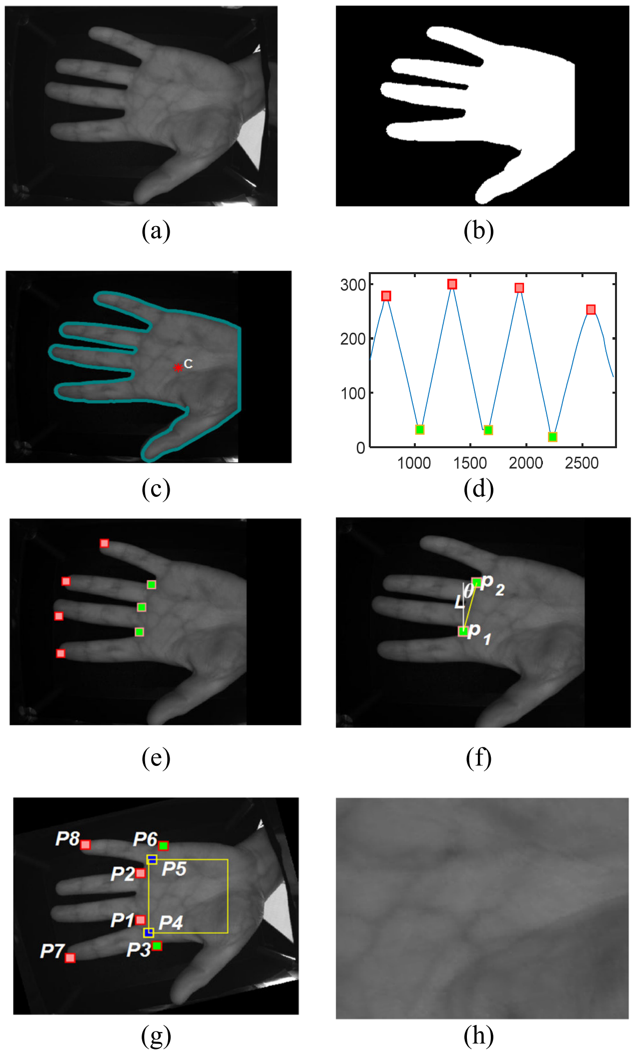


FIGURE 1. **(a) Raw palm vein image, (b) Thresholding using Otsu's method, (c) Locating boundary points and the centroid, (d) RDF distribution (e) Peak and valley points, (f) The angle θ between $P\bar{1}P_2$ and vertical line L, (g) Locating p4 and p5 and defining the ROI boundaries, (h) The segmented ROI.**

between P4 and P5. More raw palm vein images and their corresponding ROIs are shown in Fig.2.

The second step in image preprocessing involves rendering the ROI more suitable for feature extraction and matching. As shown in Fig. 1 (h), the distribution of gray levels within the palm region is excessively concentrated and the contrast between the veins and the background is very low. This makes feature extraction a strenuous task. We thus need to amplify the local intensity of the veins with respect to the background. The Jerman multiscale vesselness filter [34] is adopted to perform this task. This is an improved version of the filters proposed in preceding related work by Frangi *et al.* [35], Satos *et al.* [36], and Li *et al.* [37], for example. All these filtering techniques are Hessian-based methods in which the enhancement function uses the Hessian eigenvalues to boost the intensity values of elongated structures such as veins. The eigenvalues are sorted in ascending order according to their

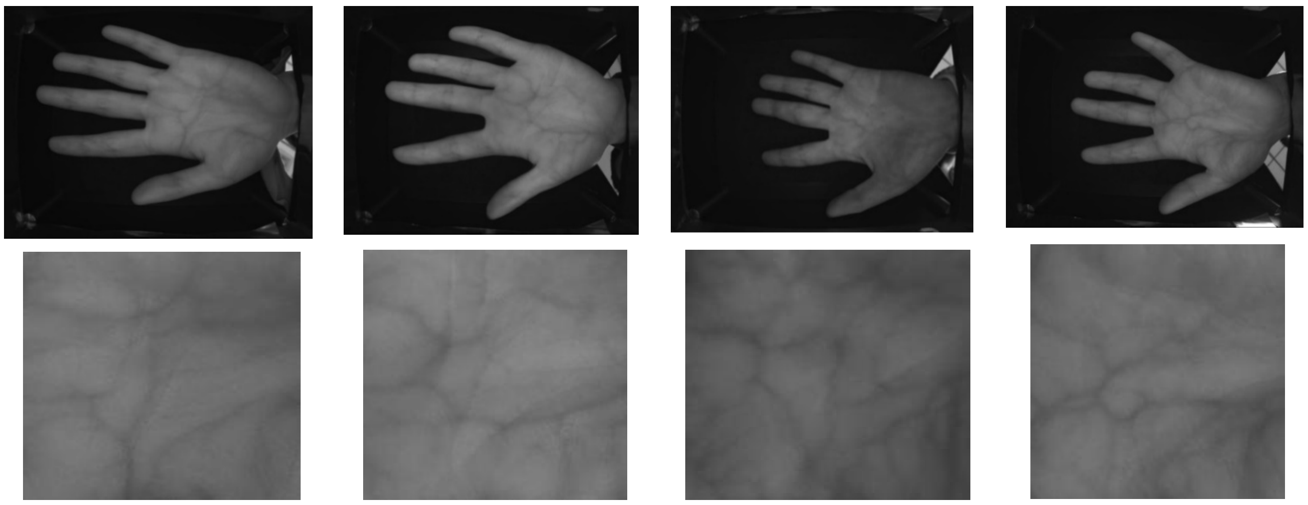


FIGURE 2. Raw palm vein images (first row) and their corresponding ROIs (second row).

absolute values: $\lambda_1 < \lambda_2$. Let $I(X)$denote the 2-dimensional palm vein image at coordinate $X = [x_1, x_2]$. The Hessian value of $I(X)$ at X and scale s is then represented by a 2×2 matrix:

$$H_{ij}(X,s) = s^2 I(X) * \frac{\partial^2}{\partial x_i \partial x_j} G(X,s) \quad for\ i,j = 1,2, \tag{2}$$

where $G(X,s) = \left(2\pi s^2\right)^{-1} exp(-XX^T/2s^2)$ is a bivariate Gaussian filter and $*$ denotes convolution. The Jerman enhancement function is computed as:

$$v = \begin{cases} 0 & if\ \lambda_2 \le 0 \vee \lambda_\rho \le 0, \\ 1 & if\ \lambda_2 \ge \lambda_\rho/2 > 0, \\ \lambda_2^2 \left(\lambda_\rho - \lambda_2\right) \left(\frac{3}{\lambda_2 + \lambda_\rho}\right)^3 & otherwise \end{cases} \tag{3}$$

where

$$\lambda_\rho(s) = \begin{cases} \lambda_3 & if\ \lambda_3 > \tau\, max_x \lambda_3(X,s) \\ \tau\, max_x \lambda_3(X,s) & if\ 0 < \lambda_3 \le \tau\, max_x \lambda_3(X,s) \\ 0 & otherwise \end{cases} \tag{4}$$

λ_3 is equal to λ_2 in case of 2D images and τ is a parameter between 0.5 and 1 that controls the uniformity of the filter output, where choosing a small τ results in more intense output values. The Jerman filter was applied to all the ROIs with τ set to 1 and different Gaussian scales were used, ranging from 1 to 3 with a step size of 0.5. The maximum filter response values of all the 5 scales (Fig. 3 (a) to (e)) are taken to form the final output, as shown in Fig. 3 (f). It is now clear that that the contrast is very high between the veins and the background compared with the unprocessed ROI in Fig. 1 (h). In order to speed up the training of the CNN model, the filtered image is scaled down to 40×40 before passing it as input to the CNN.

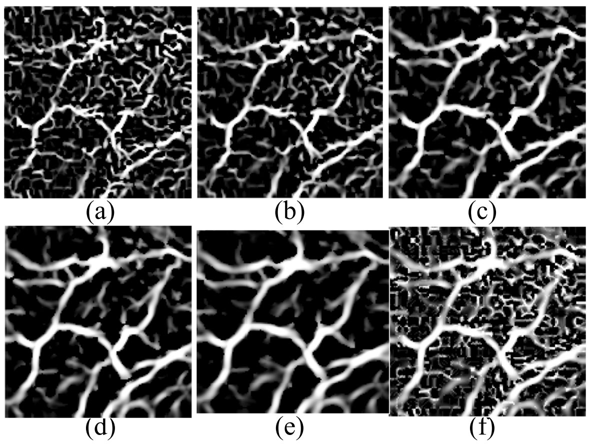


FIGURE 3. Example of Jerman filtering being applied on the ROI in Fig.1. (a) to (e) showing the filter output using five different scales from 1 to 3 with a step size of 0.5. (f) The result of taking the maximum value obtained from the different scales.

To demonstrate the importance and effectiveness of the preprocessing step using the Jerman filter, the raw ROIs which have not been preprocessed were tested for matching in our experiment against the preprocessed ones with the Jerman filter.

