[UCI]I804:24011-200000267440

August 2019

Doctor's Degree Thesis

Deep Network Design based on Independent Component Analysis and Application of User Recognition

Graduate School of Chosun University

Dept. of Control and Instrumentation Engineering

Jae-Neung Lee

Deep Network Design based on Independent Component Analysis and Application of User Recognition

독립성분분석기반 딥 네트워크 설계 및 사용자 인식의 응용

August 23, 2019

Graduate School of Chosun University

Dept. of Control and Instrumentation Engineering

Jae-Neung Lee

Deep Network Design based on Independent Component Analysis and Application of User Recognition

Advisor: Prof. Keun-Chang Kwak

This thesis is submitted to The Graduate School of Chosun University in partial fulfillment of the requirements for the Doctor's degree in Control and Instrumentation Engineering

April 2019

Graduate School of Chosun University

Dept. of Control and Instrumentation Engineering

Jae-Neung Lee

This is to certify that the Doctor's Thesis of Jae-neung Lee

has been approved by the Examining Committee for the thesis requirement for the Doctors's Degree in Control and Instrumentation Engineering

Committee Chairperson Prof. Sung Bum Pan	[illegible] (Sign)
Committee Member Prof. Hyun Sik Choi	Hyun-Sik Choi (Sign)
Committee Member Prof. Hong Gi Yeom	[illegible] (Sign)
Committee Member Prof. Do Hyung Kim	[illegible] (Sign)
Committee Member Prof. Keun Chang Kwak	[illegible] (Sign)

June 2019

Graduate School of Chosun University

Table of Contents

List of Figures

List of Tables

요약

독립성분분석기반 딥 네트워크 설계 및 사용자 인식의 응용

이재능
지도교수 : 곽근창 교수, Ph. D.
조선대학교 대학원 제어계측공학과

본 논문에서는 독립성분분석기반 딥 네트워크인 ICANet1의 체계적인 구조를 제안하고, ICANet2 모델로 발전시킨다. 기존에 제안된 PCANet은 deep learning의 비해 계산량이 작고, 속도가 빠르다. 하지만 데이터의 변형에 따라 성능이 낮아지는 단점이 있다. 문제점을 해결하기 위해 ICANet알고리즘을 제안한다. ICANet알고리즘에는 2가지 구조가 있다. ICANet1 알고리즘의 첫 번째 단계는 평균제거단계이고, 중심영상을 구한다. 그 다음은 중심영상을 이용하여 공분산을 구하고, 그 값을 통해 고유벡터를 얻고, 마지막으로 고유벡터는 독립성분분석 필터의 입력으로 사용된다. 여기서 센터링 (centering) 과 화이트닝 (whitening) 단계가 실행되고, 가중치 업데이트를 통해 최적 독립성분 행렬이 얻어진다. 이 행렬을 이용하여 평균제거단계를 통해 얻어진 패딩영상과 합성곱이 계산된다. 이것이 바로 ICANet2의 첫 번째 단계이다. ICANet2의 경우에는 고유벡터를 사용하지 않고, PCA의 특징벡터를 이용한다. PCA 특징벡터의 상관관계를 제거하기 위해, ICA알고리즘을 이용하여 통계적 독립인 벡터를 찾는다. 통계적 독립 벡터를 이용함으로써 PCA의 단점을 개선 할 수 있고, 이를 통해 성능을 개선할 수 있다. ICANet의 2번째 단계는 1단계와 거의 동일하며, 1단계에서 얻은 데이터를 사용하기 때문에 차원의 수만 감소된다. 2단계에서 얻은 합성곱 데이터를 이용하여 출력 단계의 입력으로 사용된다. 여기서 해싱코드 단계와 히스토그램을 이용해서 독립성분분석 네트워크 특징을 얻

을 수 있다. 최종 히스토그램에서 얻어진 특징 값은 벡터 화하여 분류기의 입력으로 사용된다. ICANet을 사용함으로써, 딥러닝 보다 구조가 얕기 때문에 성능은 떨어 질 수 있으나, 인식속도에서 월등한 면을 보여주고, 특히 모바일에서도 사용이 가능 할 것으로 기대된다. 분류기는 유클리드 거리, 제곱유클리드 거리, 도시블록 거리, 민코프스키 거리, 코사인거리, 상관관계 거리 그리고 스피어만 거리, SVM를 이용하여 성능을 나타낸다. 제안된 방법의 성능을 검증하기 위해 FERET 얼굴데이터, CU-FACE 데이터, CU-ECG 데이터, 노이즈 ECG 데이터베이스를 사용하였다. 실험결과 시간에 영향을 받은 Dup1, Dup2데이터의 경우 ICANet1, 2가 PCANet보다 좋은 성능을 가져왔다. 또한 잡음 심전도 데이터를 사용했을 때 PCANet보다 향상된 성능을 나타냈다. 이로써 ICANet알고리즘의 유효성을 증명하였다.

Ⅰ. Introduction

1.1 Research background and review of papers

Deep learning is defined as a set of machine learning algorithms that are dependent on the derived neural network of machine learning and attempt a high level of abstraction (a task that summarizes key content or functions in large amounts of data or complex data) through a combination of several nonlinear transformation techniques. It can also be said that it is a field of machine learning that teaches computers how people think in a big frame. When there is any data, it is expressed in the form that the computer understands (for example, the pixel information is represented by a column vector in the case of images). As a result of these efforts, various deep learning techniques such as deep neural networks, convolutional neural networks, and deep belief networks are used for computer vision, speech recognition, natural language processing, It is applied to the field of voice/signal processing and shows cutting-edge results [1-3]. However, there is a limit to deep learning. For example, the problem of time and cost for collecting big data, the efficiency problem that must be learned through many repetitions, the theoretical verification of algorithm except slope descent is poor, and the black box can be listed [4-6]. In other words, a convolutional neural networks (CNN) is a basic deep learning architecture that has been applied to text, image, and speech recognition. In many industries, deep learning interest is constant. In addition, the most important thing is that CNN can automatically extract and learn hidden representations of data on a number of blocks consisting of a convolutional layer, activation function layer, and max-pooling layer. However, how to choose parameters and configurations, including the filter sizes, the number of layers, and the pooling function, is a big challenge. Consequently, the network structure such as AlexNet, ResNet and GoogLeNet are growing deeper. Despite the great successes of deep learning, common researchers are not yet solving its feature learning mechanism and optimal network configuration problem. Accordingly, Chen studied the principle component analysis network

(PCANet), which is a shallow and intensive deep learning network. Also, a simple deep learning network, which have to be very easy, even trivial, to train and to adapt to different database and tasks. Consequently, such a basic network could serve as a good reference for researchers to justify the use of more advanced processing components or more sophisticated architectures for their deep learning networks empirically. Therefore, The PCANet algorithm has been proposed by Chan. However, PCANet is still correlated, and the ICANet algorithm has been proposed to remove the correlation. Although the ICANet algorithm has appeared recently, it has not been systematically generalized. Therefore, this paper aims to generalize algorithms from ICANet1 ~ 2 and extend the concept. In order to improve the limitations of PCA in the face recognition field, various studies on ICA have been conducted and we propose deep ICANet1, 2 algorithm that improves the disadvantages of PCANet and generalized to improve performance using practical data. In addition, we validate the ICANet algorithm by comparing the performance with the existing PCANet algorithm and analyzing the influence of the parameters. Lastly, the ECG data is applied for authentication security, and the face data is used for comparing the performance of the algorithm.

The contribution of this paper is as follows. First, we generalized the ICANet1 algorithm. The proposed ICANet1 algorithm is poorly explained theoretically and structurally. Therefore, this paper improves the theoretical lack of ICANet1's algorithm. Second, we modeled a new structure called ICANet2. Third, the validity of the algorithm was verified by using the electrocardiogram data and the image data obtained in real life. To study ICANet algorithm, we list the trend of several algorithms as follows.

We first list the trends of PCA, which is the origin of the PCANet algorithm. The PCA was proposed by Jolliffe in 2002 [7]. First, the derived algorithms of PCA are listed in order of domestic and foreign. Chungnam National University Bui [8] studied in-video human detection and pose estimation using motion data based on RPCA algorithm. Chosun University [9] performed personal identification using the electrocardiogram signal data based on MPCA algorithm. The existing one dimensional electrocardiogram signal was changed into three-dimensional shape to improve the recognition rate. Seoul National University Park [10] classifies images using kernel principal component analysis with image database class. Kim [11] of Suwon University used numerical incremental principal component analysis based on handwritten numerical data.

The following are overseas trends using RPCA. Jin [12] detected contrast-filled vessels using X-ray data based on robust principal component analysis (RPCA). Lu [13] was classified using face image and object data based on RPCA. We use Li [14] image data based on RPCA to detect moving objects. Vaswani [15] performed subspace tracking and restoration using RPCA based on image data. Liu [16] performed tumor classification using RPCA based on MRI data. Sun [17] used geo-spatial classification using Hyper-spectral data based on RPCA. Zhong [18] classified faces and numbers using robust 2DLDA based on face and numerical data. Yan [19] face and numerical data based on RPCA.

The following are trends using KPCA. Kuang [20] performed intrusion detection using KPCA based on communication data. Liu [21] performed mental fatigue estimation using KPCA-HMM based on EEG data. Vinay [22] performed face recognition using face data based on KPCA. Quan [23] performed facial recognition using face data based on KPCA. Romero [24] classified the images using KPCA based on satellite image data. Xia [25] performed image classification using KPCA based on hyper-spectral image data. Yuan [26] studied smoke detection using image data based on KPCA. Hotta [27] performed a scene classification using KPCA based on image data. Xiao [28] performed novelty detection using KPCA based on image data of square, spiral, and banana motifs.

The following are research trends of various SPCA. Chaib [29] performed a scene classification using sparse PCA based on image data. Ashutosh [30] classifies objects using standardized PCA based on hyper-spectral image data. Uddin [31] classifies images using segmented PCA based on hyper-spectral image data. Xiao [32] performed face recognition using face data based on sparse PCA. Hu [33] performed leukemia analysis using sparse PCA based on medical statistical data. Pierrefeu [34] analyzed mild cognitive impairments using structured sparse PCA based on MRI medical data. Wang [35] performed image classification using sparse PCA based on hyper-spectral image data. Chail [36] performed VHR scene classification using hyper-spectral image data based on sparse PCA. Li [37] analyzed and classified images using a supervised version of PCA based on hyper-spectral image data. Liu [38] performed facial recognition using face data based on symmetry PCA. Xi [39] uses numeric image data based on separable PCA for number classification.

The following are research trends for various IPCA. Dagher [40] studied face recognition using incremental PCA based on face data. Rozza [41] studied spam filtering using isotropic PCA based on alphabetical data obtained from e-mail. Yuan [42] performed scene separation for visual monitors using incremental PCA based on scene data. Qu [43] performed object recognition using incremental PCA based on object data. Qing [44] performed face detection using incremental PCA based on scene data. Wang [45] diagnosed the circuit breaker using Improved PCA based on feature data. Yin [46] performed visual tracking using incremental PCA based on 3D image data. Alorf [47] performed face detection and recognition using the improved PCA based on face data. The following lists the algorithms derived from LDA. Bengherabi [48] performed facial recognition using regularized LDA based on facial data. Yuan [49] used terrain classification using spectral-spatial LDA based on hyper-spectral data. He [50] classified the terrain using a nonlinear compressed sensing-based LDA topic based on polarimetric synthetic aperture radar image data. Zeng [51] classifies objects using a multi-linear discriminant analysis network based on object data. The following lists ICA algorithms and algorithms derived from ICA. First, Hong Seongjun [52] in Korea performed facial expression recognition using face data based on ICA. Hwang [53] studied mode separation of buildings using ICA based on sound data. Baek [54] classifies images using ICA based on image data. Kwon [55] used image data based on ICA to classify the red pepper images. Kang [56] studied the translational-warping mode of dry matter using ICA based on sound data. Kim [57] analyzed the vibration source contribution of plate structures using vibration signal data ICA. Kim [58] studied the identification of vibration source signals of structures using ICA based on sound data. Jeon [59] studied frequency binarization using frequency domain ICA based on sound data. Park [60] studied blind signal separation using

ICA based on voice data. Hwang [61] studied a new mode separation technique using the acceleration response of structures using sound data based on ICA. Lee [62] designed a dynamic noise reduction filter using ICA. Jung [63] designed a partial discharge pattern classifier using ICA based on high - dimensional data.

