International Journal on Emerging Technologies 11(4): 282-291(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

Mutual Component Convolutional Siamese Network for Heterogeneous Iris Recognition

Jwad Ali Ridha[1] and Jameelah Harbi Saud[2]
[1]Informatics Institute for Postgraduate Studies,
Iraqi Commission for Computer and Informatics, Baghdad, Iraq.
[2]College of Science, Mustansiriyah University, Baghdad, Iraq.

(Corresponding author: Jwad Ali Ridha)
(Received 15 May 2020, Revised 22 June 2020, Accepted 06 July 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Iris recognition system is a special system employed to identify individuals according to their iris. The iris patterns are extremely randomized and unique. However, iris images captured by different sensors have heterogeneous characteristics. Matching heterogeneous iris patterns can degrade the rate of recognition and the system will not be as accurate as when all iris images are captured by the same sensor. Therefore it is a challenging issue to design a model that finds the matching between heterogeneous iris patterns. In this paper, a deep learning based framework, to address the issue of heterogeneous iris recognition, is presented. The framework termed as Mutual Component Convolutional Siamese Network (MCCSN), incorporates a learnable distance measure module into Siamese Neural Network (SNN) architecture, when the identical twin branches of this network are formed from a pre-trained Convolutional Neural Network (CNN), and sharing all parameters. Based on deep features extracted from input heterogeneous pairs simultaneously by the base CNN branches, the learnable distance measure module is designed to calculate the distance between heterogeneous iris patterns, and its parameters are learned according to the binary label attached with each pair during training process. The binary labels determine the similarity and dissimilarity between the iris images of each pair. The experiments showed that our MCCSN achieved 100% test accuracy and 0.07% test loss.

Keywords: Convolutional Siamese Network, Heterogeneous Iris recognition, Mutual Component Distance.

I. INTRODUCTION

The science of recognizing individuals based on their distinctive characteristics is called biometric. According to these characteristics, biometric systems are categorized as physiological, and behavioral. Behavioral biometric systems include all methods that depend on computation and data derived from an action, therefore indirectly measure the characteristics of the human body. Voice, keystroke and signature are considered behavioral characteristics. Physiological biometric systems comprise all methods that depend on direct computation of a specific part of person body. Iris, face, fingerprint, and DNA are considered physiological characteristics [1].

Traditional methods adopted for personal identification can be classified into either knowledge-based methods or token-based methods. For example, knowledge-based methods can use password created by the user for identification, whereas Identification (ID) cards are used by token-based methods. Nevertheless, traditional methods get unreliable if, for instance, the password is forgotten or the card is lost. Consequently, the demand for reliable identification methods becomes highly significant research area [2]. The iris is one of the most reliable biometrics used for security purposes. The human iris has extensively rich patterns. The details of iris texture are randomly determined for the human eye through the embryonic development, and they vary from one individual to another even between the right and the left eyes of the same individual. Moreover, the environmental impacts cannot change the texture of the iris [3].

Although iris recognition is relatively modern field, the progress in this field has been so fast and effective. The approaches to adopt machine learning methods are even more modern. One of the effectively used machine learning techniques in iris recognition systems is Artificial Neural Network (ANN) [4].

Over past years, sensors for obtaining iris images had significant changes, by upgrading existing sensors and developing new ones [5]. When the iris images are obtained by different sensors, the recognition rate will be degraded due to heterogeneity of iris patterns. Therefore, the performance of the recognition system will not be as accurate as when all images are obtained by the same sensor. Such issue is called “cross-sensor iris recognition” [6] or “heterogeneous iris recognition” [7]. The variation among different types of iris sensors such as optical lens, illumination wavelength and image resolution causes heterogeneous characteristics of iris patterns, as a result, heterogeneous iris recognition is rising as a significant issue [8].

The reliability of iris recognition system is considered the motivation of this paper. The iris recognition system is highly useful for several applications of computer vision. The iris patterns have many advantageous that make the iris recognition system one of the most reliable biometric systems. Moreover, the iris is highly protected since it is an internal organ of the eye. It is also considered relatively stable feature over lifetime.

The main contributions of this paper can be summarized into the following points:

— Design of a robust model based on Siamese architecture of neural networks, where the parameters of this model are shared between heterogeneous iris

samples and initialized simultaneously. These sharable parameters can help to decrease the error rate since the difference will be only between the inputs.
— Design of a learnable distance measure module to decrease the distance between the heterogeneous iris samples of same subject and increase the distance between the heterogeneous iris samples of different ones.
— Time reduced by applying transfer learning technique.
The identical branches of proposed model are formed from a pre-trained model, where these branches are frozen during training the proposed model.

II. LITERATURE SURVEY

The amount of publications that address the problem of heterogeneous iris recognition is little. Garea Llano *et al.*, [9] proposed a cross-sensor iris verification system for iris images acquired by different sensors, by applying fusion methods at the level of segmentation stage. Their system comprises capturing one or more images of the iris for the same subject by multiple sensors, and, then, applying at least two segmentation algorithms to localize the pupil and iris boundaries. Liu *et al.,* (2016) proposed a deep learning based framework called DeepIris, for heterogeneous iris verification. Their model contains a layer called pairwise filter layer, which takes pairs of heterogeneous iris images as input, where a couple of pairwise filters convolve the input images and summarize their feature maps to generate the similarity map [10]. Their experimental results have shown that the suggested method achieves promising performance for both cross-sensor and cross-resolution iris verification. Gangwar and Joshi (2016) also proposed a deep learning based method for iris representation [11]. Their model named as DeepIrisNet, is constructed based on deep Convolutional Neural Network (CNN) architecture. The experiments showed that the deep architecture of CNN can represent the micro-structures of iris patterns effectively. Nalla and Kumar (2016) proposed a domain adaptation framework, based on the nearest neighbor classification, to address the cross-domain (cross-spectral/cross-sensor) iris recognition issue [12]. Their experiments showed that the improvement in the performance of proposed model in the cross-spectral iris recognition is higher in comparison with the performance of the cross-sensor iris recognition. Liu *et al.*, (2017) proposed a code-level approach in heterogeneous iris recognition [8]. Their model transforms the number of iris templates in the probe into a homogenous iris template based on gallery sample.

Table 1: Summary of related work on heterogeneous iris recognition.

