VOL. 12, NO. 24, DECEMBER 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

CROSS DOMAIN IRIS RECOGNITION USING DEEP NEURAL NETWORKS

Hemanth B, Dandi Rashmi. G. Hegde and Lokanath M
School of Electronics Engineering, Vellore Institute of Technology University, Vellore, India
Email: hb17863@gmail.com

ABSTRACT

Iris recognition has taken up much of the research space, owing to its heavy usage in security purposes. However, any practical application in safekeeping and defense would require quick results and high accuracy with minimal conditions. Our paper aims to obtain matching between visible and near infrared spectrums with the usage of Multi layer perceptron. We present reproducible experimental results from POLY U database.

Index Terms: feature extraction, image classification, image segmentation, support vector machine, multilayer perceptron.

INTRODUCTION

Personnel identification has, for decades, been a prime topic for research. It has widespread applications across many fields ranging from data analytics to national security. All authentication systems are mostly based on the human characteristics of face, finger, iris and voice. The distinction between people in these characteristics gives image processing techniques a chance to distinguish and match them. Iris recognition is the most advanced form of biometric identification with its factors like speed and accuracy that are higher than other forms.

There are many different algorithms already present that match iris images. Some of these include Daugman, Avila and Li ma. A comparison of their techniques efficiency is given below:

Avila-97.89%
Li ma-98%
Tisse-89.37%
Daugman-99.9%

The problem, however, with all these techniques is that, there is a drastic drop in accuracy when images of two different spectrums are taken. A comparison between the images of near infrared and visible spectrums leads the failure of these algorithms.

In 2016, N. Pattabhi Ramaiah, Ajay Kumar proposed an algorithm in their paper "Towards more accurate iris recognition using cross-spectral matching " that considerably increased the matching efficiency in different spectrums. They propose a classification framework based on NBNN domain adaptation in order to improve matching performance. This is further extended with the help of SPM and mahalanobis distance for classification of images into different categories. With help of these techniques, they were able to extend the accuracy of cross spectral matching.

Our paper explores the idea of using Multi layer perceptron to match images. Multi layer perceptron is a neural network that uses hidden neurons to calculate appropriate weights between inputs and target classes, for prediction. This approach led to higher efficiency in two areas: 1) It has increased the speed of matching 2) The cross spectral matching efficiency reached upto 71.25%.

BLOCK DIAGRAM FOR PROPOSED ALGORITHM

A block diagram illustrating our proposed algorithm is shown in Figure-1.

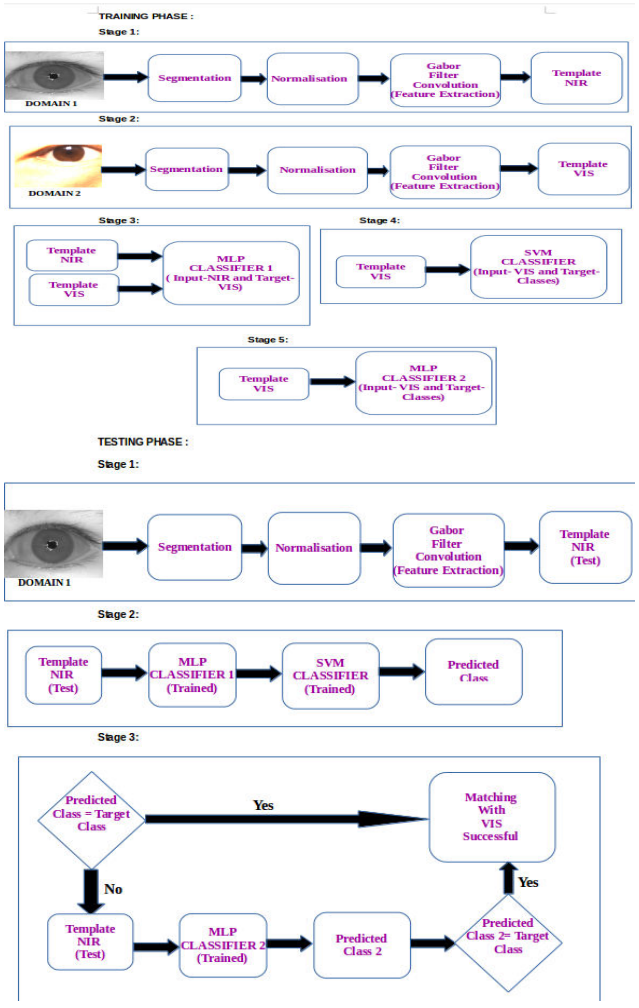

Figure-1. Block diagram of proposed algorithm of for cross domain iris classification and matching.