Overseas, Wu [64] performed gender recognition using multi-scale ICA based on image data. Sun [65] performed tumor classification using microarray data based on ICA. Martis [66] performed ECG beat classification using the ECG data based on ICA. Wang [67] performed video event classification using ICA based on video data. Yu [68] performed image classification based on image data and ICA. Falco [69] performed hyper-spectral image classification using the effectiveness of different ICA based on image data. Stewart [70] performed a single-trial classification using ICA based on EEG data. Xiao [71] studied sparse representation using image reconstruction ICA. Pruim [72] studied the removal of motion artifacts using robust ICA based on MRI data. Ruhi [73] studied eye blink artifact denoising using wavelet ICA based on EEG data. Coloigner [74] performed cancer classification using semi-nonnegative ICA based on image data. Tao [75] performed SAR data classification using Tensorial ICA based on image data. Griffanati [76] performed hand classification using fMRI data based on ICA. Chai [77] performed driver fatigue classification using EEG data based on entropy rate bound minimization analysis ICA. Sai [78] performed an automated classification using wavelet-ICA based on EEG data. Zhang [79] performed facial recognition using ICANet based on image data. Lv [80] recognized saccadic ECG signals using ICA based on EOG signal data.

The following is a research trend about deep learning. Because ICANet is part of deep learning, a comparative analysis is needed. Ruslan [81] classified images and motion using a hierarchical dirichlet process deep Boltzmann machine based on motion data and image data. Chen [82] performed hyper-spectral data classification using a stacked auto-encoder based on image data. Dobhal [83] performed human activity recognition using binary image data based on CNN. Hasan [84] performed activity recognition using sparse auto-encoder based on video image data. Du [85] performed action recognition using hierarchically bidirectional RNN based on skeleton data.

Neverova [86] performed human identification using dense clockwork RNN based on image data. Wang [87] performed action recognition using three channel deep convolutional neural networks based on depth map image data. Koohzadi [88] performed human action recognition using a CNN structure based on video data. Kim [89] performed human activity classification using deep CNN based on motion data. Liu [90] performed action recognition using two CNN based on video data. Daniel [91] transformed the structure of CNN based on image data to generate motion data and picture data. Chen [92] performed action recognition using the structure of CNN based on image data. Wu [93] performed human identification using CNN structure based on image data. Ni [94] performed an action representation using a CNN structure based on video data. Rahmani [95] performed human action recognition using robust non-linear knowledge transfer model based on motion capture data. Wang [96] performed human action recognition using a CNN structure based on image data. Finally, there are studies on noise using ICA. Yang [97] proposed a new framework for measuring sternal cardio-mechanical signals from moving subjects using multiple sensor. Interference detection and removal method with minimal distortion of the EMG was developed based on the independent component analysis by Zheng [98] ICA is more resistant to noise than PCA. In this paper, we demonstrate the superiority of ICANet by adding noise to the data.

1.2 Composition of paper

This study is composed as follows. Section 1 describes the motivation for ICANet's algorithm development and lists the derived algorithms of the PCANet algorithm that overcomes the shortcomings of deep learning. It shows the emergence background of PCANet and ICANet. In Section 2, we show the research trends of PCA, LDA, ICA and LBP algorithms, and show the proposed methods of PCANet, LDANet, and ICANet algorithms. Section 3 presents ICANet1 and ICANet2. In Section 4, we prove the validity of the proposed algorithm using face and ECG database and compare the effect of each parameter. Finally, Section 5 present concludes.

Ⅱ. PCANet and its variances with ICA

2.1 PCA

PCA is a method of identifying patterns in data and expressing the data in such a method as to highlight their similarities and differences. Since patterns in data could be hard to find in data of high dimension, where the graphical representation is not available, PCA is a useful tool for analyzing data.

One of the main advantage in PCA is that once we have found these patterns in the data, and we compress the data, for example by reducing the number of dimensions, without a lot of loss of information.

The PCA is a set of mutually orthogonal basis vectors indicating the direction of maximum dispersion of data. PCA is widely used for image data. The biggest problem in recognition problem using image data is the high level of data. Dimensioning the high dimension can lead to efficient features. It is a method of expressing an arbitrary image by orthogonal basis image that expresses the overall characteristics of a learning image using statistical characteristics of a learning image, that is, decomposing into an Eigen image and linear combination of the inherent image. PCA was applied to face recognition by Pentland in 1991.

```
Algorithm 1 PCA
Input : a D-dimensional training set $X = x_1, x_2, ..., x_N$ and
        the new (lower) dimensionality d (with d≤D)
1. Computing the mean $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x^i$
2. Computing the covariance matrix $Cov(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x}_i)(x_i - \bar{x}_i)^T$
3. Finding the spectral decomposition of Cov (x), obtaining the
   Eigenvectors $V_1, V_2, ..., V_D$ and their corresponding Eigenvalues
   $\lambda_1, \lambda_2, ..., \lambda_D$. Noting that the Eigenvalues are sorted, such that
   $\lambda_1 \geq \lambda_2 \geq \bullet \bullet \bullet \geq \lambda_D \geq 0$
4. For any $x \in R^D$, its new lower dimensional representation is:
   $y = (\lambda_1^T(x-\bar{x}), \lambda_2^T(x-\bar{x}), ..., \lambda_d^T(x-\bar{x}))^T \in R^d$ and the original x can be
   approximated as
   $x \approx \bar{x} + (\lambda_1^T(x-\bar{x}))\lambda_1 + (\lambda_2^T(x-\bar{x})\lambda_2 + \bullet \bullet \bullet + \lambda_d^T(x-\bar{x})\lambda_d$
```

2.2 ICA

In this section, we present the ICA algorithm used in the setting of recognition problem and describe the procedure as it develops for several different architectures.

Firstly, we address the motivation of ICA. Suppose that they are in a room where two people are speaking at the same time. The room has two microphones at different places, and we can get two voice signals which we can denote by $x_1(t)$ and $x_2(t)$, with x_1 and x_2 the amplitudes, and t the time index. The signal recorded by the microphone is the weighted sum of the speech signals emitted by the two speakers, which we denote by $s_1(t)$ and $s_2(t)$. We can express this as a linear equation:

$$x_1(t) = a_{11}s_1 + a_{12}s_2 \tag{1}$$

$$x_2(t) = a_{21}s_1 + a_{22}s_2 \tag{2}$$

where a_{11}, a_{12}, a_{21}, and a_{22} are parameters that depend on the distances of the microphones from the speakers. It would be very useful if we could now estimate the two original speech signals $s_1(t)$ and $s_2(t)$, using only the recorded signals $x_1(t)$ and $x_2(t)$. In shortly, the motivation for developing ICA is similar to the cocktail-party problem (CPP). In other words, it is a method to independently select only the signal that the developer desires in a lot of noise. Therefore, there is an advantage of being robust against noise.

Secondly, we indicate the definition of ICA with an algorithm. To define ICA, we could use a statistical "latent variables" model. Supposed that we observe n linear mixtures, $x_1,\ldots,$ x_n of n independent components.

$$x_j = a_{j1}s_1 + aj_2s_2 + \ldots + a_{jn,}s_n,\ \mathrm{for}\, all\, j \tag{3}$$

We have originally defined the time index t in the ICA algorithm, suppose that each mixture, x_j as well as each independent component, s_k is a random variable, instead of a suitable time signal. The observed data $x_j(t)$, the microphone signals in the CPP, are then a sample of this random variable. Without loss of generality, we could suppose that both the mixture variables and the independent components have zero mean. If this is not true, then observable variables x_i could always be centered by subtracting the sample mean, which makes the model zero mean.

It is useful to use vector-matrix notation instead of the sums. Let x be the random vector with $x_1,\ldots,x_n$ elements; likewise s is the random vector of $s_1,\ldots,s_n$ elements. Let A be the matrix with elements a_{ij}. Generally, bold lowercase letters indicate vectors, and bold upper-case letters denote matrices. All vectors are understood as column vectors; thus x^T, or the transpose of x, is a row vector. Using this vector-matrix notation, the above mixing model is written as

$$x = As \tag{4}$$

Sometimes we need the columns of matrix A; denoting them by a j the model can be also written as

$$x = \sum_{i=1}^{n} a_i s_i \tag{5}$$

The statistical model in equation (4) is called independent component analysis, or ICA model. The ICA model is a generative model, which means it describes how the observed data are generated in the process of mixing s_i components. The independent components are latent variables that mean they cannot be observed directly. In addition, the mixing matrix is supposed to be unknown. All we observe is the random vector x, and we have to estimate both A and s with it. This should be done under as general assumptions as possible. The independent components are latent variables, meaning that they cannot be directly observed. Also, the mixing matrix is assumed to be unknown. All we observe is the random vector x, and we must estimate both A and s using it. This must be done under as general assumptions as possible.

The starting point for ICA is the very simple assumption that the components s_i are statistically independent. Statistical independence will be rigorously defined in Section 3. It will be shown below that, we must also assume that the independent component must have non-Gaussian distribution. However, in the basic model, we do not assume these distributions to be known (if they are, the problem can be considerably simplified.) For simplicity, we also assume that the unknown mixing matrix is square. This assumption, however, can be sometimes withdrawn, as will be shown in Section 4.5. Then, after estimating the matrix A, we can compute its inverse, set W, and obtain the independent component simply by:

$$s = A^{-1}X = Wx \tag{6}$$

As demonstrated earlier, the output $Y = [y_1, y_2, \ldots, y_n]^T$ is obtained by mapping s through an invertible squashing function “f” that is

$$Y = f(s) \tag{7}$$

The obtained matrices $s=[s_1,s_2,\dots,s_n]^T$ and $Y=[y_1,y_2,\dots,y_n]^T$ could be interpreted as pre-synaptic and postsynaptic activation of neurons, respectively. This is achieved by computing gradient ascent on the entropy of the output taken with respect to W. We define the mutual information M between "n" , random variables, $y_i, i=1,2,\dots,n$ as follows:

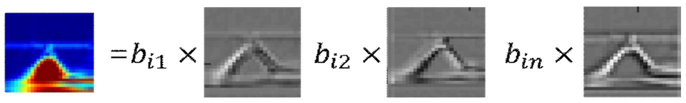
$$M(y_{1,}y_{2,\dots,}y_n) = \sum_{i=1}^{n} H(y_i) - H(y_1,y_2,\dots,y_n) \tag{8}$$

where each $H(y_i)$ denotes a marginal entropy. The mutual entropy M is always non-negative. Finally, the gradient update rule for the weight matrix W assumes the standard form.

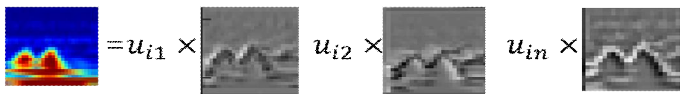
$$\triangle W \propto \nabla w H(Y)W^T W = (I+Y' s^T)W \tag{9}$$

where I is the identity matrix. Given the form of the logistic transfer function, we get $Y' = (1-2Y)$.

On the other hand, the objective of the first architecture model is to find a set of statistically independent basis images from Eigenvectors obtained with the PCA method. Figure 2.1 shows the ECG representation of ith face image obtained for this model.