Ref.	Year	Framework	Accuracy%
[9]	2015	Fusion-based method at the level of segmentation stage	96.1
[10]	2016	Deep learning based method	95
[11]	2016	Deep learning based method	98.78
[12]	2016	Domain adaptation approach	89.92
[8]	2017	Code-level approach	98.74

The model is experimented on matching cross-sensor, high-resolution versus low-resolution and, clear versus blurred iris images. Table 1 provides a summary of all discussed related works for heterogeneous iris recognition problem.

III. METHODOLOGY

In this section, the general concepts of iris recognition, neural networks and deep learning, that address the problem of heterogeneous iris recognition, are explained.

A. Iris Recognition System

Iris recognition system is a special system used to recognize individuals based on their distinctive iris patterns. The system is constructed, after iris image acquisition, from four main parts: iris segmentation, iris normalization, feature extraction and recognition (matching) [13]. In segmentation stage, the iris and pupil boundaries are localized and the region between them is segmented. Then, the segmented region is transformed from Cartesian coordinates to polar coordinates, producing a fixed rectangular region during the normalization stage. Feature extraction draws out the biometric templates from normalized image and finally, matching these templates strictly. Fig. 1 shows the block diagram of iris recognition stages.

B. CNN based feature learning

Iris has a unique texture for each individual. This texture represents the patterns of the iris. Feature extraction methods extract these patterns in an iris image. Several methods have been developed for effective feature extraction, such as Gabor filter, Log-Gabor Filter, Discrete Cosine Transform, Discrete Wavelet Transform, and Contourlet Transform [14].
Currently, much attention is paid to CNN based feature learning method in which, the image is fed to the CNN, then the feature learning algorithm extracts the features of input image directly [15]. Modern deep CNNs revealed a phenomenon that when these networks trained on images their first convolutional layers tend to learn features that are similar to Gabor filter features [16].

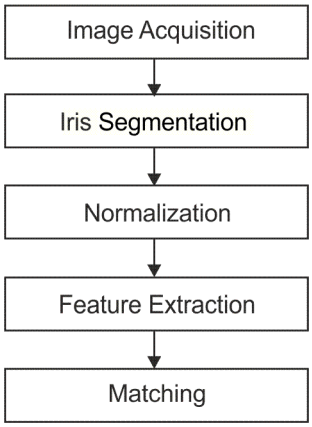

Fig. 1. Block diagram of iris recognition stages [17].

C. Manhattan Distance (MD)

Iris matching means finding similarity between iris templates generated from feature extraction stage. One of matching methods that has the least computational complexity is Manhattan Distance (MD). The MD metric

computes the distance between two points along axes at right angles. The MD between two vectors is the sum over absolute differences between their components. It is an alternative measure to Euclidean Distance (ED), used to reduce the computational complexity. The MD is 0 for vectors that are exactly the same, otherwise is more than 0 To compute the distance between two iris feature vectors, the MD function is expressed as follows [18]:

$$MD = \sum_{i=1}^{N} \left| \vec{x}_i^1 - \vec{x}_i^2 \right| \quad (1)$$

where $\vec{x}_i^1$ and $\vec{x}_i^2$ are the components of vectors $\vec{x}^1$ and $\vec{x}^2$ respectively, and N is the length of the vectors.

D. Artificial Neural Network (ANN)

ANN is an alternative computational model to the instruction based programming, designed to simulate the human's brain. Although ANNs are inspired by neuroscience, they are not biologically in detail. The methods of ANN are highly drawn from statistical physics [19]. ANN consists of a number of simple processing units each one called neuron, these neurons are operating in parallel, communicating by distribution signals to neurons in other level over a large number of weighted connections. The connections are adapted during training through associated weights to get better performance just like human brain. Each neuron applies a nonlinear function usually called activation function to its input signal to obtain output signal. Mathematically, the output on the neuron j can be expressed as follows [20]:

$$v_j = \sum_{i=1}^{n} x_i . w_{ji} + b_j \quad (2)$$

$$y_j = \varphi(v_j) \quad (3)$$

where x_i represents the input signals, w_{ji} represents the associated weights of neuron j, b_j denotes the bias value which is an additional parameter used to adjust the output, and φrepresents the activation function.

E. Activation Function

Activation function takes the output of the summing function v_j as input and defines the output of neuron y_j. The activation functions can be linear or non-linear. The activation functions adopted in proposed approach are explained below:

1) Sigmoid function: Sigmoid function is the most commonly used activation function in ANNs. It squashes output values between 0 and 1. Therefore, it is considered the best choice for neural networks of output value ranging from 0 to 1. Sigmoid function can be described in mathematical form as follows [21]:

$$y_j = \frac{1}{1+e^{v_j}} \quad (4)$$

2) Rectified Linear Units (ReLU): ReLU is a threshold-based activation function. It simply outputs 0 when $v_j \leq 0$ and outputs a linear function when $v_j > 0$ Mathematically, the function can be expressed as follows [22]

$$y_j = max(0, v_j) \quad (5)$$

3) Softmax: The softmax function is commonly used in multiclass classification neural networks. It takes as input a vector of k real numbers, and outputs a vector of $\boldsymbol{k}$ probabilities, where the $i\text{-}th$ value in the output vector corresponds to the $i\text{-}th$ class in the input vector. This function can be expressed as follows [23]:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad (6)$$

where z_i represents the probability of class i and k denotes the total number of classes.

F. Supervised Learning

Supervised learning algorithms are one of the machine learning algorithms in which the desired output is encoded with training dataset and the algorithm learns the patterns based on the target output encoded. Supervised learning is commonly used in classification problems, where the algorithm learns from the training dataset and applies classification or prediction on the test dataset [24].

G. Deep Learning

Recently, large datasets and the increasing of computing power have become easily available. Consequently, scaling up machine learning systems to be more powerful has become possible. In the context of neural networks, deep learning means that the network has more than one hidden layer. Nowadays, the numbers of layers used in deep learning range from five to more than a thousand [22].

H. Convolutional Neural Network (CNN)

CNN is the commonly used deep network form which was inspired by animal's visual cortex. They are widely used in several domains, such as object recognition, object tracking, pose estimation, text detection and recognition, visual saliency detection, action recognition, scene labeling, and many more [25]. CNN is mainly composed of multiple convolutional layers, pooling layer, and Fully Connected (FC) layers as shown in Fig. 2. Each convolutional layer generates a consecutively higher-level abstraction for the input data, called a feature map, which conserves unique information [22]. In addition to fundamental layers, some special layers can be added to accelerate the performance of CNN such as, Dropout layer [26], and Batch Normalization (BN) layer [27]. These layers are discussed next.