IV. FEATURE EXTRACTION AND MATCHING USING CNN

Deep learning [38]–[40] is a subset of machine learning in artificial intelligence. It allows computational models that comprise several processing layers to learn representations of data with multiple levels of abstraction. These methods have shown remarkable improvements in state-of-the-art applications such as visual object identification, speech recognition, object detection, drug discoveries, and genomics. Deep learning utilizes the backpropagation optimization algorithm to learn about the complex structure in large data sets by evaluating the amount of parameter change a machine should

Algorithm 1 Palm Vein Preprocessing

```
Segmentation:
Input: palm vein image
Output: ROI
 1. Remove the irregular illumination region by setting pixel values in the rectangular region with $X$ values > 650 to 0, Fig.1 (b)
 2. Apply Otsu's thresholding, Fig.1 (b)
 3. Locate the centroid $c$ and boundary points $p_n$ of the hand, Fig.1 (c).
 4. Compute the Euclidean distance between the centroid and every boundary point to obtain the RDF as follows: $d_n = \sqrt{\left((X_c, Y_c) - \left(X_{p_n}, Y_{p_n}\right)\right)^2}$, Fig.1 (d)
 5. Locate the minima and maxima of the RDF to locate the hand valleys and peaks, respectively, Fig.1 (e).
 6. Draw a line between the two valley points $p_1$ and $p_2$ to compute the angle $\theta$ between $\overline{p_1p_2}$ and the vertical line $L$ using Equation (1), Fig.1 (f).
 7. Rotationally normalize the hand by rotating it to the left by $\theta$, Fig.1(g).
 8. Find $p_3$ and $p_6$, where the boundary distance between $p_8$ and $p_6$, $d_b(p_8, p_6)$ is set to be equal to $d_b(p_8, p_2)$ and $d_b(p_7, p_3)$ is set to be equal to $d_b(p_7, p_1)$
 9. Find $p_4$ and $p_5$ as follows: $p_4 = \frac{(X_{p_1}, Y_{p_1}) + (X_{p_3}, Y_{p_3})}{2}$, $p_5 = \frac{(X_{p_2}, Y_{p_2}) + (X_{p_6}, Y_{p_6})}{2}$
10. Locate the ROI by identifying the square region whose length is equal to the distance between $p_4$ and $p_5$

    ROI vein enhancement:

    Input: unprocessed ROI image, $I(X)$.
    Output: vein-enhanced ROI, $v$.
11. For each image pixel location, $X$, compute the Hessian matrix $H_{ij}(X, S)$, $i, j = 1, 2$ using Equation (2) with 5 different Gaussian filter scales ranging from 1 to 3 with a step size equal to 0.5, $S = \{1, 1.5, 2, 2.5, 3\}$.
12. Compute the eigenvalues ($eigH_{ij}(X, S) \longrightarrow \lambda_{1,2}$) of each Hessian matrix in each scale.
13. Sort the eigenvalues in an ascending order according to their magnitude (i.e., $|\lambda_1| < |\lambda_2|$)
14. For each scale, $s$, use the Jerman enhancement function $v^s_{\tau=0.5}$ (Equation (3)) and set $\tau$ to 0.5 in Equation (4) to compute the response of the Jerman filter at every pixel location to finally obtain 5 different responses (Fig.3 (a-e))
15. Compute the final filtered image (Fig.3 (f)) by taking the maximum filter response values of all the 5 scales as follows: $v_{\tau=0.5} = \max_{s=1,1.5,2,2.5,3}\{v^s_{\tau=0.5}\}$
```

apply to determine the level of representation in each layer from that in the preceding layer. A convolutional neural network is a widely recognized deep learning method. Deep CNNs have shown success in problems of classification and recognition [41], [42]. They learn to perform classification tasks directly from image pixels, without the need for manual feature extraction prior to presenting the training examples. A CNN is made up of two main sections: The first section performs feature extraction using a series of convolutional layers, followed by pooling and activation of the function layers. The second section does the job of a normal classifier with a fully connected layer, softmax layer, and a classification output layer. In this work, the output classification layer consists of 100 neurons, corresponding to the number of clients participated in this experiment. These neurons enable 100 different possible binary vectors, each with all zeros, except in the place of the client it represents, which is 1.

For each client, 8 images were randomly chosen for training the CNN, so a total of 800 images were used in the training phase. The remaining 400 images were split equally into validation and test sets to evaluate the model hyperparameters and the performance of the optimized fine-tuned CNN, respectively.

V. AN OVERVIEW OF BAYESIAN OPTIMIZATION

Almost all optimization problems in machine learning are black-box optimization problems, in which the objective function for which we are trying to find the absolute minima cannot be expressed in closed form and its derivative cannot be evaluated [43], [44]. The only possible way to evaluate the function is to sample at a point x and obtain a possibly noisy output.

The aim of the research reported in this paper is to find the CNN hyperparameters that yield the minimum validation error, in the hope that the final model generalizes well to the test data. Hyperparameter optimization for an unknown function $f(x)$ is represented in the form:

$$x^* = \arg\max f(x) \tag{5}$$

where X refers to the search space of x. Bayesian optimization is an approach derived from Bayes' theorem to guide the search towards the minimum of an objective function. Bayes theory states that a given observation point E, the posterior probability $P(M|E)$ of a model M is proportional to the likelihood $P(E|M)$ of observation E, given model M, multiplied by the prior probability of the model $P(M)$:

$$P(M \mid E) \propto P(E \mid M) P(M) \tag{6}$$

The idea behind Bayesian optimization is that it uses the prior distribution of the objective function $f(x)$ as well as the observation points obtained from the previous trials of training the model to obtain a posterior distribution of the model. The posterior information is then utilized to select the next sample points that are expected to minimize $f(x)$ [45]. The selection of these new sample points is performed according to criteria represented by an acquisition function, A, to locate the new points where the expected improvement is most probable. To reduce the number of total sampling points before the end of the optimization process, the acquisition function takes into consideration both exploitation (sampling points that are expected to yield low output value) and exploration (sampling points from areas of a high degree of uncertainty) [46].The search space in a Bayesian optimization algorithm is realized by using a Gaussian process. A Gaussian process (GP) is a process in which any subset of observations is assumed to follow a Gaussian distribution, with mean $m : \mathrm{x} \rightarrow \mathbb{R}$ and covariance function $\mathrm{k} : \mathrm{x} \times \mathrm{x} \rightarrow \mathbb{R}$. The Gaussian process can be formulated as:

$$f(x) \sim GP(m(x), k(x, x')) \tag{7}$$

In the Gaussian process, $f(x)$ is not just an estimate for that point but a one-dimensional Gaussian distribution function over all possible values of $f(x)$. For convenience, we assume that the Gaussian process mean function $m(x) = 0$ and that its covariance matrix is the commonly used squared exponential covariance function, denoted as:

$$k(x_i, x_j) = e^{\left(-\frac{1}{2}\|x_i - x_j\|^2\right)} \tag{8}$$

where x_i and x_j are the i^{th} and j^{th} samples. The closer x_i and x_j are to one another, the higher their covariance $k(x_i, x_j)$. The posterior distribution of $f(x)$ is computed as follows:

1. Given a set of observations $D_{1:t} = \{x_n, f_n\}_{n-1}^{t}, f_n = f(x_n)$ assume the function f values are samples from a multivariate normal distribution $f \sim N(0, K)$, where

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_1) & \dots & k(x_1, x_t) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_t) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_t, x_1) & k(x_t, x_2) & \dots & k(x_t, x_t) \end{bmatrix} \tag{9}$$

2. At the newly computed sample point x_{t+1}, compute $f(x_{t+1})$ as follows:

$$\begin{bmatrix} f_{1:t} \\ f_{t+1} \end{bmatrix} \sim N\left(0, \begin{bmatrix} K & k \\ k^T & k(x_{t+1}, x_{t+1}) \end{bmatrix}\right) \tag{10}$$

where $f_{1:t} = [f_1, f_2, \cdots, f_t]$ and
$k = [k(x_{t+1}, x_1)\, k(x_{t+1}, x_2) \cdots k(x_{t+1}, x_t)]$

According to the Gaussian process, $[f_{1:t}, f_{t+1}]$ follows the $t + 1$ multivariate normal distribution, i.e. $f_{t+1} \sim N(\mu_{t+1}, \sigma_{t+1}^2)$ where,

$$\mu_{t+1}(x_{t+1}) = -k^T K^{-1} f_{1:t} \tag{11}$$
$$\sigma_{t+1}^2(x_{t+1}) = -k^T K^{-1} k + k(x_{t+1}, x_{t+1}) \tag{12}$$

The mean and variance of f_{t+1} indicate that the Gaussian process does not estimate an exact value but a one-dimensional probability distribution for f_{t+1}. There are different kinds of acquisition functions, such as the lower confidence bound [47], the probability of improvement [48], and the expected improvement [49]. For example, the degree of improvement, I, according to the expected improvement acquisition function, is defined as:

$$I(x) = \max\{0, \max\{0, f_{t+1} - f(x^+)\} \tag{13}$$

where f_{t+1} is the function value at the next sampled point that is expected to improve the pproximation of the function and $f(x^+)$ is the maximum observed point after obtaining t samples. The next sampled point, x_{t+1}, is given by:

$$x_{t+1} = arg \max_x E(\max\{0, f_{t+1} - f(x^+)\}) \tag{14}$$

Consequently, the probability density function of I follows ~~the~~ normal distribution, with mean $\mu(x) - f(x^+)$ and standard deviation equal to $\sigma^2(x)$ and is given by:

$$f(I) = \frac{1}{\sqrt{2\pi}\sigma(x)} e^{\left(-\frac{\mu(x) - f(x^+) - I}{2\sigma^2(x)}\right)}, \quad I > 0 \tag{15}$$

The expected improvement $E(I)$ is the integral over the function in equation 15:

$$E(I) = \int_{I=0}^{\infty} I f(I)\, dI = \sigma(x)[Z\emptyset(Z) + \varphi(Z)] \tag{16}$$
$$Z = \frac{\mu(x) - f(x^+)}{\sigma(x)} \tag{17}$$

where $\emptyset(\cdot)$ and $\varphi(\cdot)$ are the cumulative distribution function and probability density function of the standard normal distribution, respectively. Algorithm 2 summarizes the steps of the Bayesian optimization algorithm.

Algorithm 2 Bayesian Optimization Algorithm

```
1. For t = 1, 2, . . .
2. Find $\boldsymbol{x_i}$ by optimizing the acquisition function $\boldsymbol{A}$ over
   function $\boldsymbol{f}$, $\mathbf{x_i} = \mathbf{arg\,min}_{\boldsymbol{x}} \boldsymbol{A}(\boldsymbol{x}|\boldsymbol{D_{1:t-1}})$
3. Sample the objective function $\boldsymbol{y_t} = \boldsymbol{f}(\boldsymbol{x_t})$
4. Augment the data $\boldsymbol{D_{1:t}} = \{\boldsymbol{D_{t-1}}(\boldsymbol{x_t}, \boldsymbol{y_t})\}$ and update the
   posterior of function $\boldsymbol{f}$
5. End for.
```

VI. HYPERPARAMETER TUNING WITH BAYESIAN OPTIMIZATION

The CNN architecture as well as the training options need to be estimated ahead of the training process. Figuring out these hyperparameters to minimize the matching error and avoid overfitting is not a straightforward task and ~~it~~ can consume a lot of computational time if we decide to use a brute-force search before obtaining satisfactory matching results. Our plan is to automate the process of tuning the hyperparameters with the help of Bayesian optimization, in which the optimal values are reached within a limited number of search attempts, hence guaranteeing the best CNN model will be obtained. Bayesian optimization uses an n-dimensional multivariate Gaussian model to construct a posterior distribution over the objective function, given a set of observations. In this work, n is the number of hyperparameters that need to be optimized, while the observation is the validation error computed after training the CNN using the newly sampled hyperparameters for each iteration (objective function evaluation).