(a)



(b)

Figure. 2.1. ECG representation. (a) ECG representation for the first architecture (ICA1). (b) ECG representation for the second architecture (ICA2).

The face-image representation for this architecture comes as a linear combination of statistically independent basis images and a feature vector $b_i=[b_{i1},b_{i2},\ldots,b_{ir})$ for ith face image. The feature vectors for training the image set are obtained by the product of features found with PCA method and the inverse of weight matrix being the results of the ICA. The purpose of the second architecture model is to find a factorial face code. Here, the face-image representation is represented by a linear combination of the statistically independent feature vector $u_i=[u_{i1},u_{i2},\ldots,u_{ir})$ and the basis image. As an example, figure 2.1 illustrates the face representation of the ith face image for the second model. Here, the factorial codes for face images are obtained by taking a product of a zero-mean matrix of the training image set and the weight matrix produced from the ICA.

In short, ICA is related to the method called blind source separation (BSS). The "source" indicates here the original signal, i.e. the independent component, like the speaker in the cocktail party problem. "Blind" means we know very little, if anything, about the mixing matrix, and make little assumptions on the source signals. ICA is the probably the most widely used method for BSS. Meanwhile, FastICA offers most of the advantages of neural algorithms: It is parallel, distributed, computationally simple, and requires little memory. Stochastic gradient methods seem to be preferable only when fast adaptivity in changing environment is required.

Another advantage of the FastICA is that unlike ordinary ML algorithms, which only work for a given class of distributions, it can estimate both sub- and super-Gaussian independent components. Also, the FastICA algorithm is the most widely used method for blind source separation problems, it is computationally efficient and requires less memory than other blind source separation algorithms. The other advantage is that independent components can be estimated one by one that further decreases computational load.

2.3 Deep Learning

Deep Learning (DL) is a subset of machine learning which is getting a lot of attention for its recent success in solving particularly hard large-scale problems in such areas as image classification, signal processing, natural language processing, and motion classification. DL is a refinement of the Artificial Neural Networks, which, as discussed earlier, “roughly” emulate how the human brain learns and solves problems.

Before we dive into how deep learning works, it’s important to understand how Artificial Neural Networks (ANN) work. An ANN is made up of interconnected group of neurons, like the neuron network in the human brain. In a simple way, a neuron in a neural network is a unit that receives a number of inputs (x_i), computes them, and sends the output to the other nodes or neurons in a simple deep learning network. Weights (w_j), or parameters, indicates the strength of the input connection and can be either positive or negative. The inputs are multiplied by the associated weights ($x_1 w_1$, $x_2 w_2, \ldots, x_i w_i$) and the neuron adds the output from all inputs. At the last step, the neuron performs a computation or activation function. The activation function (sigmoid function and ReLU are popular) allows the ANN to model complex nonlinear patterns. Figure 2.2 shows an example application of deep learning.

Figure 2.2. Applications of deep learning

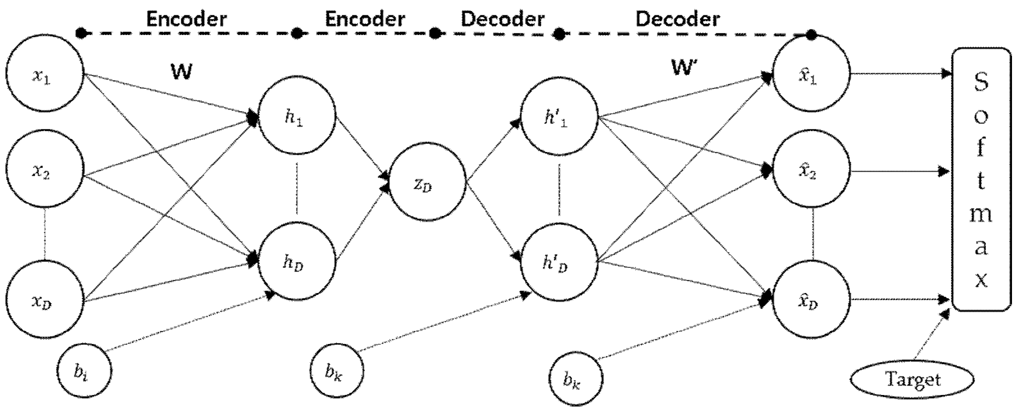


Figure 2.3. Simple deep learning architecture called stacked auto-encoder

Figure 2.3 represents a simple deep learning architecture called stacked auto-encoder. The first layer is the input layer, and this is where features $(x_{1,} x_{2,} x_3)$ are inputted. The second layer is the hidden layer. Any layer that is not an input or an output layer is a hidden layer. The term DL basically means that there are multiple levels of hidden layers. Networks typically contain more than three hidden layers, but in some cases, they can feature more than 1,500 hidden layers. Hidden layers are essentially a hierarchical grouping of different variables that provide a better-defined relationship. Currently, most DL algorithms are supervised. Thus, DL models are trained against the known truth.

The benefit of multiple hidden layers is that when a pattern needs deeper investigation, this can be done with additional hidden layers. Image classification is an example where DL can achieve high performance on very hard visual recognition tasks (even exceeding human performance in certain areas). We illustrate this point with an example of how additional hidden layers help perform facial recognition. Figure 2.4 shows image recognition using deep learning.

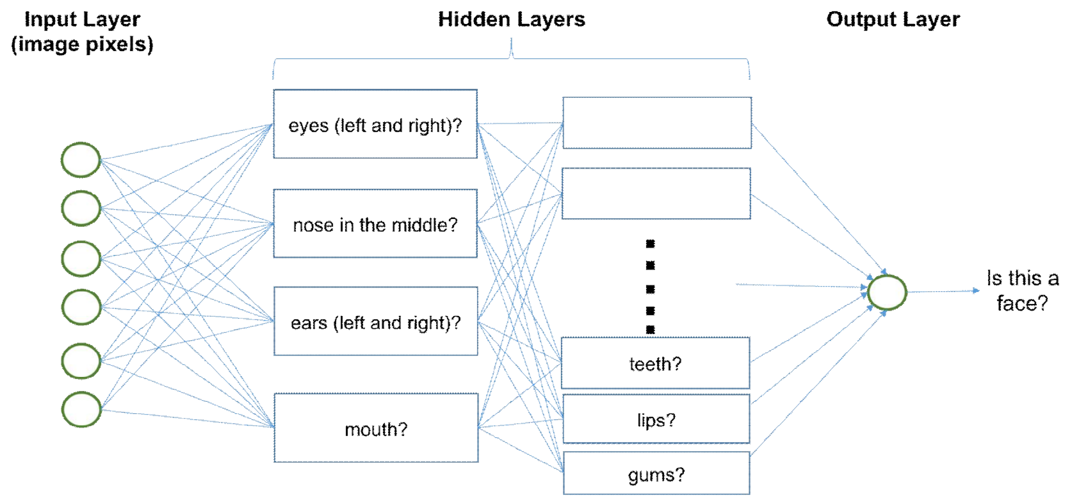


Figure 2.4. Image Recognition using DL

When a picture is inputted into a DL network, it is first decomposed into image pixels. The algorithm would look for patterns of shapes at certain locations in the data. The first hidden layer tries to uncover specific facial patterns: eyes, mouth, nose, ears. Adding an additional hidden layer decomposes the discovered facial patterns into more granular attributes. For example, the "mouth" can be further deconstructed into "teeth", "lips", "gums", etc. Additional hidden layers can further evolve these patterns to recognize subtlest nuances. In the end, a deep learning network can break down a very complicated problem into a set of simple problems. Deep learning also features a good performance in many areas. Below we outline the disadvantages and advantages of the deep learning algorithms.

【Advantages】

- DL significantly outperforms other solutions in multiple domains, including speech recognition, language and image processing, etc.
- With DL there is no need for feature engineering, which one of the most time-consuming steps in machine learning applications.
- DL can be relatively easy adapted to new problems like computer vision, time series processing, etc. where convolutional neural

networks, recurrent neural networks, long short-term memory, and the like techniques are increasingly used.

【Disadvantages】

- DL requires a large amount of data - if we only have thousands of sample, DL is not likely to outperform other algorithms.
- From computational perspective DL is extremely expensive to train; complex models may take weeks to train using hundreds of machines equipped with expensive GPUs.
- DL does not have much in the way of strong theoretical foundation, which further leads to the next disadvantage.
- Determining topology/flavor/training method/hyper parameters for a deep learning algorithm is rather an art than a science.
- What is learned is not always easy to comprehend. Other classifiers (such as decision trees, logistic regression, etc.) yield the results, which are much easier to understand.

2.4 PCANet

In this section, we first review the PCANet [99][100]. Its architecture is shown in Figure 2.5 and can be divided into three stages that include 10 steps. Figure 2.6 shows the PCANet workflow of extracting the features from the train dataset. Table 2.1 shows the core parameters of the PCANet.

Table 2.1 The core parameters of PCANet

Parameters	definition
$m \times n$	The size of input image
N	The number of data
$k_1 \times k_2$	The patch size. k_1 are k_2 odd integers and satisfy $1 \le k_1 \le m, \quad 1 \le k_2 \le n$
$L_1 L_2$	The number of filters of two stages. $1 \le L_1 \le k_1 k_2, \quad 1 \le L_2 \le k_1 k_2$
$h_1 h_2$	The block size. $1 \le h_1 \le m, \; 1 \le h_2 \le n, \; Constraint: h_2 = [nh_1/m]$
R	The overlap ratio of block. which means R varies from 0 to 0.9 with the interval 0.1

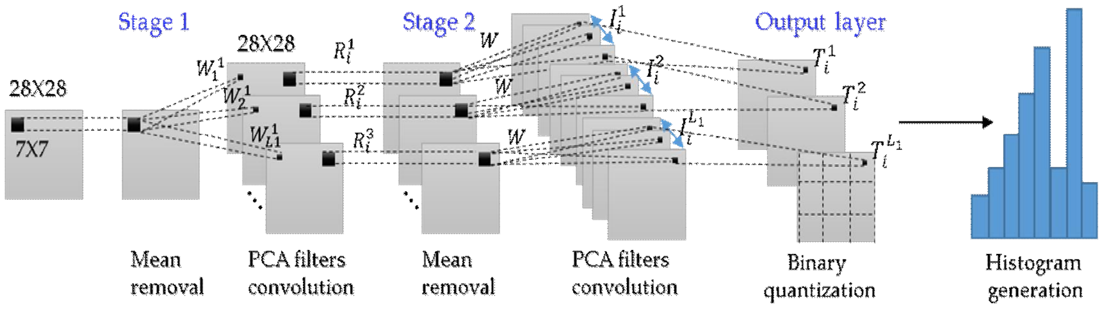


Figure 2.5. Diagram of PCANet.

The detailed description of the workflow:

(1) The original input consists of training images, such as ECG and face. If Step (4) is performed before, the input image is the PCA output.
(2) PCA filter bank extracts the PCANet filters.
(3) The PCA filterbank outputs Eigenvectors.
(4) Original images are convolved with the PCANet filters output for the next step.
(5) V is convolved with the PCANet filters output for the next step.
(6) After two PCA steps, the output image is the PCANet output.
(7) The output image is binarized and block-wise histograms are calculated. Here, we create a weight map and proceed with binary quantization and weighted combination of the elements in the input data.