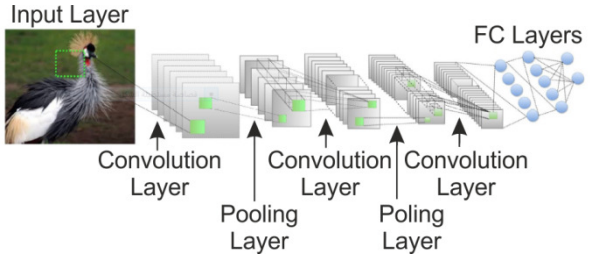

Fig. 2. Common CNN architecture [16].

1) Convolutional Layer: Convolutional layer represents the key unit of CNN. It is simply, a set of neurons arranged to form the feature maps. The parameters are set of learnable filters convolving with the input feature map to generate a separate two dimensional activation maps stacked together, producing the output feature map. Neurons of the same feature map share the parameters; therefore, the network complexity is reduced by reducing number of parameter [25].

2) Polling Layer: Pooling is a computational operation that reduces the dimensionality of feature maps. Pooling increases the network robustness and enables it to be invariant under distortions and small shifts. In pooling, a set of values within a receptive field are grouped into a smaller number of values, according to the pooling form (max or average pooling) and the size of its receptive field. Typically pooling is performed over on non-overlapping blocks where the stride size is equal to the pooling size [22].

3) Fully Connected (FC) Layer: FC layer is also called Multi-Layer Perceptron (MLP). In this layer, neurons are arranged in one dimension and fully connected to all neurons in the preceding layer, as in traditional networks [25]. Usually, a small number of FC layers are added after the convolutional layers for classification task [22].

4) Dropout Layer: Several approaches have been presented to reduce overfitting neural networks. One of these methods is Dropout technique, which prevents overftting problem and provides a way to combine many different architectures of ANN efficiently. The term Dropout means dropping out (temporarily removing) randomly chosen units in a neural network, along with all their connections [26].

5) Batch Normalization (BN) Layer: BN is a technique used to reduce the phenomenon of Internal Covariate Shift (ICS), based on normalizing each feature in layer inputs, by making it have the mean of zero and the standard deviation of 1 [27].

I. Siamese Neural Network (SNN)

SNN is metric learning neural network with two identical neural network branches of shared weights initialized and update simultaneously during training [29]. In SNN, the network branches are responsible for feature extraction from the inputs, while the joining layer learns the metric between the extracted feature vectors. With this characteristic, SNN has been used in several application with many modifications have been developed [28]. Fig. 3 shows an abstract architecture of the SNN.

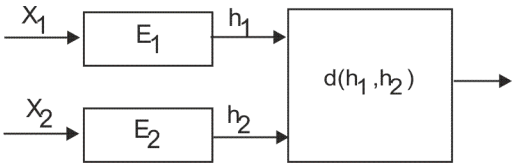

Fig. 3. Abstract architecture of SNN [29].

J. Training of Feed-forward Neural Network (FNN)

In general, training the FNN comprises two main consecutive parts: the forward pass when the outputs are configured according to the given inputs and the current parameters, and the backward pass when the parameters are updated according to the error [29].

1) Forward Pass: Forward pass is the easy part of training process in which the activation of each neuron in one layer is computed according to the neuron inputs and learning parameters, and then those activations are used as the inputs for the next layer where the activations are computed according to next layer parameters, and so on until the final output of network is obtained [29].

2) Computing the Error: The error is simply a function of difference between the outputs and the targets [29]. Several functions are used to compute the error of the networks. These functions are called loss or cost functions. The loss functions used in proposed approach are clarified bellow:

a) Cross Entropy Loss (CEL): CEL loss function is widely used in classification field. Mathematically, it can be expressed as fallows [29]:

$$\mathcal{L} = -\frac{1}{n}\sum_{k=1}^{n}[t_k \ln y_k + (1-t_k)\ln(1-y_k)] \quad (7)$$

where n is the total number of training samples, t_k is the target value and y_k is the output value.

b) Contrastive Loss (CL): CL loss function is highly useful in Siamese networks. Unlike traditional loss functions, such as CEL, where the error is computed as a sum over samples, the CL function computes the error over pairs of samples. This function can be written as fallows [30]:

$$\mathcal{L} = (1-y)\frac{1}{2}(ED)^2 + (y)\frac{1}{2}[max(0, m-ED)]^2 (8)$$

where y is the similarity label, $m > 0$ is a margin, and ED denotes the Euclidean Distance between the outputs of SNN branches.

3) Backward Pass: The method in which the errors are sent backwards through the network is called Back Propagation (BP) of error. BP algorithm is used to compute the gradient in weight space of FNNs. It is about understanding how the cost function is sensitive to the adjustment of the weights in a network [29].

K. Training Optimizers

Actually, the loss function, used in BP is an average over loss functions for individual samples. This means the gradient is an average over gradients computed separately, for each input. When dealing with large number of training samples, this strategy can be, significantly time consuming, thus learning gets slower. An idea to speed up learning regardless the number of training samples is to consider the gradient by computing the loss for a mini batch of randomly chosen training samples. This method is termed as Stochastic Gradient Descent (SGD) [31]. A variation of SGD, called Momentum-based SGD [32], is used to accelerate the original algorithm and reduce the oscillations around local minima that appear with SGD, via using a velocity vector generated by adding a fraction to the gradient vector, and then using this vector to update the weights instead of the original gradient vector. An adaptive gradient algorithm called Root Mean Square Propagation (RMSProp) [33], is used to make the convergence easier by using adaptive learning rate through dividing gradient values by the root of squared gradient.

L. Transfer Learning

The transfer learning means training a base network on base dataset and task, and then transferring the learned feature layers to another network called target network to be trained on new target dataset and task. Usually, this is done by choosing the first *n* layers of the trained base network to be the first *n* layers of the new target network. Then, additional layers are added to the target network proportional to the new target task. The additional layers of the target network are the only layers that are trained toward the target task, while the transferred feature layers are left frozen, meaning that the error is propagated into additional layers only [16].

IV. PROPOSED SYSTEM

The framework for proposed heterogeneous iris recognition system is shown in Fig. 4. This framework comprises three phases: the CNN training phase, Mutual Component Convolutional Siamese Network (MCCSN) training phase, and MCCSN testing (identification) phase. All phases share the fundamental preprocessing module of the system. In CNN training phase, the base CNN is trained for feature learning. During this phase, the parameters are automatically learned and employed to extract feature vectors of input iris images during MCCSN training phase, where two

branches constructed from the base CNN are used for input pairs. In MCCSN training phase, the model learns the similarity and dissimilarity between feature vectors of each input pair. The parameters for learnable distance measure function of MCCSN are adapted and used for learnable matching process during identification phase. The architecture of MCCSN is of SNN architecture type in which the parameters of both CNN branches are shared.