The training stage involves 5 stages, which will be explained here briefly but will be further elaborated in the coming sections.

- **First and second stage:** These stages involve obtaining the binary templates from each domain through various pre-processing steps and feature extraction by through convolution with a 2-D Gabor filter.
- **Third stage:** In this stage, the templates obtained from the first and second stages are trained using using our multilayer perceptron (MLP) classifier, such that the NIR templates are fed as inputs to achieve VIS images as targets through regression.
- **Fourth stage:** This stage consists of training the VIS templates to achieve classification by feeding them to our support vector machine (SVM) classifier.
- **Fifth stage:** Here, the visible templates are loaded as input to another multilayer perceptron classifier so as to perform classification.

For both the fourth and fifth stages, the same visible templates are used which are obtained from stage 2. This completes the training phase.

The testing phase involves 3 stages which will be briefly looked into here but will be elaborated later.

- **First stage:** In the first stage, test image templates are obtained for NIR images through the same procedure followed in the training phase.
- **Second stage:** In this stage, the test image templates are fed into the trained MLP classifier of the third stage in training, and this output is further fed into the training SVM classifier so as to predict class values.
- **Third stage:** If the predicted class values match that of the expected, the NIR subject is matched to the VIS subject. Now if it didn't match the expected class, we pass the NIR test image values through our second trained MLP classifier from stage five of the training phase, and if the predicted class matches that of the expected, we combine them with the other matched classes from the second stage.

This is thus our algorithm to match near-infrared classes with visible classes and vice-versa.

IRIS DETECTION

Segmentation

The first step in detecting the Iris portion of any eye would be to find it. Segmentation of an eye involves various steps in order to achieve that objective.

The first step is to detect where roughly the Iris could be located. This is done with the use of a transform known as the Hough transform.

Firstly, in order to make detection of circles easy, the prominent edges around the eye is located. This is done through an edge detector called the canny edge. Canny edge uses a simple concept of gradient extraction to take out the edges. A Gaussian filter is first applied on the image of suitable size so as to smoothen the image, and the filtered and convolved image is then used to acquire gradients across various directions.

Mathematically, a 2-D gaussian filter is given by

$$g(x,y) = \frac{1}{2\pi\sigma^2 e^{\frac{-x(n-1)x^2+y^2}{2\sigma^2}}} e^{\frac{-x(n-1)x^2+y^2}{2\sigma^2}} \quad (1)$$

The gradient amplitude is obtained and the various angle across each as well, by simple trigonometry.

For finding out gradients, convolution across each point is done with the following matrices-

G_x= [-1 1;-1 1] & G_Y= [1 1;-1 -1]

Magnitude of the gradients are :

$$M(i,j) = \sqrt{(G_x I(i,j))^2 + (G_y I(i,j))^2} \quad (2)$$

where $G_x I(i,j)$ is the convolved output across each point 0 along x-direction and $G_y I(i,j)$ is the convolved output across each point along y-direction.
The Orientation along each point is given by

$$\delta(i,j) = arctan(G_y I(i,j)/G_x I(i,j)) \quad (3)$$

In-order to make the detected regions more visible so as to extract features later on, a simple contrasting function is used to to enhance the bright and darker regions, in other words, increase the contrast of the region that is going to be detected against those which aren't. The output of this is subjected to non-maxima suppression.

In order to find out whether the magnitudes of the gradients have local maxima in their respective direction, we use non-maxima suppression with an orientation image. After the orientation image is applied, bilinear interpolation is used to find out the local maxima across each direction. Hence edges can be detected from this, and these points are are used further as inputs to a hysteresis function.

Hysteresis is used to segregate the edge points into two distinctions-namely, high and low. The reason behind is that the latter doesn't form an edge while the former does. First the image is made into a column vector and then it is subjected to a constant comparison of each value with a certain threshold. If the value is greater, it is resized and saved and if it is smaller, it is treated as noise pixels and removed.