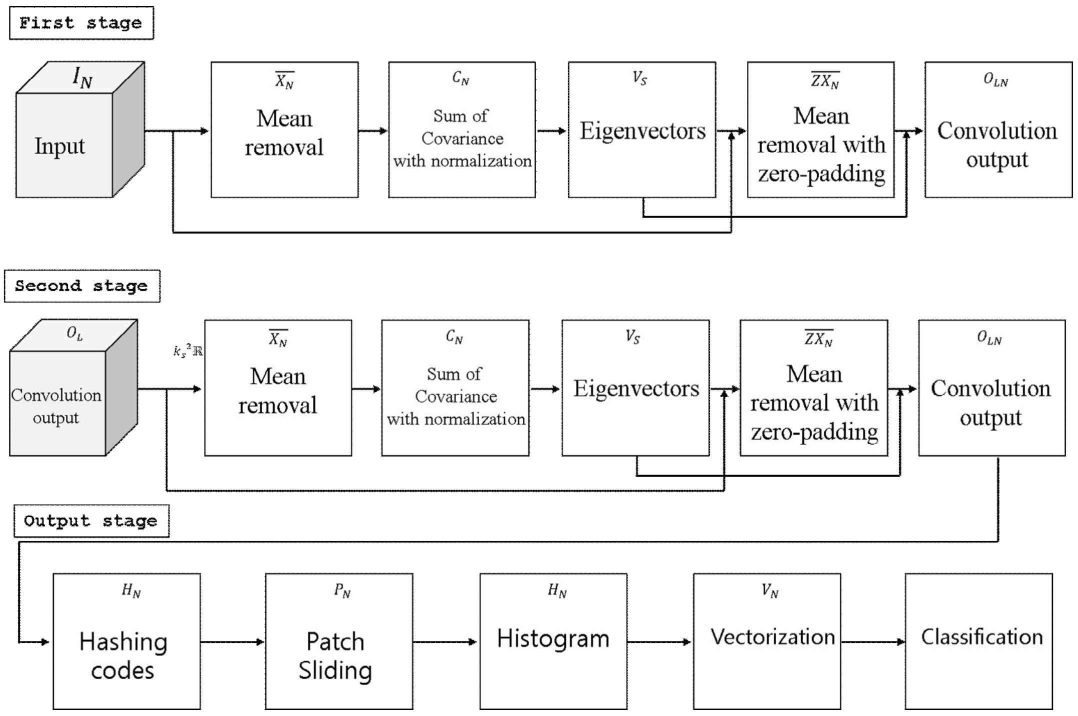


Figure 2.6. Workflow of PCANet.

Suppose we have N input training images $I_i, i = i = 1,2,\ldots,N), I_i \in R^{m \times n}$, and the patch size (or 2D filter size), where k_1 and k_2 are odd integers satisfying $1 \le k_1 \le m, 1 \le k_2 \le n$. Further, the number of filters in the layer is L_1, i.e., L_1 is the first stage and L_2 is the second stage. In the following, we describe the PCANet structure in detail. Let N input images $I_i, i = i = 1,2,\ldots,N)$ be concatenated as follows:

$$I = [I_1 I_2 \cdots I_N) \in R^{m \times Nn} \tag{10}$$

2.4.1 First Stage of PCANet

As shown in figure 2.6, the first stage of PCANet includes the following:

【Step 1】 The first patch sliding process.

Before sliding the images are padded to $I_i' \in R^{(m+k_1-1)\times(n+k_2-1)}$ out-of-range, input pixels are assumed zero. This ensures that all weights in the filters reach the entire images. We use a patch of the size $k_1 \times k_2$ to slide each pixel of the ith image $I_i' \in R^{(m+k_1-1)\times(n+k_2-1)}$, and subsequently reshape each $k_1 \times k_2$ matrix into a column vector, followed by concatenation to obtain a matrix:

$$X_i = [x_{i,1}\, x_{i,2} \cdots x_{i,mn}] \in R^{k_1 k_2 \times mn} \tag{11}$$

where $x_{i,j}$ denotes the jth vectorized patch in I_i. Thus, for all input training images $I_i, i = i = 1,2,\ldots,N)$, we can obtain the following matrix.

Figure 2.7 shows an example of the first patch sliding process. The data of the square matrix are vectorized according to the patch size. For example, if the patch size is [5 × 5], a square matrix is set as. In figure 2.7, a CIFAR image is used as an example. When there is an image of 32 * 32 * 3, a 5 * 5 patch image is resized to a row vector of 25 * 1, and the patch is moved to one image to obtain the mean removal value, which is also referred to as a center image in the PCA. The size of the image X_p after all patch slides is 25 * 784 25 * 784 ($k_s^2 \times (m-k_s+1)(n-k_s+1)$). Then, $patchX - patch\overline{X}$ is calculated to obtain the central image.

$$X = [X_1\, X_2 \cdots X_N] \in R^{k_1 k_2 \times Nmn} \tag{12}$$

$$X = [X_1\, X_2 \cdots X_N] \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)N} \tag{13}$$

【Step 2】 : The first mean remove process.

In this step, we subtract the patch mean from each patch and obtain the following:

$$patch\overline{X_i} = [\overline{X}_{i,1}\, \overline{X}_{i,2} \cdots \overline{X}_{i,mn}] \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)N} \tag{14}$$

By subtracting each patch image from the above equation, we can obtain the following expression.

$$\overline{X} = patchX - patch\overline{X_i} \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)} \tag{15}$$

【Step 3】 : The first PCA process.

In this step, the Eigenvalues and Eigenvectors of $\overline{X}$ are calculated from equation (15) using the PCA algorithm. The covariance is obtained by repeating the above equation from all patches and images.

$$\overline{X_{Cov}} = \sum_{N=1}^{NL} (\overline{X} \times \overline{X}^T) \tag{16}$$

where N is the number of images and L is the number of filters. The Eigenvector can be obtained according to the following equation.

$$\overline{X_{Cov}} \times \overline{X_{Cov}}^T v_{i=}\lambda_i v_i \tag{17}$$

$$\overline{X_{Cov}}^T \times \overline{X_{Cov}}(\overline{X_{Cov}}^T v_i) = \lambda_i(\overline{X_{Cov}}^T v_i) \tag{18}$$

$$v_i = \overline{X}^T v'_i, \quad s.t. \ v'_i = \overline{X}^T v^i \tag{19}$$

Let the number of L be chosen as the Eigenvector obtained above. The resulting Eigenvector is linearly combined with the center image of the zero-padded original image.

$$I^s i,l = v_i X_{padding} \tag{20}$$

s is the stage and the first stage is one. *l* means Eigenvectors 1 to *l*. By subtracting each patch image from the above equation, we can obtain the following expression. For each input image $I_i \in R^{m \times n}$, we obtain L_1 output images $I^s_{i,l}\, l = 1,2,...L_1, I^s_{i,l} \in R^{m \times n}$ after the first stage of PCANet. We denote I^s as

$$I^s = [I^s_{1,1} \cdots I^s_{1,L_1} \cdots I^s_{N,1} \cdots I^s_{N,L_1}] \in R^{m \times NL_1 n} \tag{21}$$

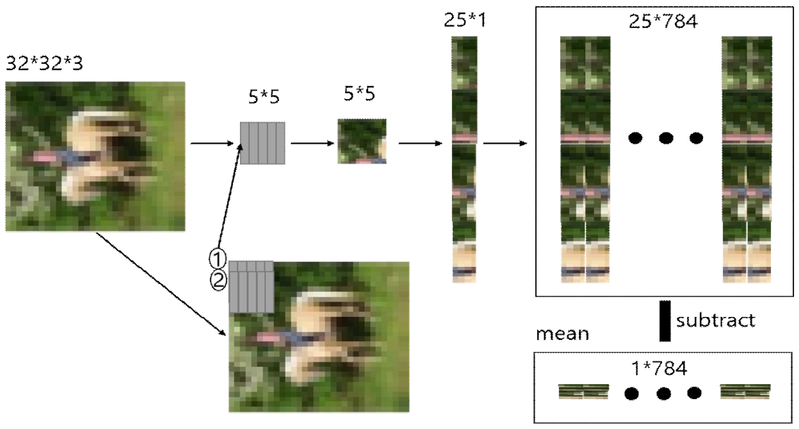


Figure 2.7. Example of patch step

2.4.2 Second Stage of PCANet

【Step 1】 : The second patch sliding process.

Almost repeating the same process as the first stage, as shown in figure 2.6, the second stage of PCANet also includes three steps:

Similar to step 1, we use a patch of size $k_1 \times k_2$ to slide each pixel of the ith image $I'_{i,l} \in R^{k_1 k_2 \times mn}$, $l = 1,2,...,L_1$ and obtain a matrix as follows:

$$Y_{i,l} = [y_{i,l,1} \;\; y_{i,l,2} \;\; \cdots y_{i,l,mn}] \in R^{k_1 k_2 \times mn}, l = 1,2,...,L_1, i = 1,2,...,N \tag{22}$$

where $y_{i,l,j}$ denotes the jth vectorized patch in $I^I_{i,l}$ Therefore, for all the input training images $I_i, i = i = 1,2,...,N)$, we can obtain the following matrix.

$$Y_{i,l} = [Y_{1,l} \, Y_{2,l} \cdots X_{N,l}] \in R^{k_1 k_2 \times Nmn} \tag{23}$$

We concatenate the matrices of all the L_1 filters and obtain a matrix

$$Y = [Y_1 \, Y_2 \cdots Y_{L_1}] \in R^{k_s^2 \times (m - k_s + 1)(n - k_s + 1)N} \tag{24}$$

【Step 2】 The second mean remove process

In this step, we subtract the patch mean from each patch and obtain the following:

$$patch\overline{X_i} = [\overline{Y}_{i,1} \overline{Y}_{i,2} \cdots \overline{Y}_{i,mn}] \in R^{k_s^2 \times (m - k_s + 1)(n - k_s + 1)N} \tag{25}$$

By subtracting each patch image from the above equation, we can obtain the following expression.

$$\overline{Y} = patch\,Y - patch\,\overline{Y_i} \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)} \tag{26}$$

【Step 3】 The second PCA process.

In this step, the Eigenvalues and Eigenvectors of $\overline{Y}$ are calculated from equation (26) using the PCA algorithm. The covariance is obtained by repeating the above equation from all patches and images.

$$\overline{Y_{Cov}} = \sum_{N=1}^{NL} (\overline{Y} \times \overline{Y}^T) \tag{27}$$

where N is the number of images and L is the number of filters. The Eigenvector can be obtained according to the following equation.

$$\overline{Y_{Cov}} \times \overline{Y_{Cov}}^T v_{i=} \lambda_i v_i \tag{28}$$

$$\overline{Y_{Cov}}^T \times \overline{Y_{Cov}} (\overline{Y_{Cov}}^T v_i) = \lambda_i (\overline{Y_{Cov}}^T v_i) \tag{29}$$

$$v_i = \overline{Y}^T v'_i, \quad s.t. \;\; v'_i = \overline{Y}^T v^i \tag{30}$$

Let the number of L be chosen as the Eigenvector obtained above. The resulting Eigenvector is linearly combined with the center image of the zero-padded original image.

$$I^s i, l = v_i X_{padding} \tag{31}$$

s is the stage and the first stage is one. l means Eigenvectors 1 to l. By subtracting each patch image from the above equation, we can obtain the following expression. For each input image $I_{i,l}^s \in R^{m \times n}$, we obtain L_2 output images $I_{i,l,\iota}^{ss}, \iota = 1,2,\ldots,L_2, I_{i,l,\iota}^{ss} \in R^{m \times n}$ after the second stage of PCANet.

Thus, we obtain NL_1L_2 image $I^{ss}_{i,l,\iota},\ l=1,2,...,L_1, \iota=1,2,...,L_2, i=1,2,...,N, I_i \in R^{m\times n}$ after and second stages. We denote I^{ss} as

$$I^{ss} = [I^{ss}_{1,1,1} \cdots I^{ss}_{1,1,L_2} \cdots I^{ss}_{1,L_1,1} \cdots I^{ss}_{1,L_1,L_2} \cdots I^{ss}_{N,1,1} \cdots I^{ss}_{N,1,L_2} \cdots I^{ss}_{N,L_1,1} \cdots I^{ss}_{N,L_1,L_2}] \tag{32}$$

2.4.3 Output Stage of PCANet

【Step 1】 Binary quantization.