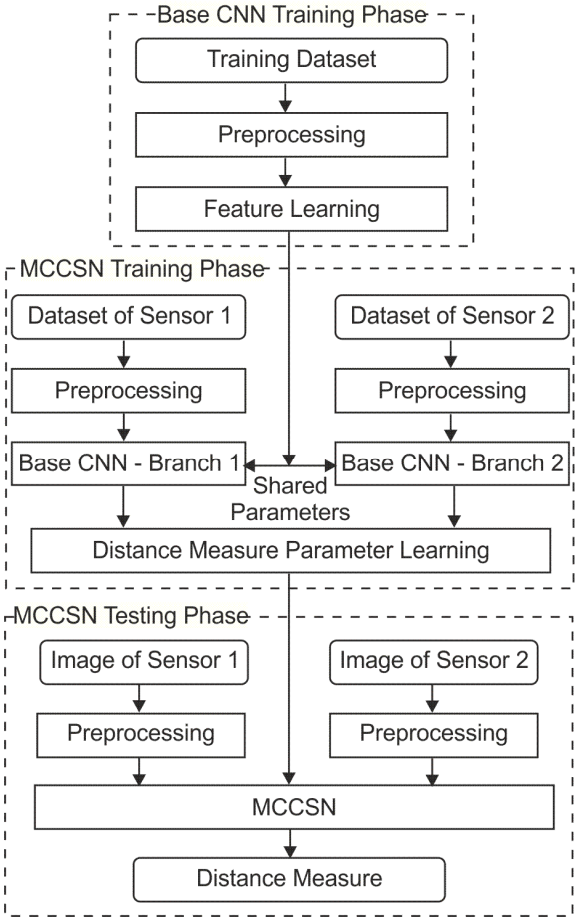

Fig. 4. The framework of proposed system.

A. Preprocessing Module

The preprocessing module consists of four main stages: iris segmentation, normalization, image enhancement, and finally, selecting Region of Interest (ROI) from the lower half of iris, which is relatively noise free. The input of this module is the iris image and the output is a noise free ROI.

B. The Architecture of MCCSN

The two branches of MCCSN are formed from a pre-trained base CNN. These branches are responsible for feature extraction process based on what feature learning that the base CNN gained during training for classification. The output vectors of two branches are combined and passed to the Mutual Component Distance (MCD) layer to compute the distance between mutual components of feature vectors, and then the resultant vector is passed through three of FC layers to decrease the error. Next layer is BN layer to enhance the training process. Finally, the distance is measured by the weighted sum layer and the result is squashed between 0 and 1 by using sigmoid function, Fig. 5 describes the MCCSN architecture.

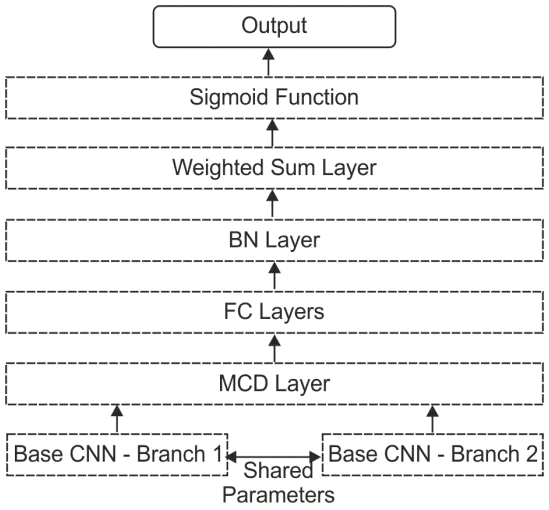

Fig. 5. The architecture of MCCSN.

1) The Architecture of Base CNN: The base CNN consists of a set of convolutional layers with filters of varying size and stride of 1. The activation function applied in all convolutional layers is a ReLU function. The convolutional layers are optionally followed by max-pooling layer with filter size of 2 and stride of 2. The last layer before the output layer is a flatten layer which reshapes the convolutional feature map to a feature vector. The BN layer is added to reduce ICS, and dropout layer is used to prevent the network from overfitting. The output layer is FC layer activated by softmax function. The base CNN is designed deeply and carefully. The output feature maps before flatten layer are one dimensional, so that the flatten layer just rearranges them without changing their shapes. Table 2 states the base CNN layers where the preprocessed image is passed through them. The output shape of input, convolutional, and pooling layers is described in a three dimensional form of (height × width × depth), whereas the output shape of flatten and later layers is described in a one dimensional form.

2) Mutual Component Distance (MCD) Layer: Images obtained by different sensors can have heterogeneous characteristics. Using one of distance measure functions to match between such images can result in false recognition. The main idea of MCCSN is to create a learnable distance measure function to learn the similarity and dissimilarity between heterogeneous iris images. The distance measure function chosen to be learnable is the MD function that has been expressed in Eq. (1). The first step to create such learnable function is by breaking apart Eq. (1) into two steps, as follows:

$$\vec{m} = |\vec{x}_i^1 - \vec{x}_i^2| \quad (9)$$

$$MD = \sum_{i=1}^{n} \vec{m}_i \quad (10)$$

The components of the MCD vector $\vec{m}$ represent the absolute differences between the mutual components of feature vectors corresponding to the input pair. Eq. (9) is implemented in MCD layer where the inputs of this layer are the feature vectors generated by the identical branches of MCCSN, simultaneously. Eq. (10) is postponed to the weighted sum layer in order to reformulate it as a learnable function.

Table 2: The proposed base CNN layers.

Layer	Activation	Stride	Output shape
Input	–	–	48 × 210 × 1
Convolution	ReLU	1	44 × 206 × 32
Convolution	ReLU	1	40 × 202 × 32
Convolution	ReLU	1	37 × 199 × 64
Convolution	ReLU	1	34 × 196 × 64
Pooling	–	2	17 × 98 × 64
Convolution	ReLU	1	15 × 96 × 128
Convolution	ReLU	1	13 × 94 × 128
Pooling	–	2	6 × 47 × 128
Convolution	ReLU	1	4 × 45 × 256
Convolution	ReLU	1	2 × 43 × 256
Pooling	–	2	1 × 21 × 256
Flatten	–	–	5376
BN	–	–	5376
Dropout	–	–	5376
FC	Softmax	–	108

3) FC Layers: Applying weighted sum over MCD vector components directly increases the error rate. In our proposed system, three of FC layers are used to decrease the error. These layers act as dimension reduction layers, since the size of MCD vector $\vec{m}$ is reduced while it is passed through them, generating $\vec{m}'$. The activation functions of all these layers are ReLU.