After edge detection, circle detection is easier. Hough transform involves drawing of circles of different radii until circles are detected. This is down by converting the coordinates and plane into polar form, so that, from the centre of each edge point, different radii circles can be drawn. Now, edge pixels across the same direction and

distance form a line, and hence, in this case, circles are detected.

Using this coordinates of the centre and radii of both the iris and pupil are found, whereas the image with noise values only are stored separately.

Since these are the coordinates and not the actual pixel values of the position of both the iris and pupil, we have to convert them back into Cartesian plane. Thus we find the matrix position of the centre and distance from it to the end of the boundary of pupil and iris in the Cartesian plane, thus the matrix plane. Removal of eyelashes and other such features is done by thresholding the line pixels, by finding out the regions before the boundary and hence removing the other parts as noise, thereby being left with only the detected iris and pupil of the eye.

Normalisation

Normalisation is the process of image reading to a particular standard followed by feature extraction. Due to various variations like lighting, exposure, camera-quality, etc, the radial dimensions of the pupil may change. This therefore creates a problem while extracting the Iris texture will affect the features from the iris and pupil and subsequently affects the matching process. Thus normalization is done so as to remove such variations.

After normalization, the image is convolved with a 1-D Gabor filter to extract features.

A gabor filter set with a given direction gives a strong response for locations of the target images that have structures in this given direction. It helps to extract more important features of the iris, thereby reducing the total size of the original image into a much smaller size, which can be used for faster computations.

The impulse function of a typical 2-D Gabor filter is a product of a sinusoidal function and a Gaussian function.

It is represented as

$$g(x,y,\psi,\gamma,\sigma,\lambda,\theta) = e^{-\frac{x'^2+\gamma^2y'^2}{2\sigma^2}} e^{(i(2\pi\frac{x'}{\lambda}+\psi))} \quad (4)$$

where

$\boldsymbol{\lambda}$ represents what type of sinusoidal wave to use to act as a filter best, $\boldsymbol{\theta}$controls the elliptical rotation, $\boldsymbol{\psi}$ represents the number of times the ellipses have to be shifted from the centre.$\boldsymbol{\sigma}$ is the standard deviation of the gaussian part of the equation.

The gabor filter output is in complex form, and thus, in order to extract a template from it, the filtered image is passed along a mask which has thresholds so as to convert the whole polar template into a binary template. This binary template is used for comparison and matching.

MULTILAYER PERCEPTRON

Introduction

A multi layer perceptron comprises of 3 components-input layer, hidden layer and the output layer. Each layer is made up of units called as neurons, which are responsible for the computations in the MLP. The flow of data takes places between the neurons of each layer, and in our algorithm, each neuron is connected to every other subsequent neuron of each layer. An MLP can have only one each of input and output layers, but can have more than one hidden layers. The direction moves from input to the hidden and ending at the output layer. The data across each layer is defined by a function known as the activation function. The hidden nodes comprises of units known as weight and bias. These two factors influence the input data in-order to provide or predict the output data, via thresholds.

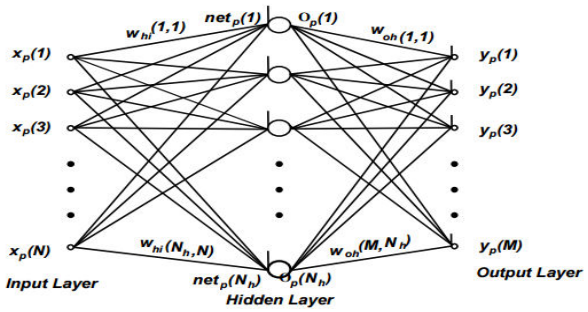

Figure-2. A multi layer perceptron model.

Let $x_p(1)$,$x_p(2)$.....$x_p(N)$ be the input neurons which are the input vectors for the pth training pattern (See Figure. 2)[11]. Hence, for every pattern of every sample, there are input neurons for each as vectors in the training pattern.

Let $y_p(1)$,$y_p(2)$,.....$y_p(M)$ be the output vectors for each sample of each training vector. Thus for each input neuron x_p there is an output neuron y_p. The N-dimensional vector x_p and the M dimensional vector yp consisting of N and M neurons respectively, comprises the input and output vectors for each training sample.