In this step, we binarize the outputs of the second stage of PCANet, and obtain

$$P_{i,l,\iota} = H(I^{ss}_{i,l,\iota}), l = 1,2,...,L1;\ \iota = 1,2,...,L2;\ i-1,2,...,N \tag{33}$$

where $H(\bullet)$ is Heaviside step function whose value is one for positive entries and zero otherwise. We denote P as

$$P = [P_{1,1,1} \cdots P_{1,1,L_2} \cdots P_{1,L_1,1} \cdots P_{1,L_1,L_2} \cdots P_{N,1,1} \cdots P_{N,1,L_2} \cdots P_{N,L_1,1} \cdots P_{N,L_1,L_2}] \tag{34}$$

【Step 2】 Weight and sum.

Around each pixel, we view the vector of L_2 binary bits as a decimal number. This converts the binary images $P_{i,l,\iota}$ back into integer-valued images as follows:

$$T_{i,l} = \sum_{\iota=1}^{L2} 2^{\iota-1} P_{i,l,\iota} \tag{35}$$

We denote T as

$$T = [T_{1,1} \cdots T_{1,L_1} \cdots T_{N,1} \cdots T_{N,L_1}] \in R^{m\times NL_1 n} \tag{36}$$

【Step 3】 Block sliding.

We use a block of size $h_1 \times h_2$ to slide each of the L_1 images $T_{i,l}, l = 1, \ldots, L_1$, with overlap ratio R, and then reshape each $h_1 \times h_2$ matrix into a column vector, which is then concatenated to obtain a matrix

$$Z_{i,l} = [z_{i,l,1} \cdots z_{1,l,2} \cdots z_{i,lB}] \in R^{h_1 h_2 \times B} \tag{37}$$

where $z_{i,l,j}$ denotes the jth vectorized patch in $T_{i,l}, l = 1, \ldots, L_1$. B is the number of blocks when using a block of size $h_1 \times h_2$ to slide each $T_{i,l}, l = 1, \ldots, L_1$, which overlap ratio R and given by

$$\begin{aligned} B = & [1 + (m + k_1 - 1 - h_1)/stride1] \\ & \bullet [1 + (n + k_2) - 1 - h_2)/stride2], \end{aligned} \tag{38}$$

where stride1 and stride2 are vertical and horizontal steps, respectively,

$$\begin{aligned} stride1 &= \partial((1-R) \bullet h_1) \\ stride2 &= \partial((1-R) \bullet h_2) \end{aligned} \tag{39}$$

and $\partial(.)$ means round off. As we can see from equation (39), the number of blocks B is increasing as the overlap ratio R is increasing. For L_1 images, we concatenate $Z_{i,l}$ to obtain a matrix

$$Z_i = [z_{i,1} \cdots z_{i,2} \cdots z_{i,B}] \in R^{h_1 h_2 \times L_1 B} \quad i = 1, 2, \ldots, N \tag{40}$$

We denote Z as

$$Z_i = [z_{1,1} \cdots z_{1,B} \cdots z_{N,1} \cdots z_{N,B}] \in R^{h_1 h_2 \times L_1 BN} \tag{41}$$

【Step 4】 Histogramming

We compute the histogram (with 2^{L_2} bins) of the decimal values in each column of Z_i and concatenate all the histograms into one vector and obtain

$$f_i = [Hist(z_{i,1,1}) \cdots Hist(z_{i,1,B}) \cdots Hist(z_{i,L_1,1}) \cdots Hist(z_{i,L_1,B})] \in R^{(2^{L_2})L_1 B} \tag{42}$$

which is the "feature" of the input image I_i and Hist(.) denotes the histogram operation. We denote f as

$$f_i = [f_1 \cdots f_N) \in R^{(2^{L_2})L_1 BN} \tag{43}$$

The feature vector is then sent to a classifier, for example, support vector machine (SVM), Euclidean distance, etc.

Algorithm 2 PCANet

```
1.  Input: training data $x_{i=1}^{N}, X_i \in R^{m\times n}, L_1, L_2, The\,patch\,size\,is\,k_1 \times k_2 i$

2.  Reading: Image data or signal data

3.  Initialization of variable:
    The size of the patch ← select the size of the patch in PCANet
    The number of filters ← select the number of filters in PCANet

4.  Stage 1 of PCANet
        - Patch mean-removal: $\overline{Y} = patch\,Y - patch\,\overline{Y_i} \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)}$

        - Compute convolution using PCA:

              $v_i = \overline{Y}^T v'_i, \quad s.t. \ \ v'_i = \overline{Y}^T v^i$

        - Output : $I^s i, l = v_i X_{padding}$

5.  Stage 2 of PCANet
        - Patch mean-removal: $\overline{Y} = patch\,Y - patch\,\overline{Y_i} \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)}$

        - Compute convolution using PCA:

              $v_i = \overline{Y}^T v'_i, \quad s.t. \ \ v'_i = \overline{Y}^T v^i$

        - Output : $I^s i, l = v_i X_{padding}$

6.  Output layer
        - Binary hashing: compute the decimal-valued image

          $P_{i,l,\iota} = H(I_{i,l,\iota}^{ss}), l = 1,2,\ldots,L1;\ \iota = 1,2,\ldots,L2;\ i - 1,2,\ldots,N$

        - Histogram:

    $f_i = [Hist(z_{i,1,1}) \cdots Hist(z_{i,1,B}) \cdots Hist(z_{i,L_1,1}) \cdots Hist(z_{i,L_1,B})] \in R^{(2^{L_2})L_1 B}$

        - Output : $f_i = [f_1 \cdots f_N) \in R^{(2^{L_2})L_1 BN}$
```

2.5 LBP

The LBP (local binary pattern) is a feature developed for classifying the original image texture and is used for other image recognition applications such as face recognition. The local binary pattern (LBP) is a value calculated for every pixel of the image, and is an index value obtained by coding the relative brightness change of the surrounding 3 x 3 region of each pixel in binary (If the pixel is brighter than the center pixel, it has a value of 1; if it is dark, it is coded as 0, and then, the binary connected with these values is used as an index for the local texture). Figure 2.8 shows the principle of basic LBP.

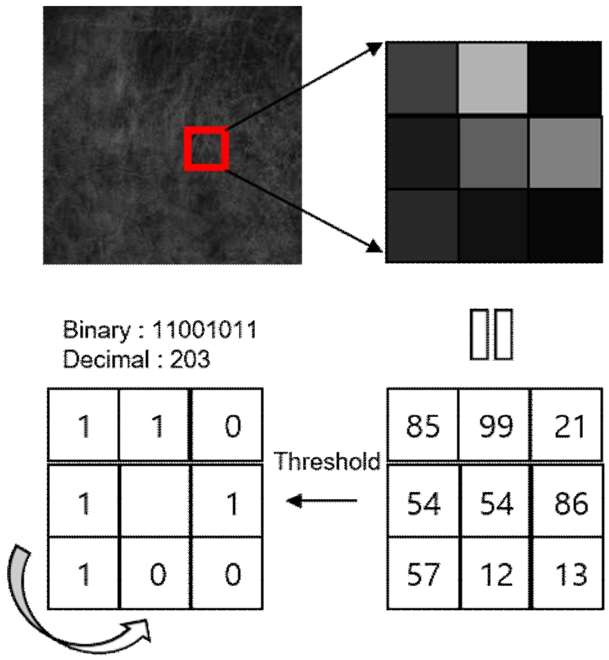


Figure 2.8. Principle of basic LBP

It is the original LBP application that uses this histogram as a texture model for the corresponding image area after obtaining the histogram for the index value calculated through each pixel (for the purpose of classifying the texture of lawn, forest, land, wall, etc).

In the following example of using LBP for face recognition, the face region is divided into cells of a certain size. Then, a histogram for LBP is obtained for each cell (a histogram of LBP index values belonging to the cell). There is a method of using the vector obtained by connecting the histograms obtained in a row as a final feature

A pixel located at the center in a 3x3 cell and a neighboring eight pixels are compared with each other. In figure 2.8, the value of the center pixel is 54, so if the neighboring pixel value is greater than or equal to 54, it is set to 1, and if it is less than 54, the threshold value is set to zero. If you list them in order, we get a binary value like 11001011. Calculating the binary value as a decimal value is 203. Possible values are the number of 256 cases from 0 to 255 in total. We calculate this for all pixels and make a histogram. It is not simply a histogram of pixel values, but a histogram of LBP values. Then a total of 256 bins will be filled. The texture of one image is represented by 256 numbers. These primitive LBP have robust characteristics even when the brightness of the image changes. Figure 2.9 shows the LBP applied face recognition.

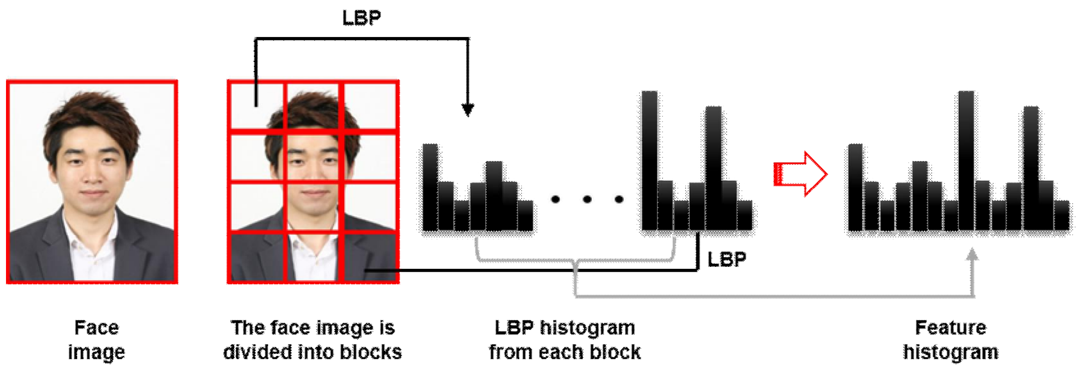

LBP
LBP
Face image
The face image is divided into blocks
LBP histogram from each block
Feature histogram

Figure 2.9. LBP applied face recognition

Ⅲ. Proposed ICANet

This section presents ICANet, the proposed method of this paper.

3.1 Statistically independent basis image ICANet (ICANet1)

ICANet1 (statistically independent basic image Net) is a modified algorithm from ICA1, which is an algorithm that extracts independent components through ICA filter using Eigenvectors obtained from PCA. The first stage is a simple deep learning type in which the PCA and ICA algorithms form a network in the form of a cascade. In the second step, three algorithms, PCA, ICA, and LBP, are performed using image patch data. Finally, the hashing code step similar to LBP and the feature using the histogram is vectorized. Finally, the final classification is done using SVM. Figure 3.1 below shows the structure of ICANet1.

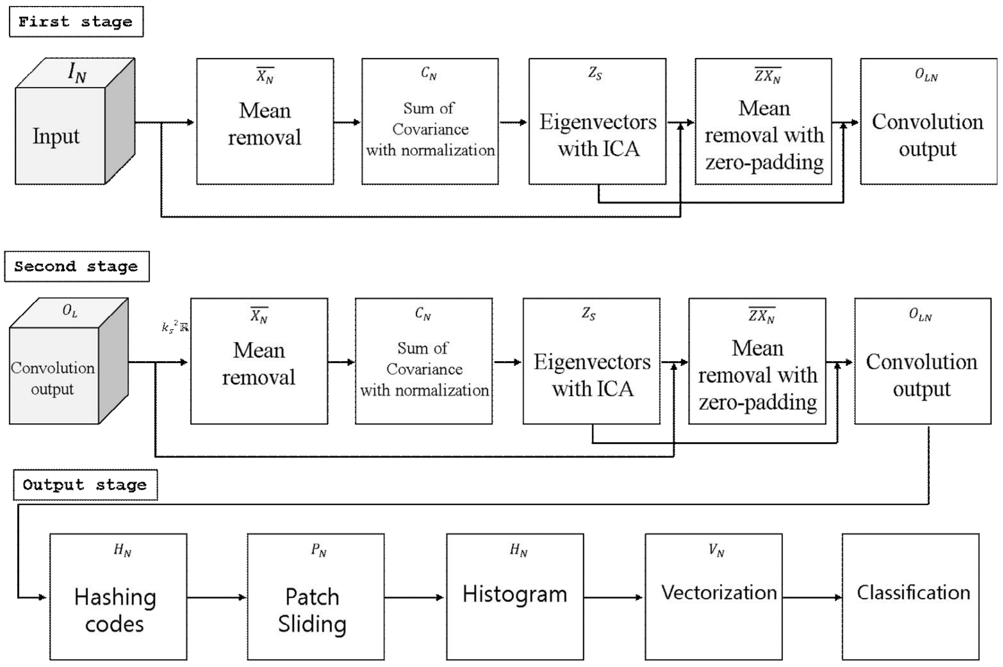


Figure 3.1. Structure of ICANet1.