4) BN Layer: Passing the MCD vector through FC layers makes the impact of ICS phenomenon appear again. Since the output of MCCSN model is ranging between 0 and 1, it is necessary to adopt a BN layer after the FC layers to reduce this impact, thus help the model to converge faster.

5) Weighted Sum Layer: In the weighted sum layer, Eq. (10) is reformulated to be in learnable form. This operation is inspired by the foundation of ANN, which is the neuron. The mathematical form of the neuron is expressed in Eq. (2) which is just a summation over elements that are multiplied by weights and adjusted by bias value, as well Eq. (3) which represents the activation function to generate the final output. Based on the above, the weighted sum over all components of the vector $\vec{m}'$ can be expressed as follows:

$$MD_w = \sum_{i=1}^{n} \vec{m}_i{}' . w_i + b \qquad (11)$$

By this equation the second step of original MD distance function is written as a learnable function. Since the desired output is 0 for similar pairs and 1 for dissimilar pairs, the computed distance is activated by a sigmoid function σ to squash output between 0 and 1 as follows:

$$y = \sigma(MD_w) \qquad (12)$$

During MCCSN training phase, the learning parameters are updated according to the desired output associated with each pair, repeatedly until convergence.

C. Generating Multiplicative Dataset

Dealing with dataset that consists of images acquired by different sensors gives an opportunity to expand it by applying Cartesian product. Let *A* be the set of sample images for a given subject, obtained by sensor1, *B* the set of sample images for the same subject obtained by sensor2, and *C* the set of sample images for other subjects, obtained by sensor2 so that *C* is equivalent to *B*. The Cartesian product of *A* and *B*, denoted by *A* × *B*, is the set of all similar pairs (*a*, *b*) where *a* is a member of *A* and *b* is a member of *B*. The Cartesian product of *A* and *C*, denoted by *A* × *C*, is the set of all dissimilar pairs (*a*, *c*) where *a* is a member of *A* and *c* is a member of *C*. The union of *A* × *B* and *A* × *C*, denoted by (*A* × *B*) ∪ (*A* × *C*), represents the final dataset which contains all similar and dissimilar pairs. The steps for generating multiplicative dataset are described in Algorithm 1.

Algorithm 1: Generating Multiplicative Dataset

Input: *Subset1* // dataset of sensor 1
Subset2 // dataset of sensor 2
Output: *pairs[]* // similar and dissimilar pairs
labels[] // similarity indicating labels

Begin
for A_k in (subset1) // all subjects in dataset1
  for a_i in (A_k) // all elements of set *A*
    for b_j in (B_k) // all elements of set *B* in dataset2
      similar[] ← (a_i, b_j)
    end for
    for b_j in (B_k) // all elements of set *C* in dataset2
      dissimilar[] ← (a_i, c_j)
    end for
  end for
end for
for all similar and dissimilar pairs
  pairs[] ← [*similar*[] , *dissimilar*[]] // arrange pairs alternatively
  labels[] ← [0 , 1]
end for
End

D. Training Schedule

The training schedule comprises two phases: CNN training for feature learning purpose, and MCCSN training for distance learning purpose. The pixel values of iris images in both phase, are simply scaled down to the range between 0 and 1 via dividing each pixel value by 255. This can help in reducing training time.

1) CNN Training Phase: In CNN training phase, the base CNN is trained on iris images for classification. While the model is classifying the inputs it learns the features of different samples. Each input image is shifted by two pixels to the right, down, and to the right and down together, to create three additional samples from each one. This can effectively accelerate the feature learning process. The input images are firstly subjected to preprocessing module. The training parameters are updated by using momentum-based SGD and RMSprop to choose the most effectively transferred learning between the two optimizers. The gradient is computed through the CEL function. The training is stopped after some epochs when the loss stops decreasing to ensure the model generalization and prevent from overfitting with the assistance of dropout operation.

2) MCCSN Training Phase: In MCCSN training phase, similar and dissimilar pairs are fed to MCCSN model alternatively, with a binary label indicating their similarities. The identical branches created from the base CNN are frozen and the learnable parameters are initialized simultaneously to generate feature vectors. The feature vectors are obtained from the dropout layer of the base CNN and the output layer is neglected for new target task. The vectors then, passed through MCCSN layers, to compute the distance between the vectors of each pair. The parameters are updated according to the binary label value attached with each pair, if the label is 0 then the parameters are updated to decrease the output, and if the label is 1 then the

parameters are updated to increase the output. These parameters are updated by using RMSprop. The gradient is computed by using the modified CL function. The modification comprises replacing the ED by $\sigma(MD_w)$. As in CNN training phase, the training is stopped after some epochs when the loss stops decreasing. Only the parameters of FC layers and weighted sum layer are updated during training while MCD layer just calculates the absolute difference between mutual components of feature vectors and has no learnable parameters. The training procedure of MCCSN is briefly described in Algorithm 2.

Algorithm 2: MCCSN Training
Input: (I_i^1, I_i^2) // training image pairs
y_i // similarity indicating label
Output: Similarity score and learnable parameters
Begin
Initialize all parameters using pre-trained base CNN
While not converge **do**:
Input image pairs and get feature vectors
Input feature vectors to MCD layer and get $\vec{m}$
Pass MCD vector to FC layers and get $\vec{m}'$
Apply BN
Compute MD_w
Squash MD_w using sigmoid function
If y_i= 0 **then**
update W to decrease MD_w
else
update W to increase MD_w
end if
end while
End

E. MCCSN Testing

To test the trained MCCSN model, the system takes the input iris image obtained by sensor1 and creates image pairs with all images in the dataset of images obtained by sensor 2, where the first image in all pair is the input image. The learnable parameters of CNN are used to generate feature vectors, while the learnable parameters of MCCSN are used to compute the distance between the sample images of each pair. The output of the model is ranging between 0 and 1, which represents the squashed value of distance. The similar pairs have a distance close to 0, and dissimilar pairs have a distance close to 1. Based on a predefined threshold, the system returns the person identity that has minimum distance with input image, otherwise, returns unknown person.