The weights from the input layer to the hidden layer(s) are $w_{hi}(N_h,N)$, where N_h is the hth hidden layer and N is the corresponding neurons from the input layer from which the weights are being calculated. Similarly, the weights from the hidden layer(s) to the output layer are $w_{oh}(N_h,M)$ where M is the corresponding neurons from the output layer.

Let the net input of the rth layer be $net_p(r)$, and the output activation function of the pth layer be $O_p(N_h)$and the total output at the pth output layer is y_p.

Let the activation function chosen be a sigmoid function.

The input to the rth layer is given by

$$net_p(r) = \sum_{s=1}^{N+1} w_{hi}(r,s).x_p(s) \quad (5)$$

And the output of the pth layer is

$$y_p(t) = \sum_{s=1}^{N+1} w_{oi}(t,s).x_p(s) + \sum_{r=1}^{N_h} w_{oh}(t,r).O_p(r) \quad (6)$$

Here, the weights from the input to output neurons are represented as w_{oi} and the weights from the hidden neurons to those of the output are represented as w_{oh}.
O_p is the output activation function for each pth training vector, which is sigmoid in this case.

Hence the overall performance of the multi layer perceptron is measured by calculating the mean square error (MSE).

The equation of MSE is given by:

$$E = \frac{1}{N_v}\sum_{p=1}^{N} E_p \quad (7)$$

where

$$E_p = \sum_{t=1}^{M}[t_p(t) - y_p(t)]^2 \quad (8)$$

Here, E_p is the error of the p[th] vector and t_p is the expected target for the same vector.
The error for the t[th] output vector is thus given by

$$E_t = \frac{1}{Nv}\sum_{p=1}^{M}[t_p(t) - y_p(t)]^2 \quad (9)$$

where the i[th] output vector for the pth training vector is expressed as $y_p(t)$.

V.B. training of perceptron model

Backpropagation training

In backpropagation theorem [11], all the weights are updated using

$$w(r,t)_{new} = w(r,t)_{old} + \alpha.\frac{-\partial E_t}{\partial w(r,t)_{old}} \quad (10)$$

where α is the learning factor.
Hence the gradients are calculated by

$$\frac{\partial E_t}{\partial w(r,t)_{old}} = -\delta_p(r).O_p(t) \quad (11)$$

where $\delta_p(r)$ is known as the delta variable, and is expressed as:

$$\delta_p(r) = \frac{-\partial E_t}{\partial net_p(r)} \quad (12)$$

The delta variables for the output and hidden neurons are given by:

$$\delta_p(s) = F'(net_p(r)).(t_p(s) - O_p(s)) \quad (13)$$

$$\delta_p(r) = F'(net_p(r)).\sum_q \delta_p(q)w(q,r) \quad (14)$$

where F' is the first derivative of the activation function, and q is the number of the layers that follows the r[th] neuron of the previous layer.
Thus the algorithm for back propagation theorem is:

Algorithm 1:

- Initialize the weights and biases and set the learning rate.
- Follow the below steps (3-6) repeatedly till convergence has been reached
- **Step 1:** Initialise the inputs with an activation function
- **Step 2:** Calculated the net input to each neuron from the given weights and biases for each layer.
- **Step 3:** For each net input, calculate the change in output error.
- **Step 4:** Using the output of step 3, update the weights and biases for each layer.

Transfer function

There are two types of functions that determine how the neurons of a system process the data. One if the activation function, which is applied on the input data, gives the total input to the neuron and the other is the output function, which gives the output. The combination of these two gives the transfer function of the input data.

The transfer function used in multilayer perceptron is sigmoidal. There are two types of sigmoidal transfer functions used in the multilayer perceptron algorithm-

Log-sigmoid and hyperbolic tangent sigmoid function.
The equations for each are:

$$\phi(u) = \frac{1}{1+e^{-u}} \quad (15)$$

$$\Psi(u) = tanh(u) = \frac{2}{1+e^{-2u}} - 1 \quad (16)$$

where $\phi(u)$ is the logistic sigmoid function and $\Psi(u)$ is the hyperbolic tangent function

Sigmoidal transfer functions are used in place of many other transfer functions because of its error minimization possibilities through gradients, and the mixing of these functions with other weighted activation functions provide transfer functions which are localized.

Optimization of weights

In-order to compute the weights in the multi layer perceptron with accuracy and speed, stochastic gradient descent is used.

Often in many other weight optimization schemes, they use the full training set to update the weights and calculate gradients if the desired output isn't met in the first epoch itself.