To find a statistically independent basic image set according to a set of faces, the independent components of the face image are separated based on the image synthesis model shown in the figure below. Since existing Eigenvectors are not statistically independent, it is the goal of ICANet1 to find Eigenvectors that are statistically independent using ICA. The face image of X is assumed a linear mixture of an unknown set of statistically independent source images S, where A is an unknown mixing matrix. The source is reconstructed by a W learning filter matrix producing a statistically independent output. The image consists of a row of input matrix X. Is provided as a set of independent base images for the face, using the input images in the row of X, where the ICA output in the row of U is the images. Such a base image may be regarded as a statistically independent set of facial features, wherein the pixel values of each feature image are unpredictable in the pixel values of the other feature images. The ICA expression consists of. The coefficients for the linear combination of the independent basic images of the U that make up each face image as shown in Fig. In this model, the coefficient matrix B is obtained from the mixing matrix A$\triangleq W_I^{-1}$. Each row of A contains a coefficient b for one image x. Figure 3.2 shows an image synthesis model.

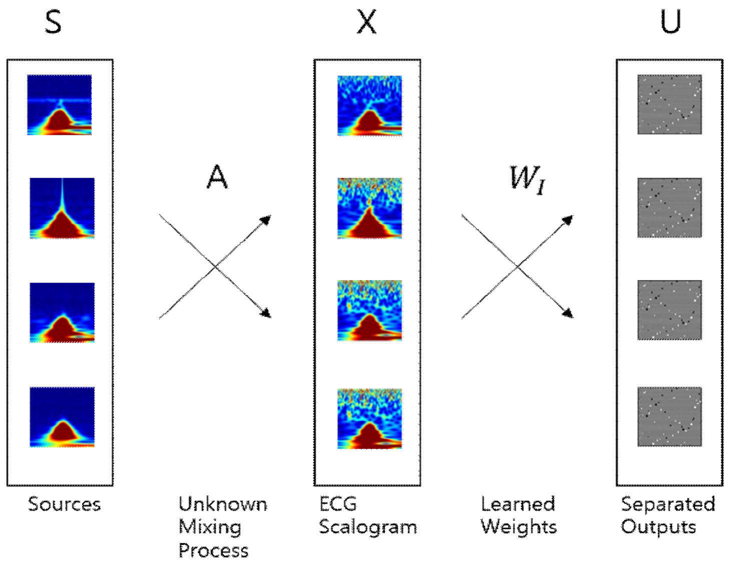


Figure 3.2. An image synthesis model.

P_m is represented by a matrix containing the first m principal component axes in the column. We perform ICA through P_m^T and generate a matrix of m independent source images in the row of U. Here, the coefficients b for the linear combination of the base images in U constituting the face images of X is determined as follows. The principal component representation of the zero-mean image sets in P_m based X is defined as $R_m = X^* P_m$. The least square error approximation value of $X_{rec} = R_m{}^* P_m^T$ is obtained by X. The ICA algorithm produces the following matrix $W_I = W^* W_z$

$$W_I{}^* P_m^T = U \rightarrow P_m^T = W_I^{-1} U \tag{44}$$

$$X_{rec} = R_m{}^* P_m^T \rightarrow X_{rec} = R_m{}^* W^{-1} U \tag{45}$$

where W_Z is the sphering matrix defined as $W_z = 2^* < XX^T >^{-\frac{1}{2}}$. Thus, the row of $R_m{}^* W_I^{-1}$ includes the least-square error approximation of the linear combination coefficient of X, which is statistically independent of U, constituting X_{rec} as in PCA. Since U is an independent component representation of a face image based on statistically independent sets of m feature images, U is given by the rows of the matrix.

$$B = R_m{}^* W_I^{-1} \tag{46}$$

To obtain the $R_{test} = X_{test}{}^* p_m$ we can obtain the test image by using the principal component representation based on the training image and calculating the $B_{test} = R_{test}{}^* W_I^{-1}$. The ICA representation of the face did not require a PCA step and was used to meet the purpose. The purpose is as follows:

1. Reducing the number of sources to a manageable number.
2. Providing a convenient way to calculate the representation of a test image. Without using the PCA step, the length of the U-row, $B_{test} = X_{test}{}^* U^T$, can be normalized to normalize U and U to obtain $B = W_I^{-1}$ and B_{test} without computing the pseudo-inverse.

3.1.1 First Stage of ICANet1

【Step 1】 The first patch sliding process.

Before sliding the images are padded to $I_i' \in R^{(m+k_1-1)\times(n+k_2-1)}$ out-of-range, input pixels are assumed zero. This ensures that all weights in the filters reach the entire images. We use a patch of the size $k_1 \times k_2$ to slide each pixel of the ith image $I_i' \in R^{(m+k_1-1)\times(n+k_2-1)}$, and subsequently reshape each $k_1 \times k_2$ matrix into a column vector, followed by concatenation to obtain a matrix:

$$X_i = [x_{i,1}\, x_{i,2} \cdots x_{i,mn}] \in R^{k_1 k_2 \times mn} \tag{47}$$

where $x_{i,j}$ denotes the jth vectorized patch in I_i. Thus, for all input training images $I_i, i = i = 1,2,\ldots,N)$, we can obtain the following matrix.

$$X = [X_1\, X_2 \cdots X_N] \in R^{k_1 k_2 \times Nmn} \tag{48}$$

$$X = [X_1\, X_2 \cdots X_N] \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)N} \tag{49}$$

【Step 2】 : The first mean remove process.

In this step, we subtract the patch mean from each patch and obtain the following:

$$patch\overline{X_i} = [\overline{X}_{i,1}\, \overline{X}_{i,2} \cdots \overline{X}_{i,mn}] \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)N} \tag{50}$$

By subtracting each patch image from the above equation, we can obtain the following expression.

$$\overline{X} = patchX - patch\overline{X_i} \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)} \tag{51}$$

【Step 3】 The first PCA process with ICA

In this step, the Eigenvalues and Eigenvectors of $\overline{X}$ are calculated from equation (51) using the PCA algorithm. The covariance is obtained by repeating the above equation from all patches and images.

$$\overline{X_{Cov}} = \sum_{N=1}^{NL} (\overline{X} \times \overline{X}^{T}) \tag{52}$$

where N is the number of images and L is the number of filters. The Eigenvector can be obtained according to the following equation.

$$\overline{X_{Cov}} \times \overline{X_{Cov}}^{T} v_{i} = \lambda_i v_i \tag{53}$$

$$\overline{X_{Cov}}^{T} \times \overline{X_{Cov}} (\overline{X_{Cov}}^{T} v_i) = \lambda_i (\overline{X_{Cov}}^{T} v_i) \tag{54}$$

$$v_i = \overline{X}^{T} v'_{i}, \qquad s.t. \;\; v'_{i} = \overline{X}^{T} v^{i} \tag{55}$$

$$U = W_{ICA} v_{i(PCA)} \tag{56}$$

Let the number of L be chosen as the Eigenvector obtained above. To apply the ICA algorithm to estimate the independent component, we apply Eigenvectors to ICA. The first ICA centering and the whitening algorithm is applied, and then the data normalization is performed. Then, the initial weight is set randomly, and the initial weight and the data obtained from the whitening are linearly combined. There are two methods of ICA (Kurtosis, Negentropy) and Kurtosis method. Therefore, the expression is as follows.

$$kurt(y) = Ey^{4}| - 3(Ey^{2})^{2} \tag{57}$$

The values obtained above and whitening data are linearly combined and normalized. Singular value decomposition is performed to remove the correlation. Then, the delta value is obtained by carrying out the dot product operation with the weight obtained from the singular value decomposition and the initial weight obtained randomly and subtracting the value from 1. If the delta value is greater than 0.000001, the data obtained from the final weight and whitening are linearized, and this value is called independent components. The obtained independent components U and the linear image of the center image of the zero padded original image are combined. A linear combination of the obtained independent components U and the center image of the zero-padded original image are performed.

$$I^s i,l = U_i X_{padding} \tag{58}$$

s is the stage and the first stage is one. l means independent elements 1 to l. By subtracting each patch image from the above equation, we can obtain the following expression. For each input image $I_i \in R^{m \times n}$, we obtain L_1 output images $I^s_{i,l} l = 1,2,...L_1, I^s_{i,l} \in R^{m \times n}$ after the first stage of ICANet. We denote I^s as

$$I^s = [I^s_{1,1} \cdots I^s_{1,L_1} \cdots I^s_{N,1} \cdots I^s_{N,L_1}] \in R^{m \times NL_1 n} \tag{59}$$

Figure 3.3 shows first stage of ICANet1 and PCA and ICA are cascaded. In the first stage, a two-dimensional convolutional output is obtained. Figure 3.4 shows feature CU-ECG database in the first stage of ICANet1.

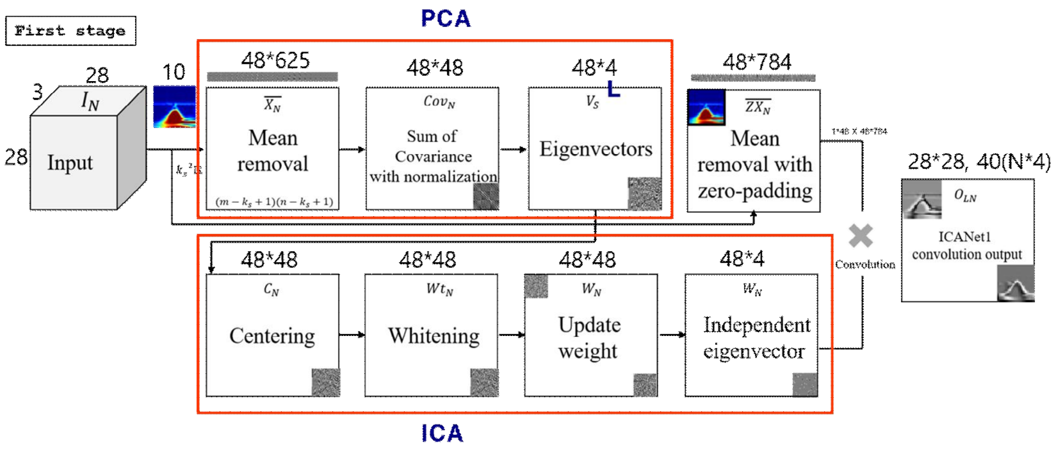


Figure 3.3. First stage of ICANet1.