A typical CNN model consists of an input layer and classification output layer, in addition to a number of hidden layers. The hidden layers comprise a series of convolutional layers which are followed by batch normalization layers, and the

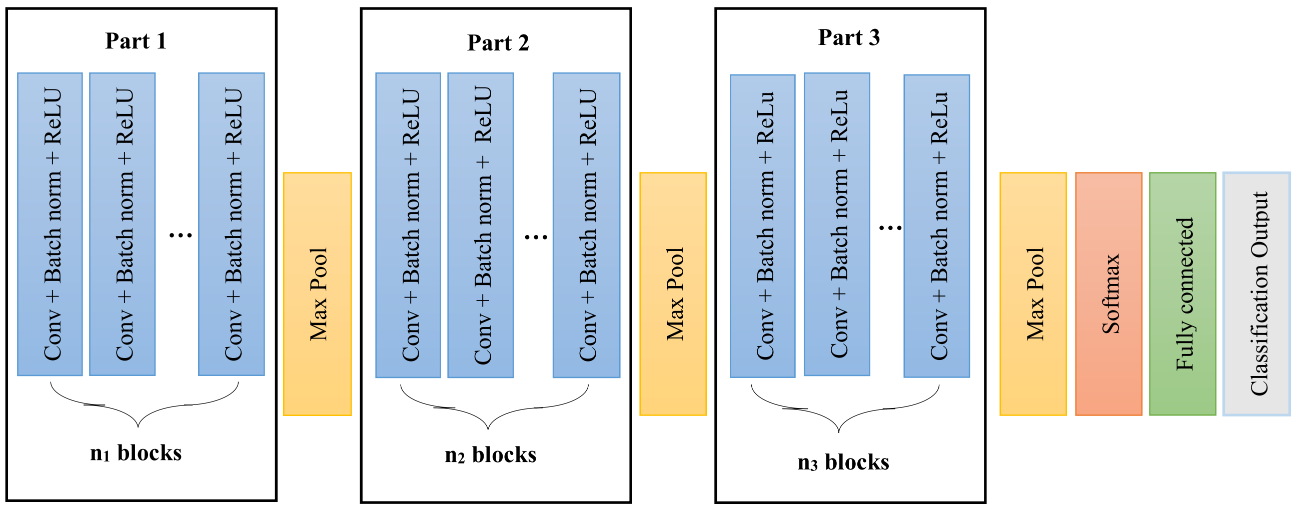


FIGURE 4. **The proposed structure of the CNN model for optimization.**

batch normalization layers are followed by a ReLU layers and then the pool layers come at the end. To optimize our network model, we need to decide the optimum number of these convolutional layers. The first set of convolutional layers learn the low-level features and the following convolutional layers learn the mid-level and high-level features.

Thus, we propose dividing the network into three main parts, where each part is followed a max pool layer and is made up of a number of blocks. Each block has one convolutional layer, one batch normalization layer and one ReLU layer. Fig. 4 shows the structure of the proposed model, with n_1, n_2, and n_3 blocks defining the number of blocks or part depth which equals the number of convolutional layers in that part. The optimization algorithm takes the number of blocks in each part (part depth) as well as the hyperparameters of the training options as input to optimize the network. Thus, the first 3 hyperparameters that need to be optimized are the number of blocks or convolutional layers in each part of the network. The number of convolutional kernels (filters) in each part is set to be inversely proportional to the square root of the part depth. Consequently, for each iteration, the number of parameters and the computational time are almost equal, regardless of the differences in part depths between the network models evaluated by Bayesian optimization. The number of convolutional layer filters is increased by a factor of 2 each time the spatial size is scaled down by a factor of 2 according to the max pool layer. This ensures a nearly equal amount of computation in each convolutional layer. This is also important since the deeper the convolutional layer is, the more features the network needs to learn. The number of convolutional layer filters in each part of the network can be formulated as follows:

$$No.conv\ filters = floor\left(\frac{\beta}{\sqrt{part\ depth}} \times 2^{part\ number-1}\right) \tag{18}$$

where the value of β is arbitrarily set to 10 and the part number is either 1, 2, or 3, depending on the part where the convolutional layer is located. The other 3 hyperparameters control the training process. These parameters are:

A. LEARNING RATE

During training of a CNN model, each convolutional layer's weight is updated to minimize the loss, with a step size referred to as the learning rate [50]. In deep convolutional neural networks, the stochastic gradient descent (SGD) is used as an optimizer to update the weights of the convolutional layers. It updates the weights in the opposite direction to the gradient of the loss function [51]. The new value of a learnable parameter (weight and bias) at iteration no. $l + 1$ for a single update, according to the SGD, is formulated as follows:

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) \tag{19}$$

where l is the iteration number, α is the learning rate and $\nabla E(\theta_l)$ is the gradient of the loss function. α takes on values between 0 and 1. In this experiment we defined a range between 10^{-6} and 1 for this hyperparameter. We did not include values lower than 10^{-6} within the range, since a learning rate that is too small causes the network to either train very slowly, never converge, or get stuck on a suboptimal solution [52].

B. WEIGHT DECAY

Adding a weight decay term to the loss function $E(w_l)$ in order to penalize large weights is a way to mitigate overfitting and improve generalization [53], [54]. The most commonly used weight decay techniques in machine learning are $l1$ and $l2$ regularization. These techniques are also referred to as weight decay methods, since they work on shrinking the weights to smaller values. $l2$ regularization forces the weights to drop towards 0 but not exactly zero, while $l1$ encourages most weights to drop to exactly 0 [55]. For this reason, $l2$ is adopted as a regularizer for our network model. The new regularized loss function with the regularization term is given as follows:

$$E_R(\theta) = E(\theta) + \lambda\Omega(w) \tag{20}$$

where w is the weight vector, λ is the regularization factor (coefficient), and the regularization function Ω(w) is

$$\Omega(w) = \frac{1}{2}w^T w \tag{21}$$

λ takes on values on the logarithmic scale between 0 and 0.1, for example 0.1, 0.001, 0.0001 [56]. Studies in the literature have assigned values for λ between 10^{-3} and 10^{-6} [57]–[59] on the logarithmic scale. We considered these values in the range we selected for this hyperparameter and defined the range to be between 10^{-10} and 10^{-1} on the logarithmic scale. This wide range gives the BO a large search space to pick up the most accurate value for this hyperparameter without being fully dependent on the recommended values or ranges obtained from previous studies.

C. MOMENTUM

The SGD algorithm can exhibit high-frequency oscillations along the path towards the minimum error [53], [60]. Adding a momentum term to the weight update is an effective way to damp these oscillations and, as a result, minimize the number of iterations and speed up convergence. The SGD with momentum (SGDM) is formulated as follows:

$$\theta_{l+1} = \theta_l - \alpha\nabla E(\theta_l) + \gamma(\theta_l - \theta_{l-1}) \tag{22}$$

where γ is the momentum factor, which determines the influence of the previous gradient step on the current iteration. In this experiment with BO, we decided not to depend only on the values of γ recommended in the literature, but to cover the whole range of γ lying between 0 and 1.

Regarding the part depth, the minimum value of the part depth is 1 while the maximum number can take on any integer value. Thus, we run Bayesian optimization multiple times with different ranges of part depths for each run. This helps to obtain an intuition about the CNN behavior with various ranges of part depth. Thus, we can find the optimum range that yields the minimum test error.

Let $\mathcal{M}$ denote our CNN model with the 6 (N = 6) hyperparameters to be optimized, where M_n, denotes the domain of the n^th^ hyperparameter. We denote the space of the hyperparameter configuration as $X = X_1 \times X_2 \times \ldots X_6$. The six-dimensional vector of hyperparameters is denoted by $\nu \in X$:$\nu = \{n_1, n_2, n_3, \alpha, \lambda, \gamma\}$ and $\mathcal{M}$, with its hyperparameters represented by ν, is denoted as $\mathcal{M}_\nu$.

Given the training and validation data, $\mathcal{D}$, our goal is to find

$$\nu^* = \arg\min_{\nu\in X} \mathbb{E}_{(D_{train}.D_{valid})} \sim \mathrm{V}(\mathcal{L}, \mathcal{M}_\nu, D_{train}.D_{valid}) \tag{23}$$

where $\mathrm{V}(\mathcal{L}, \mathcal{M}_\nu, D_{train}.D_{valid})$ is the validation protocol used to measure the loss ($\mathcal{L}$) of $\mathcal{M}_\nu$ on the validation data at the output of the softmax layer. The holdout method is used as the validation protocol to measure the loss on the validation data. The softmax and cross-entropy loss functions are defined as follows:

Algorithm 3 Optimization of CNN Hyperparameters Using Bayesian Optimization

Input: some observations, D (CNN hyperparameters and the corresponding validation error, (X_t, Y_t))
Hyperparameters: part1 depth n_1, part2 depth n_2, part3 depth n_3, learning rate α, momentum γ, weight decay λ
Output: validation error Y_t

```
1.  Initialize 6 random hyperparameters
    $X_1 = \{n_1^1, n_2^1, n_3^1, \alpha^1, \gamma^1, \lambda^1\}$
2.  For t = 1 ⟶ 40 (max number of objective function evaluations)
3.      Use the proposed CNN structure in Fig.4 to create a CNN using the hyperparameters $X_t$
4.      Set the number of convolutional kernels in each convolutional layer according to Equation (18)
5.      Train the CNN using SGD algorithm with a mini-batch size of 100 samples and max number of epochs of 40.
6.      Compute the validation error on the validation set as follows:
            $Y_t = \frac{number\ of\ misclassified\ samples}{200}$
7.      If $Y_t = 0$ then
8.          Stop the optimization process.
9.          Go to step 15.
10.     Else
11.         Update the 6-D surrogate model $D_t =$ using Equation (10),
                $\{D_{1:t-1} \cup (X_t, Y_t)\}$
12.         Sample 6 new hyperparameter $X_{i+1}$ values by optimizing the expected improvement function
            $A, X_{i+1} = \arg A(X|D)$ using Equation (16)
13.     End if
14. End for
15. Select the CNN with hyperparameters that yield the lowest validation error, $Y_t$
16. Return $Y_t$
```