Hence this not only slows down the computational speed of the process, as the whole training set has to be used every time, and so for a very large

dataset, it will take too much time, but also increases the complexity of the process, thereby reducing the optimization.

This is avoided by the stochastic gradient descent (SGD), which only considers a small amount of data from the whole training set during computation. Thus this helps to avoid complexity and time as well as compared to the other schemes.

The algorithm for stochastic gradient descent (SGD) is:

Algorithm 2:

- Arrange the data in random order.
- Calculate expectation of the objective through calculation of gradient and complexity across the whole training set.
- Choose proper learning rate ɑ such that after each epoch t, change ɑ as $\frac{\alpha_0}{\alpha_1+t}$ where α_0 is the initial learning rate and α_1 is when the change in learning rate begins.
- Update Θ value where Θ is given by :

$$\Theta = \Theta - \alpha\partial_\theta E[J(\Theta)] \quad (17)$$

where $E[J(\Theta)$ is the expectation of the objective $J(\Theta)$.

While updating Θ, use only few training data, thereby modifying the formula to:

$$\Theta = \Theta - \alpha\partial_\theta J(\Theta; x_i, y_i) \quad (18)$$

where (x_i, y_i) are the data from training.

SUPPORT VECTOR MACHINE

A support vector machine is a deep learning network that is used for classification purposes and sometimes for regression.

The basic principle behind SVM is that each vector points are plotted in an n-dimensional area are divided into classes uniquely by a hyper-plane such that the divided classes consists of only one particular vector type, which represents the features of the data.

SVM has been used for pattern recognition because of its good performance and high efficiency.

The aim behind SVM is to map the vector points from the input data space to a high dimensional feature map such that features can be categorized into different classes very easily. Thus it primarily involves classifying classes such that the hyperplane is at a large significant distance between the two classes.

Let us consider a training data of
$\{x_i, y_i\}$ where i=1,2,3…...N.

Here x_i are the inputs and y_i are the respective targets such that $y \epsilon \{-1, +1\}$.

The training data is mapped to the target data by a kernel function $\rho(.)$.

Now, the hyperplane is defined as

$$v^K\rho(x) + s = 0 \quad (19)$$

where v is an unknown vector (weight) with has the exact same dimensions as $\rho(x)$ and s belongs to the space (bias).

The input vector is matched to a particular class if and only if $v^K\rho(x) + s$ = +1 or -1; which means it belongs to one class if $v^K\rho(x) + s = +1$ and the other class if

$$v^K\rho(x) + s = -1 \quad (20)$$

Thus $v^K\rho(x) + s$ is known as the SVM classifier.

EXPERIMENTS AND RESULTS

In this section, we will elaborate on the datasets used in the proposed algorithm along with the results of the said algorithm

Dataset

We use one type of dataset for our experiment purposes-Poly-U cross spectral database. We use this database as it has both near infrared (NIR) and visible (VIS) iris images and so we can perform cross domain matching.

This database has 209 classes each, with a total of 12,540 Iris images, with each class having 60 images, out of which 30 images belong to near-infrared and the other 30 belong to visible domains respectively.

Each image belonging to 209 different subjects in the database is of the dimensions 640 X 480 pixels, both in the NIR domain as well as in the VIS domain.

Softwares used

We used two softwares for recognition and matching purposes- MATLAB & Python.

For iris recognition and template generation, MATLAB was used, and for matching purposes using our classifier, Python was used as the latter works well for large data-sets with less computational time.

Iris recognition

All the images in the database were used in the process of Iris recognition. segmentation, normalisation and encoding were done in the respective order to obtain templates for each image, having dimensions of 20 x 480 pixels each. (See Figure-3). This was done properly for a total of 209 different subjects (8360 images) comprising of the subjects in both near infrared and visible domains.

Cross domain matching

Matching between the near-infrared and visible domains are done using the proposed algorithm. Our proposed algorithm for matching is given below:

Algorithm 3

Training

- Binary templates of each of the images in their respective domains are stored column-wise having dimensions of 1 X 9600 pixels each.

- The near-infrared templates are fed into our multilayer perceptron (MLP) Classifier as inputs and the visible domain templates are taken as targets for regression.
- Simultaneously the same visible domain templates are passed as inputs to our support vector machine (SVM) classifier so as to segregate them into distinct classes (1-209).