V. RESULTS AND DISCUSSION

Since we adopted the transfer learning technique in this approach, two datasets are employed in this experiment: CASIA Dataset Version 1.0 (CASIA V1.0) which is used in the CNN training phase and CASIA Dataset Version 2.0 (CASIA V2.0) which is used in the MCCSN training phase. All images of both datasets are in bitmap (*.bmp) format. Table 3 summarizes the characteristics of both datasets.

The experiments have been carried out under the environment of Windows-7 64-bit operating system, laptop computer processor: Intel Core i7 CPU, 2.80 GHz, and (4GB) RAM.

Table 3: The The characteristics of CASIA V1.0 and CASIA V2.0.

Characteristics	CASIA V1.0	CASIA V2.0	
		Device 1	Device 2
Sensor	Homemade CASIA close-up iris camera	OKI IRISPASS-h	CASIA-IrisCamV2
Environment	Indoor	Indoor	Indoor
Session	Two	one	one
No. of classes	108	60	60
No. of images per class	7	20	20
Resolution	320×240	640×480	640×480

A. Preprocessing

Fig. 6 shows the results of preprocessing module. The size of normalized image is (420×60), whereas the size of ROI is (210 × 48).

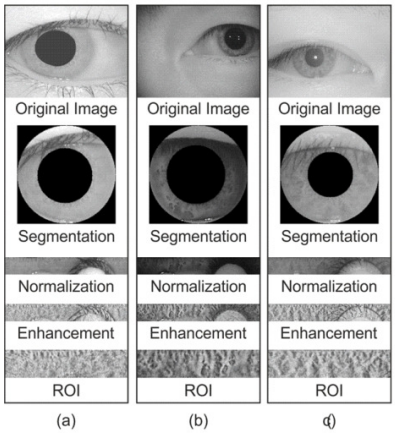

Fig. 6. Preprocessing of sample iris images, (a) CASIA V1.0, (b) sensor1 of CASIA V2.0, and (b) sensor2 of CASIA V2.0.

B. Training the Base CNN

The base CNN is trained on the classification of CASIA V1.0 images. After applying shifting on the images of dataset, the result was three samples plus the un-shifted one. As a result, the number of samples for each class became 28, where 23 of them are used for training and remaining 5 for testing. The total number of dataset images is 3024. The categorical CEL is used to compute the cost during training process. The momentum-based SGD with momentum value of 0.9 and RMSProp are adopted to train the model with the same initial learning rate of 0.001. The learning rate in both experiments is reduced by the factor value of 0.5 when the loss stops to decrease. The model is trained with batch size of 128. Table 4 lists the hyper parameters and the model accuracy and loss by using both momentum-based SGD and RMSProp optimizers. Both models have the same accuracy, while there is a small difference in loss which is 0.0092. Fig. 7 shows the plot of loss and accuracy for test dataset in both experiments.

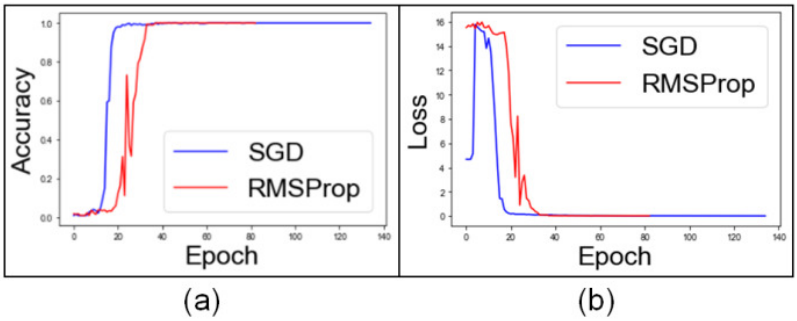

Fig. 7. A comparison between using momentum-based SGD and RMSProp for training the base CNN, (a) the plot of accuracy, and (b) the plot of loss.

Table 4: Base CNN Accuracy and Loss with Corresponding Hyperparameters.

Batch size	learning rate		Loss function	Optimizer	Loss%	Accuracy%
	Initial value	Reducing factor				
128	0.001	0.5	CEL	RMSProp	2.19	100
128	0.001	0.5	CEL	SGD	1.27	100

Table 5: MCCSN Model Accuracy and Loss With Corresponding Hyperparameters

Batch size	learning rate		Loss function	Base Optimizer	Loss%	Accuracy%
	Initial value	Reducing factor				
128	0.00004	0.9	CL	RMSProp	5.41	92.50
128	0.00004	0.9	CL	SGD	0.07	100

C. Training the MCCSN

The advantage of feature learning that the base CNN has acquired during training for classification is adopted to extract feature vectors of each input pair simultaneously during MCCSN training phase. This is done by discarding the last layer of the base CNN which is the classification layer since the new target task is extracting features. The base CNN branches are frozen during the training process of MCCSN. The trainable part of MCCSN begins after MCD layer. The set of FC layers including the weighted sum layer are trained on CASIA V2.0, after being rearranged as similar and dissimilar pairs by using Cartesian product and fed alternatively to the model. The modified CL function is used to compute the cost during training process. The RMSProp is adopted to compute the gradient. A small initial learning rate value of 0.00004 is used with reducing factor value of 0.9 when the loss stops to decrease. The model is trained with batch size of 128. Table 5 lists the hyperparameters and the model accuracy and loss by using both versions of base CNN.

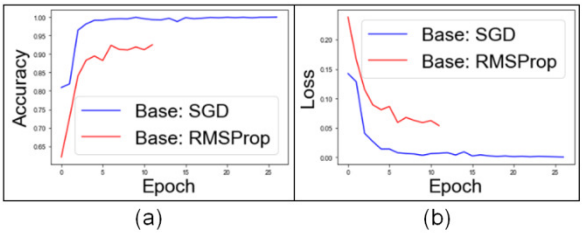

Fig. 8. A comparison between using the two versions of base CNN to initialize MCCSN, (a) represents the plot of accuracy, and (b) represents the plot of loss.

Although both versions of base CNN have same accuracy and small difference in loss, using the model learned with momentum-based SGD to form MCCSN branches gave accuracy of 100% and loss of 0.07%, while using the model learned with RMSProp gave accuracy of 92.50% and loss of 5.41%. This means that the first model has the feature learning advantage over the later one. Fig. 8 shows the plot of loss and accuracy for test dataset in both experiments.