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^{J=100} e^{z_j}} \tag{24}$$

$$\mathcal{L} = \frac{-1}{N}\sum_{n=1}^{N=200}\sum_{j=1}^{J=100} T_i \log y_i \tag{25}$$

where z_i is the output of the final max pool layer, y_i is the output of the softmax layer, T_i is the target value, J represents the total number of elements in y, and N is the total number of validation examples (N = 200). The next set of hyperparameter values is computed according to the expected improvement, as follows:

$$\mathbb{E}\left[\mathbb{I}\left(v\right)\right]=\mathbb{E}\left[\max\left(f_{min}-\epsilon,0\right)\right] \tag{26}$$

$$\mathbb{E}\left[\mathbb{I}\left(v\right)\right]=\left(f_{min}-\mu\left(v\right)\right)\Phi\left(\frac{f_{min}-\mu\left(v\right)}{\sigma}\right)+\sigma\phi\left(\frac{f_{min}-\mu\left(v\right)}{\sigma}\right) \tag{27}$$

where f_{min} is the best observed validation error so far that has been obtained from training the CNN with the selected hyperparameters, and ϵ is the predicted mean function value of the surrogate model at the next v that is expected to improve the validation error of the CNN. The validation error can be expressed as:

$$validation\ error=\frac{No.\ of\ misclassified\ validation\ samples}{Total\ number\ of\ validation\ samples} \tag{28}$$

Thus, we defined an optimization problem with a six-dimensional (6-D) objective function using the palm vein training and validation data as fixed inputs with a mini-batch size equal to 100 images. The maximum number of epochs was set to 40. For each objective function evaluation, the objective function performs the training of the CNN using the SGD algorithm as an optimizer with the newly sampled hyperparameters and returns the matching error on the validation data. Experimental settings are shown in Table 1, encompassing the parameter settings for both the training and optimization.

The optimizer is set to terminate when the validation error equals 0 or if the number of iterations (function evaluations) reaches 40. The optimizer starts with a Gaussian model as an approximation estimate of the objective function and then updates this model after every iteration towards the shortest path to the minimum value of the validation error. A flow chart of the CNN hyperparameter optimization process is shown in Fig. 5.

VII. EXPERIMENTAL RESULTS

The results of the optimum fine-tuned CNN model hyperparameters obtained from running the optimizer five times are shown in Table 2, in which each experiment has a different range of part depths. The first experiment, with a range between 1 and 3 for each part depth, yielded the best CNN model. This model converged faster than the other models in the remaining four experiments and had the lowest validation and test error. Although the training accuracy reached 100% in all five experiments, adding more convolutional layers through increasing the part depth range did not improve the performance but instead caused the network to overfit to the training data.

After 34 objective function evaluations in the search space of possible solutions using Bayesian optimization, the best CNN model defined by the minimum validation error was finally obtained when the Jerman filter was applied in the preprocessing step, as shown in Table 4 in the last raw. Table 4 records the hyperparameters and validation errors reported from 34 function evaluations. The Bayesian optimizer starts by generating six random hyperparameters, recorded in the first row in Table 4, which are the depth of each part, and the three training option hyperparameters. The objective function then trains a CNN model using these hyperparameters and returns the validation error on the validation data. The validation error, which is recorded in the first column in Table 4, is then used to update the 6-D surrogate model created by the optimization algorithm. The optimizer then samples six new hyperparameters, which are recorded in the second row in Table 4, performs the training and reports the validation error. This iterative process continues with different hyperparameters for each iteration until the optimizer stops searching for new hyperparameters when the validation error reaches exactly 0. This occurred at the 34th iteration, as shown in the last raw in Table 4. In the experiment conducted without the preprocessing step, the optimizer had to stop once it completed the last function evaluation, according to the stopping criterion. The minimum validation error was obtained at the 38th objective function evaluation and it reached a minimum value equal to 0.025. In Fig. 8 and Fig. 9, the values of the minimum observed objective and the estimated minimum objective curves are plotted for both experiments. The estimated min objective refers to the lowest estimated mean value according to the latest approximate model of the objective function, while the min observed objective refers to the lowest returned validation error value resulting from evaluations of the objective function. It can be noticed from Fig. 8 that at the 34th iteration, the validation error reached exactly 0, which means that all samples in the validation data were correctly classified. On the other hand, the validation error reached 0.025 at the 38th iteration when removing the preprocessing step with the Jerman filter, as shown in Fig. 9. It can be seen in Fig. 6 that the optimal network architecture obtained from the optimization problem in the first experiment, using the Jerman filter, has one convolutional layer in each part, while in the second experiment, without the Jerman filter, the optimal structure is more complex, as shown in Fig. 7. It has two convolutional layers in the first part and one convolutional layer in each of the other two parts. The detailed structure of the fine-tuned CNN of the proposed method with Jerman filter is illustrated in Table 5. The optimal part depths and training options for both experiments are shown in Table 3. Both models were then tested for the palm vein template creator, which is the feature extractor, for the two subsequent experiments:

A. PALM VEIN IDENTIFICATION

This is the process of questioning the biometric system regarding who a certain palm vein image belongs to. Therefore, this process involves comparing this image against every other image in the database and finding the correct match. Identification accuracy is obtained by computing the percentage of correctly classified samples as given in the following

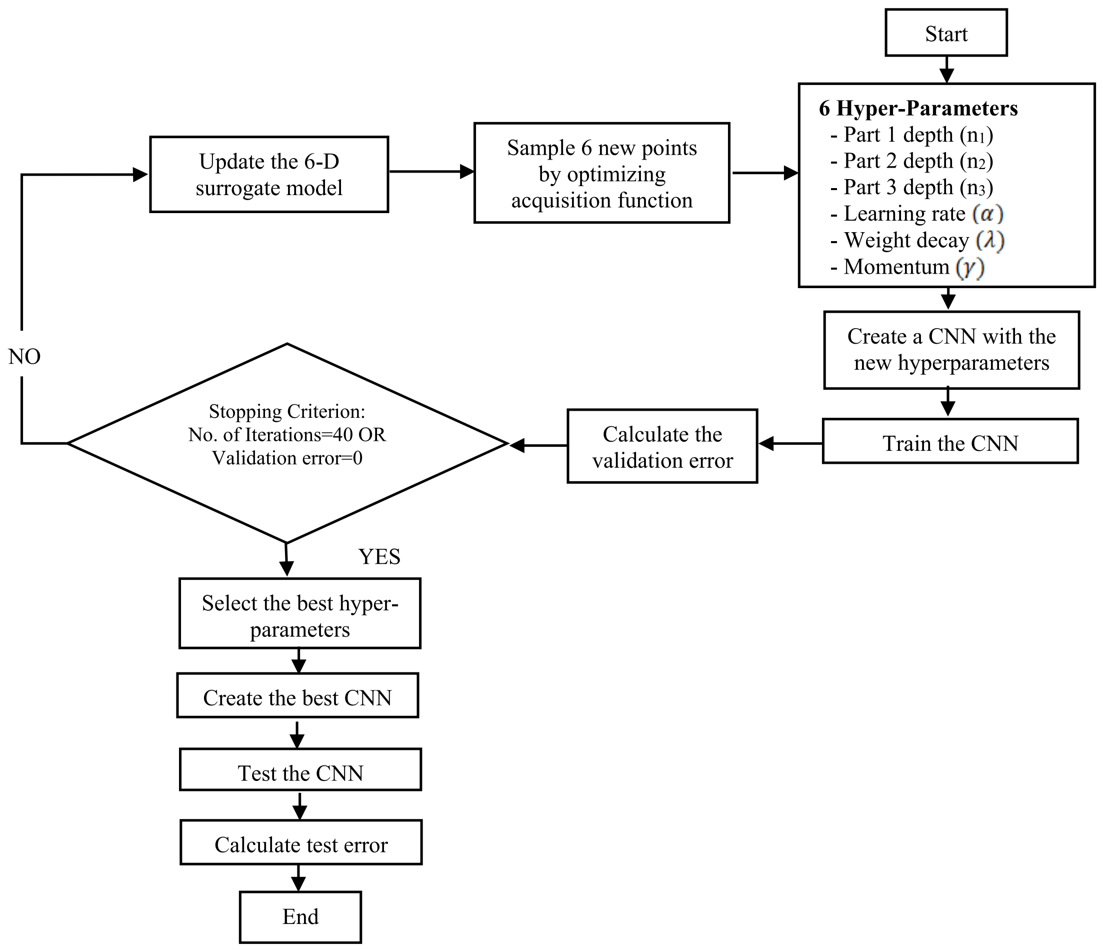

Start
6 Hyper-Parameters
- Part 1 depth (n_1)
- Part 2 depth (n_2)
- Part 3 depth (n_3)
- Learning rate (α)
- Weight decay (λ)
- Momentum (γ)
Create a CNN with the new hyperparameters
Train the CNN
Calculate the validation error
Stopping Criterion: No. of Iterations=40 OR Validation error=0
NO
Update the 6-D surrogate model
Sample 6 new points by optimizing acquisition function
YES
Select the best hyper-parameters
Create the best CNN
Test the CNN
Calculate test error
End

FIGURE 5. Flow chart of the CNN hyperparameter optimization process.

TABLE 1. Experimental settings.

	Parameter	value
Optimized parameters	Part 1 depth (n_1)	$n_1 \in \{1,2,3\}$
	Part 2 depth (n_2)	$n_2 \in \{1,2,3\}$
	Part 3 depth (n_3)	$n_3 \in \{1,2,3\}$
	Learning rate (α)	$\alpha \in [10^{-6}\ \ 1]$ on the logarithmic scale
	Momentum (γ)	$\gamma \ \in [0\ 1]$
	Weight decay (λ)	$\lambda \in [10^{-10} 10^{-1}]$ on the logarithmic scale
Dependent parameter	Number of convolutional layer kernels	$No.\ of\ \ conv\ kernels = floor\left(\frac{10}{\sqrt{part\ depth}} \times 2^{part\ number-1}\right)$
Fixed parameters	Number of training samples	800
	Number of validation samples	200
	Number of test samples	200
	Mini-batch size	100
	Max number of epochs	40
	Training optimization algorithm	Stochastic gradient descent (SGDM)
	Loss function in the softmax layer	Cross entropy
	Number of objective function evaluations for Bayesian optimizer	40
	Acquisition function for Bayesian optimizer	Expected improvement (EI)

equation:

$$Ident.Acc. = \left(1 - \frac{No.\ of\ \ misclassified\ test\ samples}{Total\ no.\ of\ \ test\ samples}\right) \tag{29}$$

The proposed model with the Jerman filter correctly classified 199 out of the 200 images of the test set, implying that the features learned by our CNN model with Bayesian optimization are highly relevant and representative. The first experiment with Jerman filter preprocessing yielded a lower

TABLE 2. The optimum CNN hyperparameters evaluated using Bayesian optimization and different ranges of part depth.