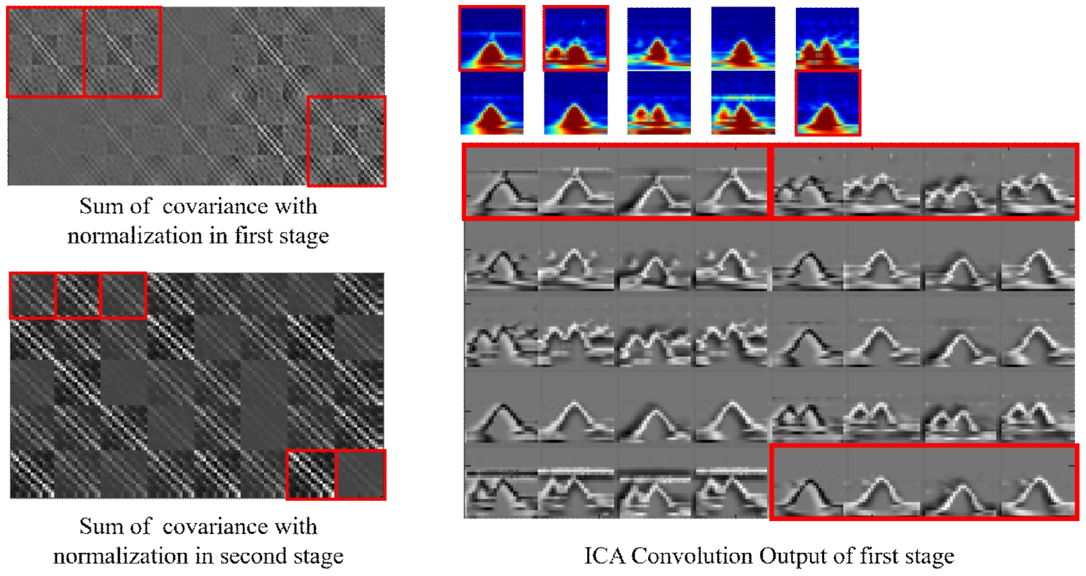


Figure 3.4. Feature CU-ECG database in first stage of ICANet1.

One covariance is generated for each image, and the second stage is affected by the number of filters to generate 40 covariance ($L_1L_2=4$), resulting in 40 convolution outputs in the first stage.

3.1.2 Second Stage of ICANet1

A hashing code is generated based on the convoluted feature and the weight map. The values generated by the hashing code are also projected onto a weight map.

【Step 1】 The second patch sliding process.

Almost repeating the same process as the first stage, as shown in Figure 3.1, the second stage of ICANet1 also includes three steps: Similar to step 1, we use a patch of size $k_1 \times k_2$ to slide each pixel of the ith image $I'_{i,l} \in R^{k_1 k_2 \times mn}$, $l = 1,2,...,L_1$ and obtain a matrix as follows:

$$Y_{i,l} = [y_{i,l,1} \;\; y_{i,l,2} \;\; \cdots \; y_{i,l,mn}] \in R^{k_1 k_2 \times mn}, l = 1,2,...,L_1, i = 1,2,...,N \tag{60}$$

where $x_{i,j}$ denotes the jth vectorized patch in I_i. Thus, for all input training images $I_i, i = i = 1,2,...,N)$, we can obtain the following matrix.

$$Y_{i,l} = [Y_{1,l} \, Y_{2,l} \cdots X_{N,l}] \in R^{k_1 k_2 \times Nmn} \tag{61}$$

We concatenate the matrices of all the L_1 filters and obtain a matrix

$$Y = [Y_1 \, Y_2 \cdots Y_{L_1}] \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)N} \tag{62}$$

【Step 2】 The second mean remove process

In this step, we subtract the patch mean from each patch and obtain the following:

$$patch\overline{X_i} = [\overline{Y}_{i,1} \overline{Y}_{i,2} \cdots \overline{Y}_{i,mn}] \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)N} \tag{63}$$

By subtracting each patch image from the above equation, we can obtain the following expression.

$$\overline{Y} = patch\,Y - patch\,\overline{Y_i} \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)} \tag{64}$$

【Step 3】 The second PCA process with ICA

In this step, the Eigenvalues and Eigenvectors of $\overline{Y}$ are calculated from equation (51) using the PCA algorithm. The covariance is obtained by repeating the above equation from all patches and images.

$$\overline{Y_{Cov}} = \sum_{N=1}^{NL} (\overline{Y} \times \overline{Y}^T) \tag{65}$$

where N is the number of images and L is the number of filters. The Eigenvector can be obtained according to the following equation.

$$\overline{Y_{Cov}} \times \overline{Y_{Cov}}^T v_{i=}\lambda_i v_i \tag{66}$$

$$\overline{Y_{Cov}}^T \times \overline{Y_{Cov}}(\overline{Y_{Cov}}^T v_i) = \lambda_i (\overline{Y_{Cov}}^T v_i) \tag{67}$$

$$v_i = \overline{Y}^T v'_i, \quad s.t. \ v'_i = \overline{Y}^T v^i \tag{68}$$

$$U = W_{ICA} v_{i(PCA)} \tag{69}$$

A linear combination of the obtained independent components U and the center image of the zero-padded original image are performed.

$$I^s i, l = U_i X_{padding} \tag{70}$$

s is the stage and the first stage is one. l means independent elements 1 to l. By subtracting each patch image from the above equation, we can obtain the following expression. By subtracting each patch image from the above equation, we can obtain the following expression. For each input image $I_{i,l}^{s} \in R^{m \times n}$, we obtain L_2 output images $I_{i,l,\iota}^{ss}, \iota = 1,2,...,L_2, I_{i,l,\iota}^{ss} \in R^{m \times n}$ after the second stage of PCANet. Thus, we obtain NL_1L_2 image $I_{i,l,\iota}^{ss},\ l = 1,2,...,L_1, \iota = 1,2,...,L_2, i = 1,2,...,N, I_i \in R^{m \times n}$ after and second stages. We denote I^{ss} as

$$I^{ss} = [I_{1,1,1}^{ss} \cdots I_{1,1,L_2}^{ss} \cdots I_{1,L_1,1}^{ss} \cdots I_{1,L_1,L_2}^{ss} \cdots I_{N,1,1}^{ss} \cdots I_{N,1,L_2}^{ss} \cdots I_{N,L_1,1}^{ss} \cdots I_{N,L_1,L_2}^{ss}] \tag{71}$$

Figure 3.5 shows the second stage of ICANet1 and it is a cascade of PCA, ICA, and LBP algorithms. The second stage processes the data for each image, updating the Eigenvectors and independent Eigenvectors accordingly. Therefore, the feature values of the output stage also show different feature values. Figure 3.6 shows the CU-ECG database in the second stage of ICANet1. The number of Eigenvectors is increased based on the number of filters ($L_2 = 4$), and four feature values are extracted from one image.

The detailed description of the workflow is below:

(1) In the PCA phase, the center image is obtained, and the Eigenvector is obtained by using the normalization of the covariance.

(2) Perform the centering and whitening process in the ICA stage, and obtain an independent Eigenvector using updated weights.

(3) The center image is obtained from the basic data and convolution is performed with the zero padded image and the independent Eigenvector obtained from ICANet (2), and the ICA convolution output value is extracted.

(4) In LBP step, feature value is extracted through hashing code step, patch sliding step, histogram, and step.

(5) Finally, the feature obtained in (4) is vectorized, and classification is performed using the feature value.

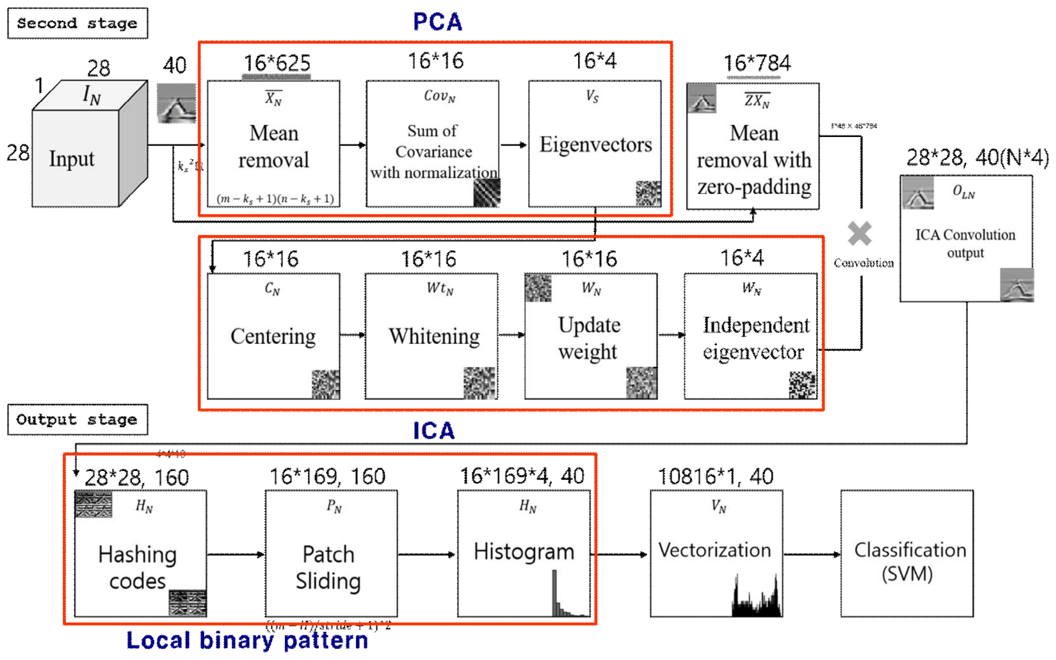


Figure 3.5. Second and output stage of ICANet1.

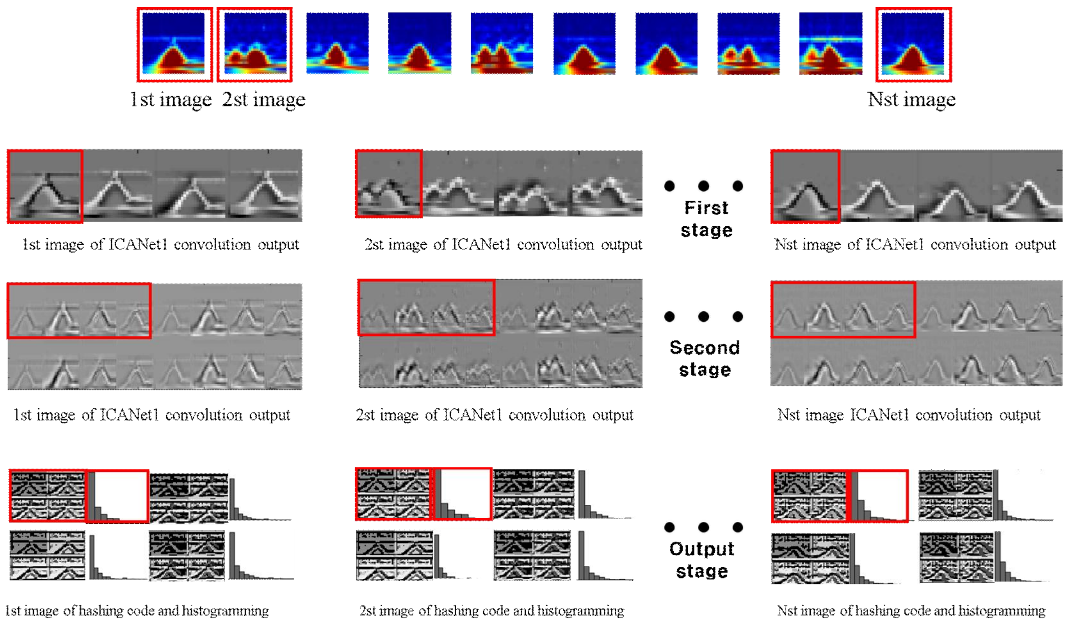
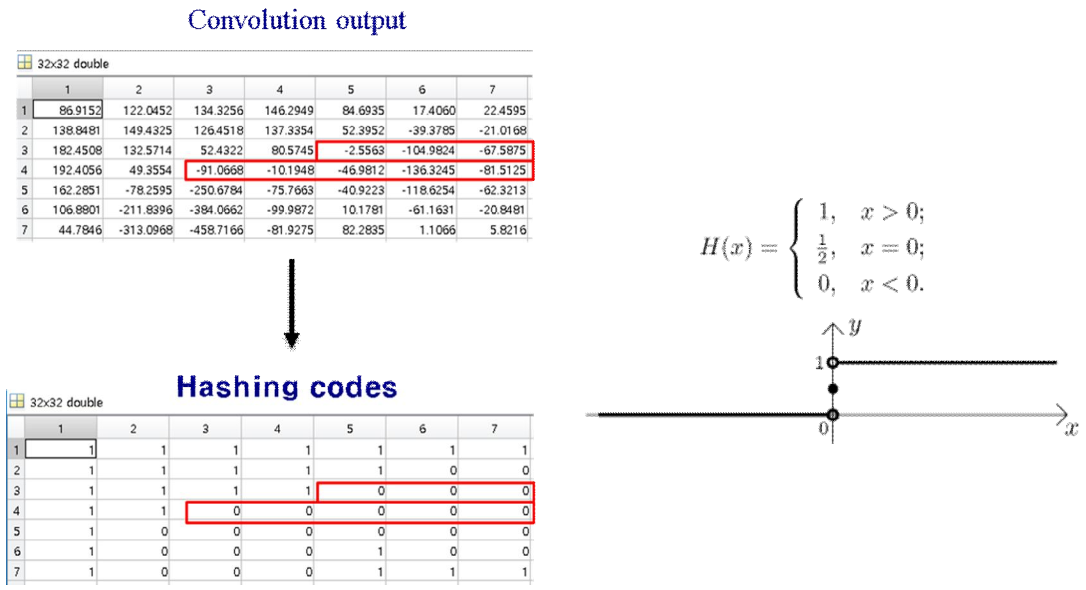