D. The Time Reduced Using Transfer Learning

The time reduced by using transfer learning technique can be obtained by first, calculating the time consumed during training the base CNN model. This means multiplying the number of epochs by the time per one epoch. As expressed below:

$$CNN_{total\ time} = CNN_{epochs} \times CNN_{epoch\ time} \quad (13)$$

Then, the total time calculated by Eq. (13) is divided by the number of epochs expended during MCCSN training phase. The resultant value represents the share time added to each epoch time of MCCSN training phase when using transfer learning technique i.e. when the CNN branches are frozen. This value represents the total time for one epoch of MCCSN training. Mathematically, this is expressed as follows:

$$MCCSN_{total\ epoch\ time} = \left(\frac{CNN_{total\ time}}{MCCSN_{epochs}}\right) + MCCSN_{epoch\ time} \quad (14)$$

The time saved with transfer learning strategy can be estimated by comparing the value computed by Eq. (14) with the epoch time consumed without using transfer learning technique. This is done by calculating the difference between the two values. For instance, the total epoch time in minutes for training the MCCSN by using RMSProp with learning rate of 0.00004 and the base CNN trained with momentum-based SGD with learning rate of 0.001, is calculated as follows:

$$MCCNN_{total\ epoch\ time} = \left(\frac{5.37 \times 135}{27}\right) + 7.55 \ = 34.38$$

The experiment showed that time consumed in one epoch during training the MCCSN model without using transfer learning is 46.25 minutes. By computing the difference between this value and the value calculated in Eq. (14), which is 34.38, the result we get is 11.87 minutes. This is the time reduced for only one epoch.

E. Testing the MCCSN

Fig. 9 shows the distance measure between 36 pairs of iris images. The image samples of the first row belong to the subset of sensor1, the image samples of the first column belong to the subset of sensor2, and the intersection of the row with the column represents the distance between the samples.

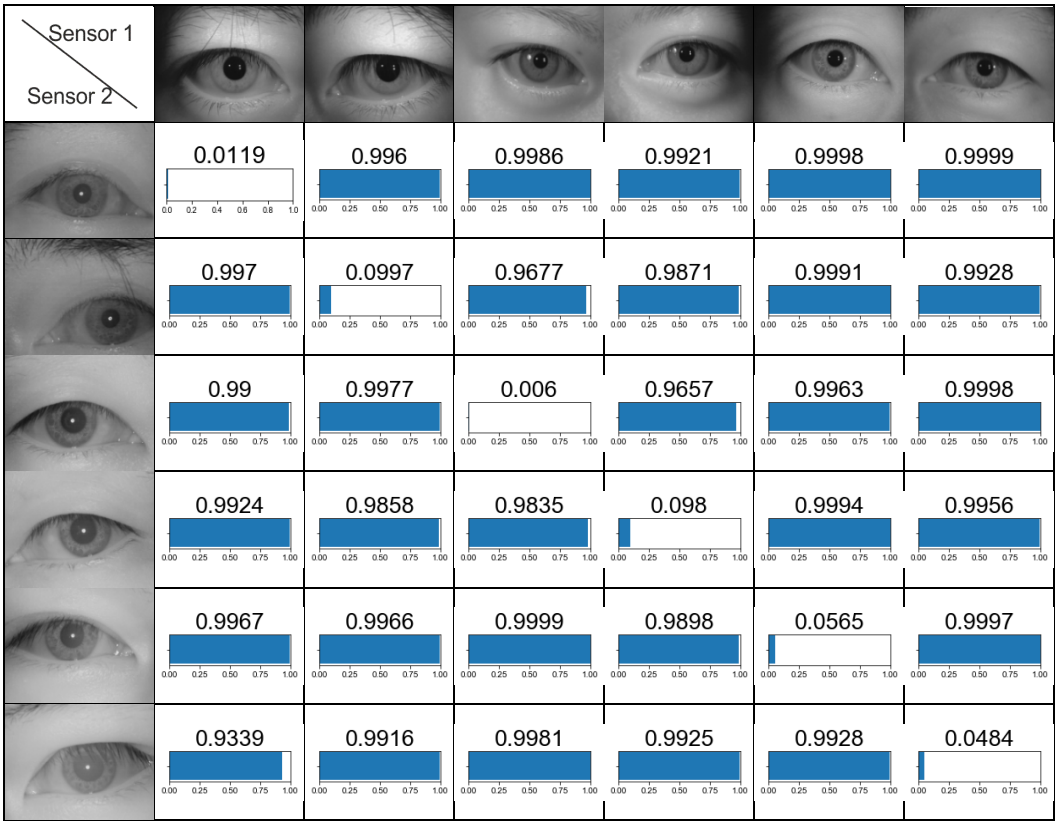

Fig. 9. Distance measure between some image samples, arranged in pairs where the images in these pairs are obtained by different sensors.

F. Comparing Proposed MCCSN with Other Methods

We conclude the evaluation of our MCCSN by comparison with the latest frameworks discussed in literature survey. Table 6 lists the models with their accuracies.

Table 6: A Comparison of MCCSN with other Frameworks.

Framework	Accuracy%
DeepIrisNet (2016) [11]	98.78
Code-level approach (2017) [8]	98.74
MCCSN	100

VI. CONCLUSION

In this paper we have presented a deep learning based approach to address the issue of heterogeneous iris recognition, based on the architecture of CNN and SNN. CNN based feature learning is an effective method to extract the features of the input image directly fed to the network. SNN is a robust metric learning neural network with two identical branches. However, training a network with two or more branches would increase time complexity of training process. Using transfer learning technique can reduce the time complexity by freezing the branches during training and using their prior knowledge. Searching for hyperparameters that fine-tune the network is the most difficult aspect of establishing a network. It can be performed by carefully testing sets of hyperparameters that should maximize the effectiveness of the learning method. The experimental analysis indicated that the highest recognition accuracy can be achieved if the base CNN is trained by using momentum-based SGD with learning rate of 0.001 reduced by the factor of 0.5, and the proposed MCCSN is trained by using RMSProp with learning rate of 0.00004 reduced by the factor of 0.9.

VII. FUTURE SCOPE

This work can be expanded in different directions. Some ideas for future works are suggested below:

- Training the base CNN on a larger dataset for further performance enhancement, thus, ensuring the generalization of proposed CNN.
- Applying proposed MCCSN on cross-spectral environment where the iris images are obtained by near-infrared versus visible illumination.

- Applying proposed MCCSN on other biometrics, or even on other fields of pattern recognition.