Exp. No	n1 range	n2 range	n3 range	$\{n_1, n_2, n_3\}$	α	γ	λ	Training Acc. (%)	validation error	Test error
1	[1 3]	[1 3]	[1 3]	{1,1,1,}	0.061456	0.4191	0.075994	100	0	0.005
2	[3 6]	[3 6]	[4 6]	{4,3,5}	0.16772	0.25609	0.012875	100	0.035	0.07
3	[6 9]	[6 9]	[7 9]	{7,6,8}	0.14361	0.0071142	0.00016076	100	0.12	0.19
4	[9 12]	[9 12]	[10 12]	{11,9,12}	0.033752	0. 74366	0. 70697	100	0.2050	0.3050
5	[12 15]	[12 15]	[13 15]	{12,13,13}	0.041247	0.006821	0.83516	100	0.250	0.3350

TABLE 3. The optimized hyperparameters of the two fine-tuned cnns obtained from both experiments.

Preprocessing	Part1 depth (n_1)	Part2 depth (n_2)	Part3 depth (n_3)	Learning rate (α)	Momentum (γ)	Weight decay (λ)	Validation error	Test error	EER (%)
Using Jeramn filter	1	1	1	0.061456	0.4191	0.075994	0	0.005	0.0353
Without Jeramn filter	2	1	1	0.065001	0.64222	0.0062575	0.025	0.035	0.9697

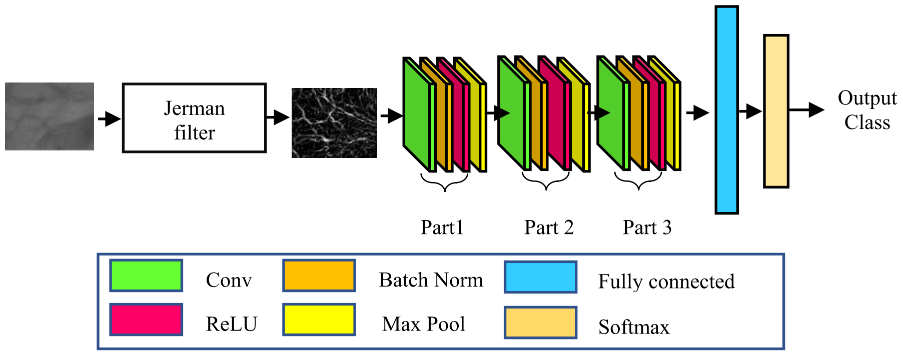


FIGURE 6. The optimized structure of the fine-tuned convolutional neural network of the proposed method with Jerman filter.

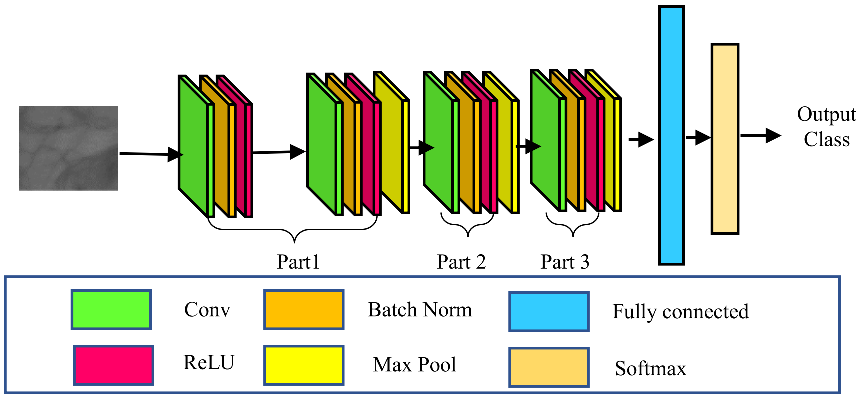


FIGURE 7. The optimized structure of the fine-tuned convolutional neural network of the proposed method without Jerman filter.

test error and hence higher identification accuracy than that of the second experiment without the preprocessing step, as shown in Table 3. The identification accuracies of the first and second experiments are 99.5 % and 96.5 % respectively, which reflects the importance of the Jerman filter in preprocessing.

B. PALM VEIN VERIFICATION

This is the process of asking the biometric system whether a palm vein image belongs to a specific person or not. It is called a one-to-one matching problem. In our experiment, the softmax layer real output values were used for distance calculation and computing both genuine and impostor matching scores.

For genuine matching, the number of matching scores is (the number of test palm vein images per palm-1) $\times$ (the number of classes). As a result, there will be 100 (1×100) genuine matching scores. For impostor matching, the number of matching scores equals (the number of test palm vein images per palm) $\times$ (total number of test images-number of

TABLE 4. The optimized hyperparameters and validation error values at every objective function evaluation.

Iter.#	Validation error	Part1 depth (n_1)	Part2 depth (n_2)	Part3 depth (n_3)	Learning Rate(α)	Momentum(γ)	Weight decay(λ)
1	0.14	3	2	3	0.51793	0.65036	0.0080666
2	1	2	2	2	5.9922e-06	0.91809	3.2361e-08
3	0.115	2	2	2	0.37706	0.28615	1.4135e-10
4	1	2	2	2	3.833e-06	0.5979	0.059719
5	0.01	2	1	2	0.12119	0.8301	1.3376e-08
6	0.23	2	2	3	0.017277	0.48846	0.0046997
7	0.01	2	1	3	0.19242	0.58195	0.00076803
8	0.035	2	1	3	0.14906	0.54882	4.5238e-09
9	0.03	1	1	3	0.292	0.24524	1.906e-06
10	0.015	2	1	1	0.19944	0.19342	0.00083422
11	0.065	3	1	3	0.14461	0.44267	2.6094e-10
12	0.055	1	3	1	0.10611	0.65914	0.076688
13	0.005	1	1	1	0.12203	0.41636	0.084532
14	0.6	1	1	2	0.16127	0.99702	0.014244
15	0.015	1	1	3	0.071033	0.75506	6.6685e-06
16	0.08	2	2	2	0.019497	0.99148	1.0217e-10
17	0.98	2	1	1	0.75676	0.84749	0.046733
18	0.025	2	1	1	0.050074	0.6412	9.4235e-10
19	0.465	2	2	2	0.87774	0.026903	0.038736
20	0.415	1	1	2	0.011725	0.064441	2.0641e-06
21	0.085	1	1	1	0.64326	0.43269	0.011076
22	0.08	2	1	1	0.0023302	0.93097	0.0002301
23	0.14	2	1	1	0.0062167	0.76122	2.0208e-10
24	0.955	1	1	3	0.00046622	0.82746	0.0002884
25	0.06	1	2	1	0.0061711	0.99664	0.00045769
26	0.01	1	2	1	0.095457	0.76193	2.8354e-10
27	0.025	1	1	2	0.03003	0.83517	7.4969e-08
28	0.03	1	1	1	0.075863	0.25271	0.00062241
29	0.015	2	1	2	0.17686	0.28045	0.0017725
30	0.015	3	2	2	0.044721	0.83099	0.043905
31	0.015	1	1	1	0.10779	0.022897	9.5918e-05
32	0.005	3	1	3	0.091579	0.77462	0.03321
33	0.025	1	1	2	0.36412	0.52588	9.4961e-08
34	0	1	1	1	0.061456	0.4191	0.075994

test images per palm) $\times$ total number of palms. As a result, there will be 39,600 ($2 \times 198 \times 100$) impostor matching scores.

Cosine similarity was utilized in this experiment for distance matching score calculation. The cosine similarity score

TABLE 5. The detailed structure of the fine-tuned CNN network model for the first experiment with Jerman filter.

	Layer	Filter Size	No. of Filters	Stride Value	Padding Size	Feature Map Size
	Input image	-	-	-	-	40×40×1
Part 1	Conv_1	4×4	10	1×1	[1,2,1,2]	40×40×10
Part 1	Batch Norm_1.	-	-	-	-	40×40×10
Part 1	ReLU_1	-	-	-	-	40×40×10
	MaxPool_1	2×2	-	2×2	[0,0,0,0]	20×20×10
Part 2	Conv_2	4×4	20	1×1	[1,2,1,2]	20×20×20
Part 2	Batch Norm_2.	-	-	-	-	20×20×20
Part 2	ReLU_2	-	-	-	-	20×20×20
	MaxPool_2	2×2	-	2×2	[0,0,0,0]	10×10×20
Part 3	Conv_3	3×3	40	1×1	[1,1,1,1]	10×10×40
Part 3	Batch Norm_3	-	-	-	-	10×10×40
Part 3	ReLU_3	-	-	-	-	10×10×40
	MaxPool_3	2×2	-	2×2	[0,0,0,0]	5×5×40
	Fully Connected Layer	-	-	-	-	1×1×100
	Softmax Layer	-	-	-	-	1×1×100
	Classification Output Layer	-	-	-	-	1×1×100

TABLE 6. Ablation experiment on the optimum hyperparameters of fine-tuned cnn model with Jerman filter.

Part1 convolutional layer	Part2 convolutional layer	Part3 convolutional layer	Momentum (γ)	Weight decay (λ)	identification Accuracy %
✓	-	-	-	-	95
✓	✓	-	-	-	97.5
✓	✓	✓	-	-	98.5
✓	✓	✓	✓	-	98.5
✓	✓	✓	✓	✓	99.5

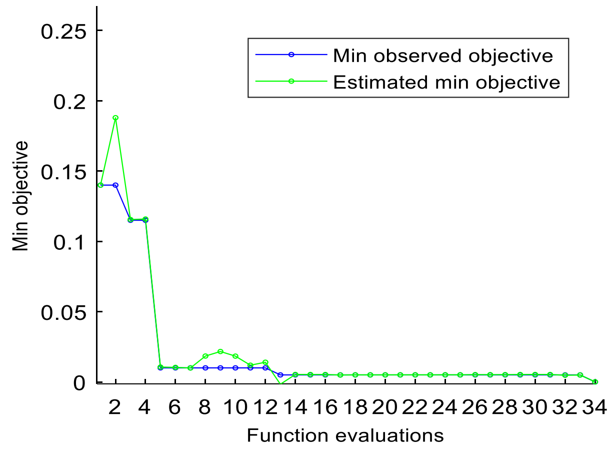


FIGURE 8. Min objective vs. number of function evaluations for the experiment with Jerman filter.