Testing

- The output of our MLP classifier is passed as inputs to our SVM classifier for testing.
- If matching occurs, output 1 else output 0 (output A).
- If no matching occurs, (output 0), then input those particular vectors into our second MLP classifier which is trained for classification.
- If matching occurs, output 1, else output 0 (output B).
- Replace the output 0 values in output A with their respective values obtained in output B.
- Calculate performance (error-rate) and hence, accuracy of the proposed algorithm and plot the ROC curve.

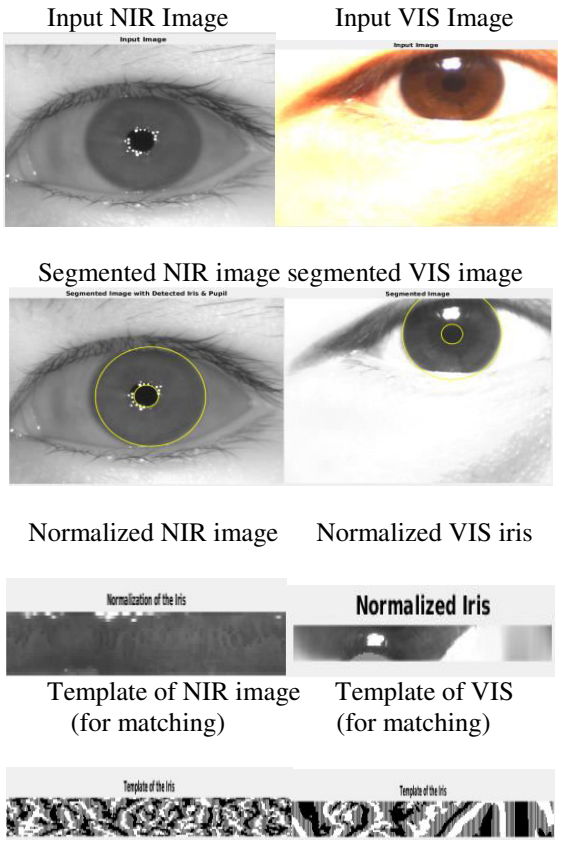

Figure-3. Iris recognition for NIR and VIS images.

The performance of the proposed algorithm is measured through the receiver operating characteristic (ROC) curve and the equal error rate.

As we can see from the ROC curves (See figures 4 and 5) which gives us a plot between the false acceptance rate and true acceptance rate, the performance of the NIR-NIR matching is the highest, and the EER of it is 4.2 %, then comes VIS-VIS matching, with an EER of 5.4% and the least is NIR-VIS matching, with an EER of 28.75%.

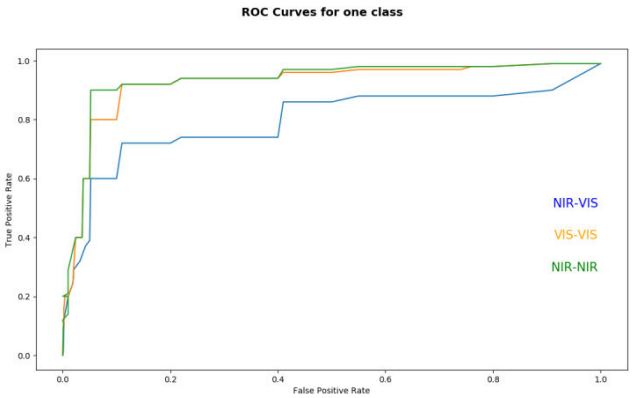

Figure-4. ROC curve for one class for same domain and cross domain matching.

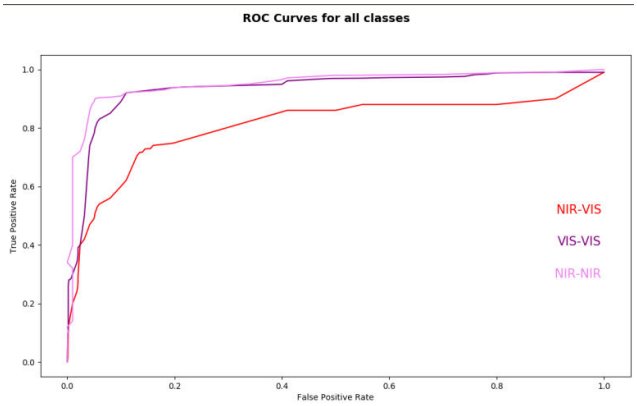

Figure-5. ROC curve for all classes for same domain and cross domain matching.