REFERENCES

[1]. Singh, J. P., Jain, S., Arora, S., & Singh, U. P. (2019). A Survey of Behavioral Biometric Gait Recognition: Current Success and Future Perspectives. *Archives of Computational Methods in Engineering*, 1-42.

[2]. Cui, J., Wang, Y., Tan, T., Ma, L., & Sun, Z. (2004, August). A Fast and Robust Iris Localization Method based on Texture Segmentation. In *Biometric Technology for Human Identification* (pp. 401-408). International Society for Optics and Photonics.

[3]. Bowyer, K. W., Hollingsworth, K., & Flynn, P. J. (2008). Image Understanding for Iris Biometrics: A Survey. *Computer Vision and Image Understanding*, *110*(2), 281-307.

[4]. De Marsico, M., Petrosino, A., & Ricciardi, S. (2016). Iris Recognition through Machine Learning Techniques: A Survey. *Pattern Recognition Letters*, *82*, 106-115.

[5]. Matey, J. R.,&Kennell, L. R. (2009). Iris Recognition beyond one Meter. In *Handbook of Remote Biometrics* (pp. 23-59). Springer, London.

[6]. Xiao, L., Sun, Z., He, R., & Tan, T. (2013). Coupled Feature Selection for Cross-Sensor Iris Recognition. In *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)* (pp. 1-6). IEEE.

[7]. Zheng, B. R., Ji, D. Y., & Li, Y. H. (2014, May). Heterogeneous Iris Recognition using Heterogeneous Eigeniris and Sparse Representation.In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 3764-3768). IEEE.

[8]. Liu, N., Liu, J., Sun, Z., & Tan, T. (2017). A Code-Level Approach to Heterogeneous Iris Recognition. *IEEE Transactions on Information Forensics and Security*, *12*(10), 2373-2386.

[9]. Garea Llano, E., Colores-Vargas, J. M., Garcia-Vazquez, M. S., Zamudio Fuentes, L. M., & Ramirez-Acosta, A. A. (2015). Cross-Sensor Iris Verification Applying Robust Fused Segmentation Algorithms. In *2015 International Conference on Biometrics (ICB),* 17-22.

[10]. Liu, N., Zhang, M., Li, H., Sun, Z., & Tan, T. (2016). Deeplris: Learning Pairwise Filter Bank for Heterogeneous Iris Verification. *Pattern Recognition Letters*, *82*, 154-161.

[11]. Gangwar, A., & Joshi, A. (2016). DeepIrisNet: Deep Iris Representation with Applications in Iris Recognition and Cross-Sensor Iris Recognition. In *2016 IEEE International Conference on Image Processing (ICIP)* (pp. 2301-2305). IEEE.

[12]. Nalla, P. R., & Kumar, A. (2016). Toward More Accurate Iris Recognition using Cross-Spectral Matching. *IEEE transactions on Image Processing*, *26*(1), 208-221.

[13]. Harifi, S., & Bastanfard, A. (2015, September). Previous Works About Iris Recognition Stages. In *2015 Forth International Conference on e-Technologies and Networks for Development (ICeND)* (pp. 1-10). IEEE.

[14]. Fasna, K. K., Athira, P., & Remya, K. J. S. (2016). A Review on Iris Feature Extraction Methods. *International Journal of Engineering Research and General Science.*, *4*(2), 663-667.

[15]. Alaslani, M. G., & Elrefaei, L. A. (2018). Convolutional Neural Network Based Feature Extraction for Iris Recognition. *International Journal of Computer Science & Information Technology*, *10*(2), 65-78.

[16]. Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How Transferable are Features in Deep Neural Networks?. In *Advances in Neural Information Processing Systems* (pp. 3320-3328).

[17]. Kamboj, N., & Gupta, P. (2015). A Review on Segmentation Techniques for Iris Recognition System. *International Journal of Engineering and Management Research (IJEMR)*, *5*(2), 14-16.

[18]. Greche, L., Jazouli, M., Es-Sbai, N., Majda, A., & Zarghili, A. (2017). Comparison between Euclidean and Manhattan Distance Measure for Facial Expressions Classification.In *2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)* (pp. 1-4). IEEE.

[19]. Hertz, J. A. (2018). *Introduction to the Theory of Neural Computation*. CRC Press.

[20]. Haykin, S. (2010). *Neural Networks and Learning Machines, 3/E*. Pearson Education India.

[21]. Wanto, A., Windarto, A. P., Hartama, D., & Parlina, I. (2017). Use of Binary Sigmoid Function and Linear Identity in Artificial Neural Networks for Forecasting Population Density. *IJISTECH (International Journal of Information System & Technology)*, *1*(1), 43-54.

[22]. Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE*, *105*(12), 2295-2329.

[23]. Wu, Y., Li, J., Kong, Y., & Fu, Y. (2016). Deep Convolutional Neural Network with Independent Softmax for Large Scale Face Recognition. In *Proceedings of the 24th ACM International Conference on Multimedia* (pp. 1063-1067).

[24]. Dey, A. (2016). Machine Learning Algorithms: A Review. *International Journal of Computer Science and Information Technologies*, *7*(3), 1174-1179.

[25]. Aloysius, N., & Geetha, M. (2017). A Review on Deep Convolutional Neural Networks. In *2017 International Conference on Communication and Signal Processing (ICCSP)* (pp. 0588-0592). IEEE.

[26]. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *The journal of Machine Learning Research*, *15*(1), 1929-1958.

[27]. Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of *International Conference on Machine Learning*, 448–456.

[28]. Utkin, L. V., & Ryabinin, M. A. (2018). A Siamese Deep Forest. Knowledge-Based Systems*, 139*, 13-22.

[29]. Marsland, S. (2015). *Machine Learning: An Algorithmic Perspective*. CRC press.

[30]. Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality Reduction by Learning an Invariant Mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) (2*, pp. 1735-1742). IEEE.

[31]. Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of COMPSTAT'2010* (pp. 177-186).

[32]. Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, February). On the Importance of Initialization and Momentum in Deep Learning. In *International Conference on Machine Learning* (pp. 1139-1147).

[33]. Zou, F., Shen, L., Jie, Z., Zhang, W., & Liu, W. (2019). A Sufficient Condition for Convergences of Adam and RMSProp. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 11127-11135).

How to cite this article: Ridha, J. A. and Saud, J. H. (2020). Mutual Component Convolutional Siamese Network for Heterogeneous Iris Recognition. *International Journal on Emerging Technologies*, *11*(4): 282–291.