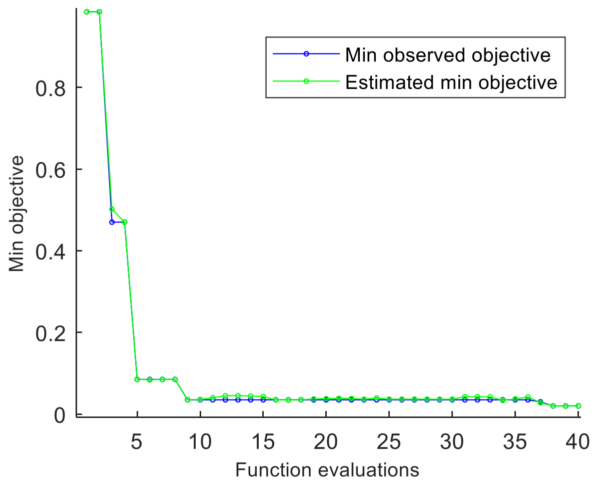


FIGURE 9. Min objective vs. number of function evaluations for the experiment without Jerman filter.

between two feature vectors x and y with an angle θ between them is:

$$similarity\,(x,y) = \cos(\theta) = \frac{x^T.y}{\|x\|\,.\,\|y\|} \tag{30}$$

The EER value can also be determined as the FAR value at the intersection point between the ROC curve and the EER line, as shown in Fig. 10 and Fig. 11. The first part of this

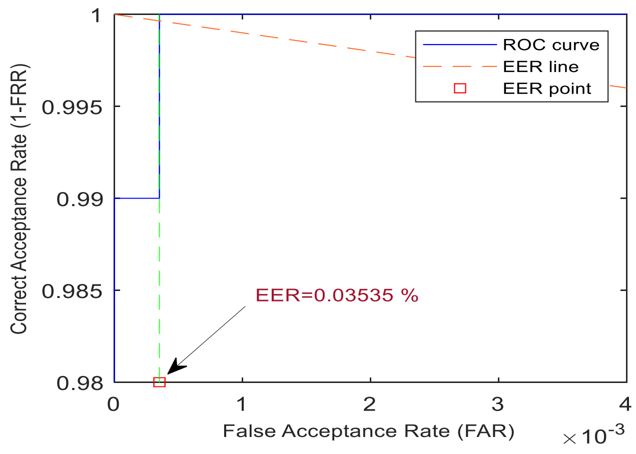


FIGURE 10. ROC curve for the experiment with Jerman filter.

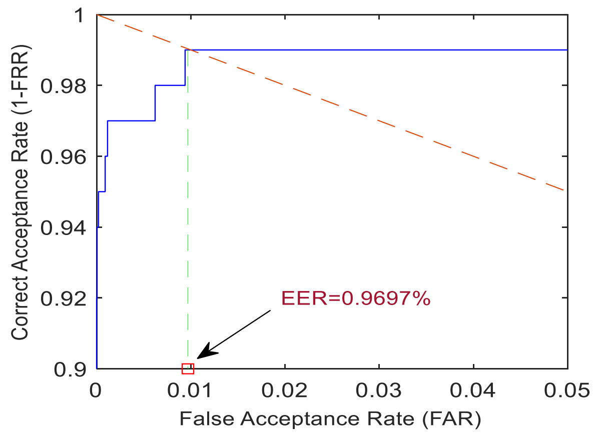


FIGURE 11. ROC curve for the experiment without Jerman filter.

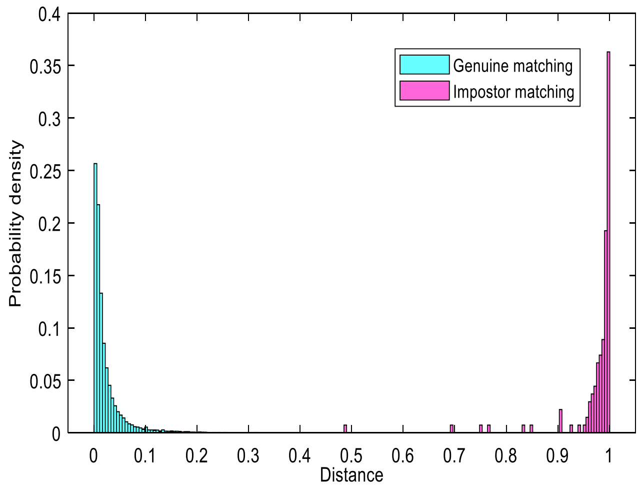


FIGURE 12. Histograms of genuine and impostor match distance using Jerman filter.

experiment, with the Jerman filter, achieved a lower EER than that without the Jerman filter, as shown in Table 3. Fig. 12 and Fig. 13 show the distribution of both genuine and impostor matching scores for the first and second experiments,

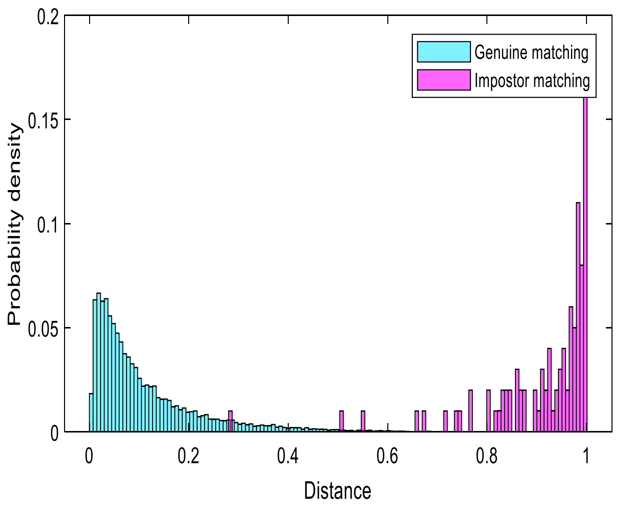


FIGURE 13. Histograms of genuine and impostor match distance without using Jerman filter.

TABLE 7. EER for each training/test fold.

Fold #	No. of Training samples	No. of Test samples	Identification Acc. (%)	EER (%)
1	800	200	99.5	0.0353
2	800	200	99.5	0.065
3	800	200	99	0.126
4	800	200	100	0.025
5	800	200	99	0.09
Average			**99.4**	**0.0683**

respectively. As shown in Fig. 12, the relative separation between both scores reflects the independence between them and the robustness of the proposed model. However, there is still a small intersection area in the distribution due to to the deformation or blurring of the palm vein ROIs. Table 6 shows the ablation experiment on the fine-tuned CNN model with a Jerman filter for the palm vein identification task. In the first stage, we trained the CNN with only the first part of the network and the following max pool layer, as well as the softmax layer, the fully connected layer, and the classification output layer, which is located at the end of the network. The identification accuracy dropped to 95%. In the next experiment, we added the second part of the network, to which we connected the max pool layer. This resulted in a significant improvement in the identification accuracy, by 2.5 %. However, adding the last part and the following max pool layer achieved a relatively small improvement, of 1%. Adding the momentum had no impact on the identification accuracy, while adding the weight decay improved the identification accuracy by 1%. To obtain some robust and unbiased estimates of the final CNN model's performance on new test samples, K-fold cross validation [61] was adopted. This technique is commonly employed for either model selection or evaluation in machine learning problems. The 1000 images assigned for training and testing were divided into k (k=5)

TABLE 8. Accuracy rate for palm vein identification.

Method	Year	Algorithm	Identification ACC (%)
Meraoumia et al. [28]	2017	PCANet with Deep Learning	96.5
Hassan et al [62]	2018	CNN	98
Thabar et al [63]	2018	CNN with autoencoder	85.16
Lisha et al [64]	2019	CNN with random forest	97.5
Proposed method	2020	CNN + Bayesian optimization +Jerman filter	99.4

TABLE 9. Eers for palm vein verification.

Method	Year	Algorithm	EER (%)
Meraoumia et.al [28]	2017	PCANet with Deep Learning	0.949
Thabar et al [63]	2018	CNN with autoencoder	3.71
Wang et al [65]	2019	U-Net	0.47
Qin et al [66]	2019	Deep belief neural networks (DBNN)	0.33
Proposed method	2020	CNN + Bayesian optimization +Jerman filter	0.0683

TABLE 10. Structure and processing time of different methods.

Method	No. of convolutional layers	No. of convolutional kernels	Time (in seconds) for one palm identification
Hassan et al [62]	4	480	0.01
Thabar et al [63]	14	736	-
Lisha et al [64]	5	-	-
Wang et al [65]	18	4864	-
Proposed method	3	70	0.0076

partitions of equal size. The final model was trained on four partitions and tested on the remaining partition. This process was repeated for each unique group so that each sample was used once in the test set and four times in the training set. Finally, the identification accuracy and EERs were averaged. The identification accuracy and EER for each fold are shown in Table 7.

VIII. DISCUSSION

The experimental results shown in Table 8 and Table 9 demonstrate the high performance of our proposed approach.

Strikingly, it can be seen that the final CNN model obtained from Bayesian optimization can achieve high identification accuracy and low EER for all the five folds shown in Table 7.

Even when the EER value is relatively high for the third fold, it is still lower than those of related state-of-the-art methods. This is a strong indication that the final CNN model generalizes well to new data. Moreover, it can be seen that it attained an average EER of 0.0683 and identification accuracy of 99.4%, which makes it superior in comparison with state-of-the-art methods.

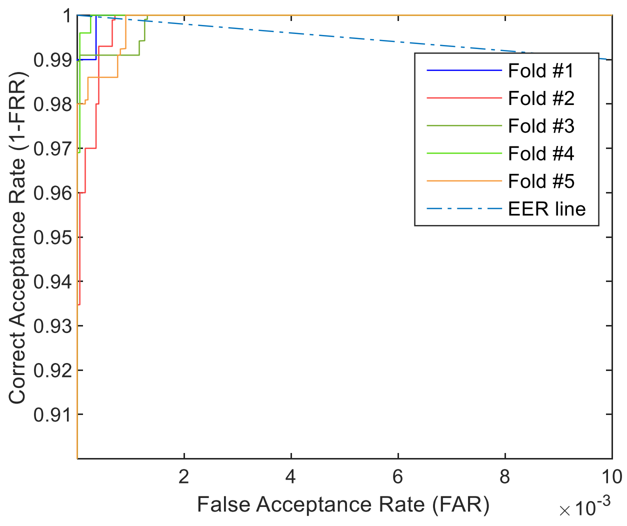


FIGURE 14. ROC curves for all training/test folds.