The accuracy of NIR-VIS is higher when compared to NIR-VIS matching using MRF (Markov Random Fields) Model [1] as well as the matching using an adaptive DA-NBNN algorithm [1], as we can infer from Table-1.

As we can observe, there is a significant improvement of accuracy of our proposed algorithm when compared to the other two schemes, with an accuracy of 71.25%.

Therefore, these results clearly show that there is a clear higher performance of our proposed algorithm for cross domain matching.

Table-1. Cross-domain iris recognition performance.

Iris matching domains	Accuracy of various algorithms		
	DA-NBNN model [1]	MRF model + DA-NBNN model	Our proposed algorithm
NIR-NIR	97.82%	-	95.8%
VIS-VIS	94.79%	-	94.6%
NIR-VIS	58.8%	61.9 %	71.25%

CONCLUSIONS

This paper has investigated cross-domain iris recognition problem and presented a new approach to accurately match iris images acquired under different domains. We developed a MLP based classification framework for cross-domain iris matching. The effectiveness of the proposed approach was evaluated for cross-spectral iris matching using the POLY-U database. The experimental results presented in section VII show that the proposed adaptation is proficient of benefiting from the usage of Python (which significantly increased the speed). However, neural network proves to cause an increase in efficiency that is substantial enough to practically apply this method. Therefore, it is reasonable to conclude that usage of this proposed algorithm will work well in a scenario where an image from a camera (VIS) needs to be matched with that of a pre defined dataset (NIR).

ACKNOWLEDGEMENTS

We would like to thank The Hong Kong Polytechnic University for providing us with their database.

REFERENCES

[1] N. Pattabhi Ramaiah, Ajay Kumar. 2017. Towards More Accurate Iris Recognition using Cross-Spectral Matching. IEEE Trans. on Image Processing. 26(1): 208-221.

[2] Parekh R., Yang J., Honavar V. 2000. Constructive Neural Networks for pattern matching. IEEE Trans. Neural Network. 11(2): 436-451.

[3] Z. Zhao and A. Kumar. 2015. An accurate iris segmentation framework under relaxed imaging constraints using total variation model. Proc. ICCV 2015, pp. 3828-3836, Santiago, Dec.

[4] L. Masek and P. Kovesi, Matlab source code for a biometric identification system based on iris patterns, 2003, http://www.csse.uwa.edu.au/~pk/studentprojects/libor/index.html.

[5] S Chaplot, L.M. Patnaik, N.R. Jagannathan. 2006. Classification of magnetic resonance brain images using wavelets as input to support vector machine and neural network. Biomedical signal processing and Control. 1(1): 86-92.

[6] Jiexiong Tang, Chenwei Deng, Guang-Bin Huang. 2016. Extreme learning machine for multilayer Perceptron. IEEE Trans. on Neural Network and Learning Systems. 27(4): 809-821.

[7] L. L. C. Kasun, H. Zhou, G.-B. Huang and C. M. Vong. 2013. Representational learning with extreme learning machine for big data. IEEE Intell. Syst. 28(6): 31-34.

[8] C. Cortes and V. Vapnik. 1995. Support-vector networks. Machine Learning. 20(3): 273-297.

[9] P. Phillips, A. Martin, C. Wilson, and M. Przybocki. 2000. An Introduction Evaluating Biometric Systems. Computer. 33(2): 56-63.

[10] C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), first ed. Springer, 2007.

[11] Walter H. Delashmit, Michael T. Manry. 2005. Recent Developments in Multilayer Perceptron Neural Networks. Proceedings of the 7th Annual Memphis Area Engineering and Science Conference, MAESC.

[12] H. Proenca and L.A. Alexandre. 2005. Ubiris: A noisy iris image database. in Image Analysis and Processing. pp. 970-977.

[13] B. Kulis, M. Sustik and I. Dhillon. 2006. Learning low-rank kernl matrices. in Proc. ACMInt.lConf Machine Learning. pp. 505-512.

[14] S.T.Q.C Directorate. Biometric Devices testing and certification. http://www.stqc.gov.in/content/about-certification/.