Such good performance may be explained as follows: 1) the complex features (both low-level and high-level features) learned by the proposed CNN model are highly representative and relevant enough to successfully discriminate between 100 different classes. Since nearly all methods for segmentation of the palm rely on the valley points between fingers, the pixel locations of these valley points vary slightly from one scan to another for the same identity. This is attributed to the difference between hand poses with respect to the scanner, for each scan. Due to these variations, the segmented ROI's for the same person will differ in the amount of translation. However, the proposed approach has proven to be translation invariant. 2) the use of the Jerman filter to enhance the vein pattern contributes to improving the model's performance, as demonstrated in both experiments (with and without Jerman filter), as shown in Table 3, Fig.10, and Fig.11. 3) Since the performance of a CNN is sensitive to the selection of hyperparameters and these hyperparameters cannot be manually set according to human expertise, the use of Bayesian optimization to automate the selection of the model structure as well as training options also contributes in finding the best model, hence improving the model's performance.

It is worth mentioning that existing approaches such as those in [28], [62]–[66] did not use an optimization algorithm to automate the selection of the hyperparameters. As a result, an excessive amount of computation time was invested in randomly training multiple models in the hope of finding a fit model for palm vein authentication. Bayesian optimization avoids unnecessarily training models and, hence, finds the best model in a few iterations. Thus, our proposed approach is more computationally efficient when compared to the previously mentioned works. Moreover, the existing deep learning based-approaches, including those mentioned above, used fixed or pretrained deep learning structures, without taking into consideration the impact of unnecessary convolutional layers on the model's performance. In the present work, the impact of adding unnecessary layers causes the network to overfit to the training data, as shown in Table 2. To solve this problem, optimization of the convolutional neural network structure (i.e., the number of convolutional layers) is performed to avoid adding unnecessary convolutional layers to the network structure. As shown in Table 10, these models not only take into consideration the impact of unnecessary convolutional layers on the overfitting problem but also the amount of time the model takes to identify one palm vein image, since adding multiple convolutional layers as well as too many convolutional kernels comes at the expense of processing time. The proposed model is computationally fast compared to the other methods shown in Table 10, since the time needed to identify one palm image is only about 0.0076 seconds. This is attributed to incorporating Bayesian optimization into the structure of the CNN, hence, building a model with the smallest, yet optimum number of convolutional layers. All experiments were conducted using MATLAB R2019a on a laptop with Windows 10 (x64), Intel(R) core (TM) i7-4510 U CPU (2.0 GHz), and 6GB RAM.

In order to conduct a completely fair and representative comparison with the existing approaches, the comparison must include studies with similar approaches and the same database. We considered that issue when comparing our proposed approach with the existing ones shown in Table 8 and Table 9. However, the training, validation and test divisions are not the same, even between the existing approaches themselves. Otherwise, any other comparison in this paper was conducted under the same testing conditions. Moreover, the proposed model is still sensitive to a large amount of rotation but this problem was solved with the robust segmentation method we employed. A model that is robust to large rotations would be more efficient for palm vein authentication. We plan to direct our future research in this direction encompassing a larger database with a larger number of identities. We also plan to implement an accurate and robust segmentation method to accurately extract the vascular pattern of the palm and use this pattern as input to a CNN to measure the performance of the model in this case compared to the existing works.

IX. CONCLUSION

In this paper, we have investigated the problem of palm vein authentication with trained models and proposed a fine-tuned deep CNN-based paradigm for palm vein authentication using Bayesian optimization. At the image preprocessing stage, the ROI was extracted from the palm vein image and then passed through a Jerman vessel enhancement filter to highlight the vein pattern in the ROI. To automate the process of manual selection of model hyperparameters, it was important to optimize both the structure and training options. Therefore, we proposed dividing the structure of our network model into three main parts. Each part comprised a number of blocks. We defined a 6-D objective function to take the number of blocks in each part as input to optimize the structure as well as the training options. The objective function returns the validation error to minimize it. A stopping criterion was set based on the validation error and maximum number of iterations. The optimization algorithm successfully found the optimal network structure and training options in a few numbers of iterations. The final CNN model achieved an average identification accuracy of 99.4% and average EER of 0.0683%, which outperformed other state-of-the-art palm vein authentication techniques.

ACKNOWLEDGMENT

This research was funded by Deanship of Scientific Research at Princess Nourah bint Abdulrahman University. (Grant No.41-PRFA-P-25). The authors declare no conflict of interest.

REFERENCES

[1] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. London, U.K.: Springer, 2009.

[2] H. Wechsler, *Reliable Face Recognition Methods: System Design, Implementation and Evaluation*. New York, NY, USA: Springer, 2010.

[3] M. Arsalan, H. Hong, R. Naqvi, M. Lee, M. Kim, D. Kim, C. Kim, and K. Park, "Deep learning-based iris segmentation for iris recognition in visible light environment," *Symmetry*, vol. 9, no. 11, p. 263, Nov. 2017, doi: 10.3390/sym9110263.

[4] Z. Yaxin, L. Huanhuan, G. Xuefei, and L. Lili, "Palmprint recognition based on multi-feature integration," in *Proc. IEEE Adv. Inf. Manage., Communicates, Electron. Autom. Control Conf. (IMCEC)*, Xi'an, China, Oct. 2016, pp. 992–995.

[5] W.-Y. Han and J.-C. Lee, "Palm vein recognition using adaptive Gabor filter," *Expert Syst. Appl.*, vol. 39, no. 18, pp. 13225–13234, Dec. 2012, doi: 10.1016/j.eswa.2012.05.079.

[6] R. Wang, G. Wang, Z. Chen, Z. Zeng, and Y. Wang, "A palm vein identification system based on Gabor wavelet features," *Neural Comput. Appl.*, vol. 24, no. 1, pp. 161–168, Jan. 2014, doi: 10.1007/s00521-013-1514-8.

[7] W. Kang and D. Feiqi, "Vein image enhancement and segmentation based on maximal intra-neighbor difference," *Acta Optica Sinica*, vol. 29, no. 7, pp. 1830–1837, 2009.

[8] L. Mirmohamadsadeghi and A. Drygajlo, "Palm vein recognition with local binary patterns and local derivative patterns," in *Proc. Int. Joint Conf. Biometrics (IJCB)*, Washington, DC, USA, Oct. 2011, pp. 1–6, doi: 10.1109/IJCB.2011.6117804.

[9] W. Lu, M. Li, and L. Zhang, "Palm vein recognition using directional features derived from local binary patterns," *Int. J. Signal Process., Image Process. Pattern Recognit.*, vol. 9, no. 5, pp. 87–98, May 2016, doi: 10.14257/ijsip.2016.9.5.09.

[10] A. Aglio-Caballero, B. Rios-Sanchez, C. Sanchez-Avila, and M. J. M. D. Giles, "Analysis of local binary patterns and uniform local binary patterns for palm vein biometric recognition," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Madrid, Spain, Oct. 2017, pp. 1–6, doi: 10.1109/CCST.2017.8167808.

[11] D. Fronitasari and D. Gunawan, "Palm vein recognition by using modified of local binary pattern (LBP) for extraction feature," in *Proc. 15th Int. Conf. Qual. Res. (QiR), Int. Symp. Electr. Comput. Eng.*, Nusa Dua, Indonesia, Jul. 2017, pp. 18–22, doi: 10.1109/QIR.2017.8168444.

[12] L. Mirmohamadsadeghi and A. Drygajlo, "Palm vein recognition with local texture patterns," *IET Biometrics*, vol. 3, no. 4, pp. 198–206, Dec. 2014, doi: 10.1049/iet-bmt.2013.0041.

[13] Manmohan, J. Saxena, K. Teckchandani, P. Pandey, M. K. Dutta, C. M. Travieso, and J. B. Alonso-Hernandez, "Palm vein recognition using local tetra patterns," in *Proc. 4th Int. Work Conf. Bioinspired Intell. (IWOBI)*, Jun. 2015, pp. 151–156.

[14] R. C. Rahul, M. Cherian, and M. Mohan C M, "A novel MF-LDTP approach for contactless palm vein recognition," in *Proc. Int. Conf. Comput. Netw. Commun. (CoCoNet)*, Trivandrum, India, Dec. 2015, pp. 793–798, doi: 10.1109/CoCoNet.2015.7411280.

[15] A. F. Akbar, T. A. B. Wirayudha, and M. D. Sulistiyo, "Palm vein biometric identification system using local derivative pattern," in *Proc. 4th Int. Conf. Inf. Commun. Technol. (ICoICT)*, Bandung, Indonesia, May 2016, pp. 1–6, doi: 10.1109/ICoICT.2016.7571956.

[16] S. Elnasir and S. M. Shamsuddin, "Palm vein recognition based on 2D-discrete wavelet transform and linear discrimination analysis," *Int. J. Adv. Soft Comput. Appl.*, vol. 6, no. 3, pp. 43–59, 2014.

[17] F. Rizki, T. A. B. Wirayuda, and K. N. Ramadhani, "Identity recognition based on palm vein feature using two-dimensional linear discriminant analysis," in *Proc. 1st Int. Conf. Inf. Technol., Inf. Syst. Electr. Eng. (ICITISEE)*, Yogyakarta, Indonesia, Aug. 2016, pp. 21–25, doi: 10.1109/ICITISEE.2016.7803041.

[18] D. Y. Perwira, B. W. T. Agung, and M. D. Sulistiyo, "Personal palm vein identification using principal component analysis and probabilistic neural network," in *Proc. Int. Conf. Inf. Technol. Syst. Innov. (ICITSI)*, Nov. 2014, pp. 99–104, doi: 10.1109/ICITSI.2014.7048245.

[19] S. Bayoumi, S. Al-Zahrani, A. Sheikh, G. Al-Sebayel, S. Al-Magooshi, and S. Al-Sayigh, "PCA-based palm vein authentication system," in *Proc. Int. Conf. Inf. Sci. Appl. (ICISA)*, Suwon, South Korea, Jun. 2013, pp. 1–3, doi: 10.1109/ICISA.2013.6579422.

[20] M. Xin and J. Xiaojun, "Palm vein recognition method based on fusion of local Gabor histograms," *J. China Universities Posts Commun.*, vol. 24, no. 6, pp. 55–66, Dec. 2017.

[21] Y. Zhou, Y. Liu, Q. Feng, F. Yang, J. Huang, and Y. Nie, "Palm-vein classification based on principal orientation features," *PLoS ONE*, vol. 9, no. 11, pp. 1–12, 2014, doi: 10.1371/journal.pone.0112429.

[22] M. Pan and W. Kang, "Palm vein recognition based on three local invariant feature extraction algorithms," in *Proc. 6th Chin. Conf. Biometric Recognit.*, 2011, pp. 116–124.

[23] V. Gurunathan, T. Sathiyapriya, and R. Sudhakar, "Multimodal biometric recognition system using SURF algorithm," in *Proc. 10th Int. Conf. Intell. Syst. Control (ISCO)*, Jan. 2016, pp. 1–5, doi: 10.1109/ISCO.2016.7727020.

[24] P. O. Ladoux, C. Rosenberger, and B. Dorizzi, "Palm vein verification system based on SIFT matching," in *Proc. Int. Conf. Biometrics*. Berlin, Germany: Springer, 2009, pp. 1290–1298.

[25] W. Kang, Y. Liu, Q. Wu, and X. Yue, "Contact-free palm-vein recognition based on local invariant features," *PLoS ONE*, vol. 9, no. 5, May 2014, Art. no. e97548, doi: 10.1371/journal.pone.0097548.

[26] M. E. C. Villarina and N. B. Linsangan, "Palm vein recognition system using directional coding and back-propagation neural network," in *Proc. World Congr. Eng. Comput. Sci.*, San Francisko, CA, USA, vol. 2, 2015, pp. 21–23

[27] L. Zhang, Z. Cheng, Y. Shen, and D. Wang, "Palmprint and palm vein recognition based on DCNN and a new large-scale contactless palm vein dataset," *Symmetry*, vol. 10, no. 4, pp. 1–15, 2018, doi: 10.3390/sym10040078.

[28] A. Meraoumia, F. Kadri, H. Bendjenna, S. Chitroub, and A. Bouridane, "Improving biometric identification performance using PCANet deep learning and multispectral palmprint," in *Signal Processing for Security Technologies*. Cham, Switzerland: Springer, 2017, pp. 51–69.

[29] D. Zhong, S. Liu, W. Wang, and X. Du, "Palm vein recognition with deep hashing network," in *Proc. 1st Chin. Conf. Pattern Recognit. Comput. Vis. (PRCV)*, Nov. 2018, pp. 38–49.

[30] S. Chantaf, A. Hilal, and R. ElSaleh, "Palm vein biometric authentication using convolutional neural networks," in *Proc. 8th Int. Conf. Sci. Electron., Technol. Inf. Telecommun. (SETIT)*, vol. 1, 2019, pp. 352–363.

[31] S. Rahman, L. Wang, C. Sun, and L. Zhou, "Deep learning based HEp-2 image classification: A comprehensive review," 2019, *arXiv:1911.08916*. [Online]. Available: http://arxiv.org/abs/1911.08916

[32] P. Doke, D. Shrivastava, C. Pan, Q. Zhou, and Y.-D. Zhang, "Using CNN with Bayesian optimization to identify cerebral micro-bleeds," *Mach. Vis. Appl.*, vol. 31, no. 5, p. 36, Jul. 2020, doi: 10.1007/s00138-020-01087-0.

[33] *CAIS Multispectral Palmprint*. Accessed: Feb. 12, 2018. [Online]. Available: http:// biometrics.idealtest.org/

[34] T. Jerman, F. Pernus, B. Likar, and Z. Spiclin, "Enhancement of vascular structures in_newline 3D and 2D angiographic images," *IEEE Trans. Med. Imag.*, vol. 35, no. 9, pp. 2107–2118, Sep. 2016.

[35] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever, "Multiscale vessel enhancement filtering," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI'98*, vol. 1496, W. M. Wells, A. Colchester, and S. Delp, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 130–137.

[36] Y. Sato, C. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis, "Tissue classification based on 3D local intensity structures for volume rendering," *IEEE Trans. Vis. Comput. Graphics*, vol. 6, no. 2, pp. 160–180, Apr./Jun. 2000.

[37] Q. Li, S. Sone, and K. Doi, "Selective enhancement filters for nodules, vessels, and airway walls in two- and three-dimensional CT scans," *Med. Phys.*, vol. 30, no. 8, pp. 2040–2051, Jul. 2003.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptive Computation and Machine Learning series). Cambridge, MA, USA: MIT Press,2016.

[39] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015, doi: 10.1016/j.neunet.2014.09.003.

[40] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[42] H. Aghdam and E. J. Jahani, *Guide to Convolutional Neural Networks*. New York, NY, USA: Springer, 2017.

[43] V. Perrone, H. Shen, M. Seeger, C. Archambeau, and R. Jenatton, "Learning search spaces for Bayesian optimization: Another view of hyperparameter transfer learning," in *Proc. NeurIPS*, 2019, pp. 12771–12781. [Online]. Available: http://arxiv.org/abs/1909.12552

[44] P. I. Frazier, "A tutorial on Bayesian optimization," 2018, *arXiv:1807.02811*. [Online]. Available: http://arxiv.org/abs/1807.02811

[45] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. D. Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2016, doi: 10.1109/JPROC.2015.2494218.

[46] E. Brochu, V. M. Cora, and N. D. Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," 2010, *arXiv:1012.2599*. [Online]. Available: http://arxiv.org/abs/1012.2599

[47] D. D. Cox and S. John, "A statistical method for global optimization," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 1992, pp. 1241–1246, doi: 10.1109/ICSMC.1992.271617.

[48] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *J. Basic Eng.*, vol. 86, no. 1, pp. 97–106, Mar. 1964.

[49] J. Mockus, "On Bayesian methods for seeking the extremum," in *Proc. Optim. Techn. IFIP Tech. Conf.*, 1977, pp. 195–200.

[50] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights Imag.*, vol. 9, no. 4, pp. 611–629, Aug. 2018, doi: 10.1007/s13244-018-0639-9.

[51] S. Raschka, V. Mirjalili, *Python Machine Learning: Machine Learning and Deep Learning With Python*, 3rd ed. Birmingham, U.K.: Packt Publishing Ltd, 2019, p. 38.

[52] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 437–478.

[53] K. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012, p. 252.

[54] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, Oct. 2016, doi: 10.1109/TGRS.2016.2584107.

[55] S. Prasad, J. Chanussot, *Hyperspectral Image Analysis: Advances in Machine Learning and Signal Processing*. Cham, Switzerland: Springer Nature, Apr. 2020, p. 107.

[56] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. New York, NY, USA: Springer, 2013, p. 144.

[57] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[58] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

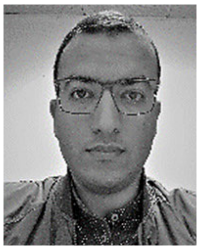
[59] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 1800–1807, doi: 10.1109/CVPR.2017.195.

[60] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, vol. 3, Feb. 2013, pp. 2176–2184.

[61] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, vol. 112. New York, NY, USA: Springer, 2013, doi: 10.1007/978-1-4614-7138-7.

[62] N. F. Hassan and H. I. Abdulrazzaq, ''Pose invariant palm vein identification system using convolutional neural network,'' *Baghdad Sci. J.*, vol. 15, no. 4, pp. 502–509, 2018, doi: 10.21123/bsj.2018.15.4.0502.
[63] D. Thapar, G. Jaswal, A. Nigam, and V. Kanhangad, ''PVSNet: Palm vein authentication siamese network trained using triplet loss and adaptive hard mining by learning enforced domain specific features,'' in *Proc. IEEE 5th Int. Conf. Identity, Secur., Behav. Anal. (ISBA)*, Jan. 2019, pp. 1–8, doi: 10.1109/ISBA.2019.8778623.
[64] Y. Lisha, L. Mengying, L. Yaqin, Y. Feng, and H. Jing, ''Palm vein classification based on deep neural network and random forest,'' *Laser Optoelectron. Prog.*, vol. 56, no. 10, 2019, Art. no. 101010.
[65] P. Wang and H. Qin, ''Palm-vein verification based on U-Net,'' in *Proc. IOP Conf. Ser., Mater. Sci. Eng.*, vol. 806, May 2020, Art. no. 012043, doi: 10.1088/1757-899X/806/1/012043.
[66] H. Qin, M. A. El Yacoubi, J. Lin, and B. Liu, ''An iterative deep neural network for hand-vein verification,'' *IEEE Access*, vol. 7, pp. 34823–34837, 2019, doi: 10.1109/ACCESS.2019.2901335.

MARWA ISMAEL OBAYYA received the B.Sc. degree in electronics and communications engineering from the Faculty of Engineering, Mansoura University, Mansoura, Egypt, in 2001, and the M.Sc. and Ph.D. degrees from the Department of Electronics and Communications Engineering, Faculty of Engineering, Mansoura University, in 2005 and 2008, respectively. She is currently an Associate Professor with the Electronics and Communications Engineering Department, Faculty of Engineering, Mansoura University. She is also an Associate Professor with the Communications Engineering Program, Electrical Engineering Department, Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia. She has several publications in the biomedical engineering, optimization, and intelligent machine learning. Her research interests include in the field of image processing, signal processing, optimization, and machine learning.



MOHAMMED EL-GHANDOUR received the B.Sc. and M.Sc. degrees from the Department of Electronics and Communications Engineering, Faculty of Engineering, Mansoura University, in 2010 and 2016, respectively. He is currently a Researcher with the Faculty of Engineering, Mansoura University. He is also a Communication Engineer with National Air Navigation Services Company (NANSC), Cairo, Egypt. He has some publications in the image processing area. His research interests include in the area of processing, analysis, and classification of biomedical signal and images.

FADWA ALROWAIS received the B.Sc. degree (Hons.) in computer and information sciences in the field of computer applications and the M.Sc. degree in computer science from the College of Computer and Information Sciences, King Saud University, Saudi Arabia, in 1996 and 2005, respectively, and the Ph.D. degree in computer science from the Faculty of Electronics and Computer Sciences, Southampton University, U.K., in 2016. She has 23 years academic experience, as she has worked with the College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, and with the College of Engineering.