Figure 3.6. Feature CU-ECG database in second and output stage of ICANet1.

3.1.3 Output Stage of ICANet1

Step 1 to Step 3 is the same as the previous PCANet algorithm. The following lists the advantages of ICANet. There are PCANet algorithms, but papers that have been studied with various structures of ICANet are a new challenge.



32x32 double

	1	2	3	4	5	6	7
1	86.9152	122.0452	134.3256	146.2949	84.6935	17.4060	22.4595
2	138.8481	149.4325	126.4518	137.3354	52.3952	-39.3785	-21.0168
3	182.4508	132.5714	52.4322	80.5745	-2.5563	-104.9824	-67.5875
4	192.4056	49.3554	-91.0668	-10.1948	-46.9812	-136.3245	-81.5125
5	162.2851	-78.2595	-250.6784	-75.7663	-40.9223	-118.6254	-62.3213
6	106.8801	-211.8396	-384.0662	-99.9872	10.1781	-61.1631	-20.8481
7	44.7846	-313.0968	-458.7166	-81.9275	82.2835	1.1066	5.8216

32x32 double

	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	1	1	1	1	0	0
3	1	1	1	1	0	0	0
4	1	1	0	0	0	0	0
5	1	0	0	0	0	0	0
6	1	0	0	0	1	0	0
7	1	0	0	0	1	1	1

$$H(x) = \begin{cases} 1, & x > 0; \\ \frac{1}{2}, & x = 0; \\ 0, & x < 0. \end{cases}$$

Figure 3.7. Hashing codes in output stage of ICANet1.

The hashing code is similar to the local binary patterns. It is a binary algorithm that codes zero if the feature value is less than zero, and one if the feature value is bigger than zero, as in figure 3.7. Therefore, the independence component obtained from ICA is simplified to zero and one. Figure 3.7 shows the hashing codes in the output stage of ICANet1.

【Advantage of ICANet】

(1) The network structure is simple and computationally efficient.

(2) The ICA filter is trained with an unsupervised algorithm using unlabeled samples, which is practical.

(3) Compared to deep learning models, each layer parameter in ICANet can be easily trained.

```
Algorithm 3 ICANet1
1.  Reading: Image data or signal data
2.  Initialization of variable:
The size of the patch ← select the size of the patch in ICANet1
The number of filters ← select the number of filters in ICANet1

3.  Stage 1 of ICANet1
        - Patch mean-removal: $\overline{Y} = patch\,Y - patch\,\overline{Y_i} \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)}$

        - Compute Eigenvector using PCA:

                    $v_i = \overline{Y}^T v'_i, \quad s.t.\ \ v'_i = \overline{Y}^T v^i$

        - Compute convolution using ICA:

                    $U = W_{ICA} v_{i_{(PCA)}}$

        - Output: $I^s i,l = UX_{padding}$

4.  Stage 2 of ICANet1
        - Patch mean-removal: $\overline{Y} = patch\,Y - patch\,\overline{Y_i} \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)}$

        - Compute Eigenvector using PCA:

                    $v_i = \overline{Y}^T v'_i, \quad s.t.\ \ v'_i = \overline{Y}^T v^i$

        - Compute convolution using ICA:

                    $U = W_{ICA} v_{i_{(PCA)}}$

        - Output: $I^s i,l = UX_{padding}$

5.  Output layer
        - Binary hashing: compute the decimal-valued image

            $P_{i,l,\iota} = H(I^{ss}_{i,l,\iota}),\ l = 1,2,\ldots,L1;\ \iota = 1,2,\ldots,L2;\ i-1,2,\ldots,N$

        - Histogram:

    $f_i = [Hist(z_{i,1,1}) \cdots Hist(z_{i,1,B}) \cdots Hist(z_{i,L_1,1}) \cdots Hist(z_{i,L_1,B})] \in R^{(2^{L_2})L_1B}$

    Output: $f_i = [f_1 \cdots f_N) \in R^{(2^{L_2})L_1BN}$
```

3.2 Factorial ICANet (ICANet2)

ICANet2 is a modified algorithm in ICANet. It extracts independent components by applying feature vectors obtained from PCA to ICA filters. To find a statistically independent basic image set according to a set of ECG. Since existing feature vectors are not statistically independent, it is the goal of ICANet2 to find feature vectors that are statistically independent using ICA. Figure 3.8 shows the structure of ICANet2.

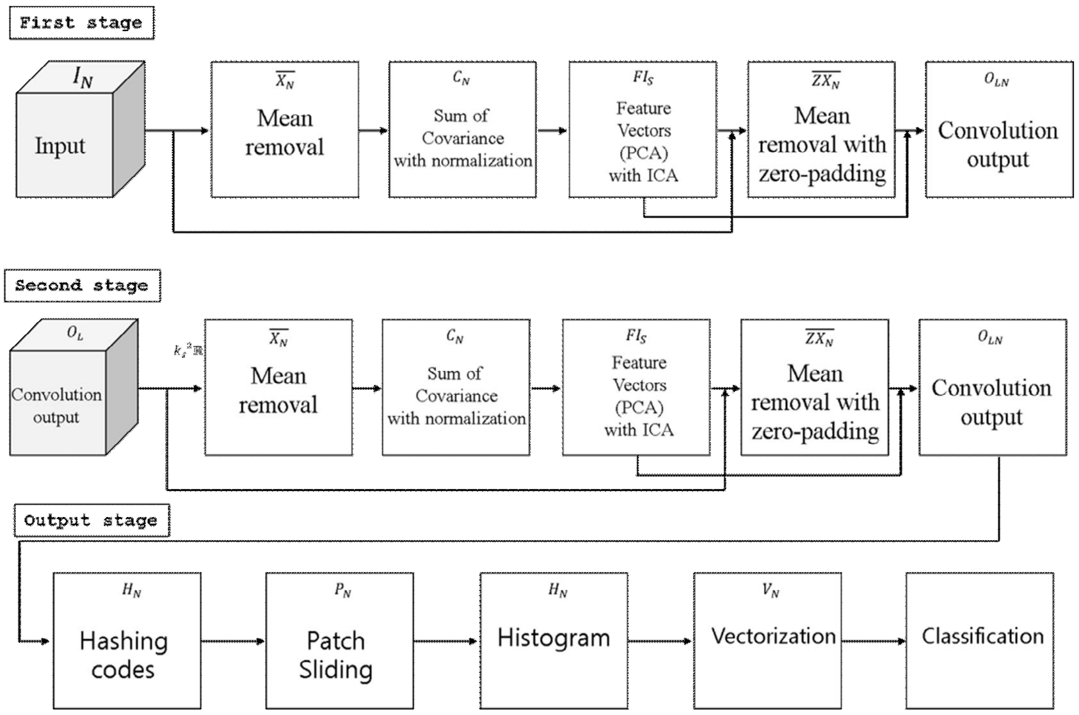


Figure 3.8. Structure of ICANet2.

The ICA in Architecture 1 finds the weight vector in the direction of statistical dependence at the population of the face image at the pixel location. Projecting the data to these weights creates an independent set of images that cannot predict the pixel gray value of one image in the gray value of another image. These independent images span the subspace of the face image defined by the first 75 PCA Eigenvectors and each face represents a measure of the linear combination of these independent template images that make up each face image.

The base image obtained in Architecture 1 was space-independent, but the coefficients coding each face are not. By changing the architecture of the independent component analysis, the coefficients were defined as statistically independent of the second representation. In other words, the second ICA architecture finds the factorial code for the face image. Changes to the architecture are consistent with transposing the input. Each facial image was treated as a coded observation with a gray value at each pixel location. ICA in Architecture 2 finds the weight vector in the direction of statistical dependency in the face code across the face set. Projecting the data to these weights produces a set of independent coding variables that will replace the "pixel location" where the value of a particular coding variable cannot be predicted by other coding variables. Each face is represented by the value taken by this new set of independent coding variables. The similarity of ICA-factorial representation (ICA2) and principal component representation is straightforward. The principal component coefficients constitute uncorrelated facial codes, while ICA2 coefficients constitute independent face codes. Figure 3.9 shows an image synthesis model in ICANet2 and figure 3.10 shows two architectures for performing ICA on images.

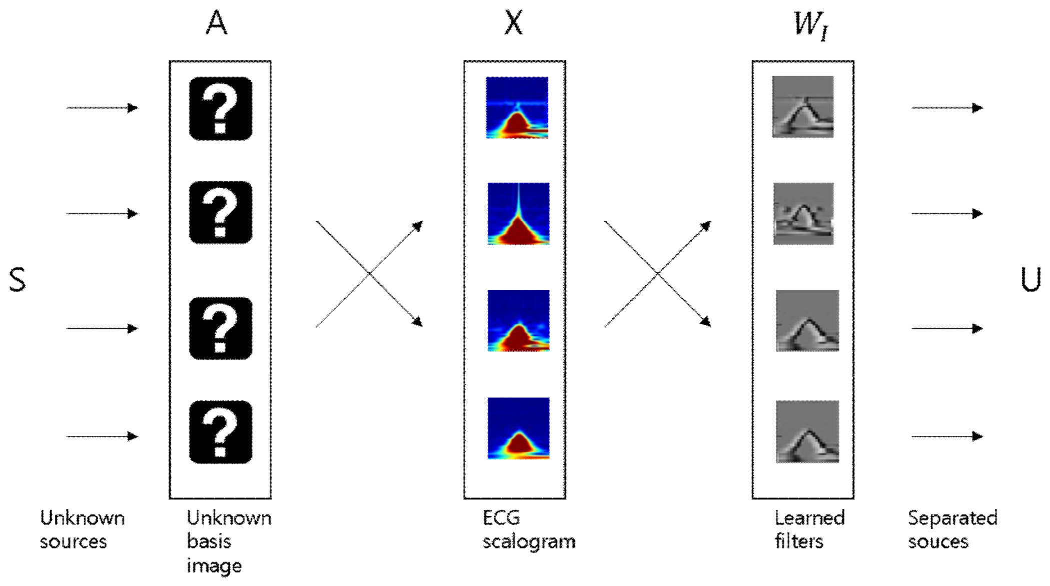


Figure 3.9. Image synthesis model in ICANet2.

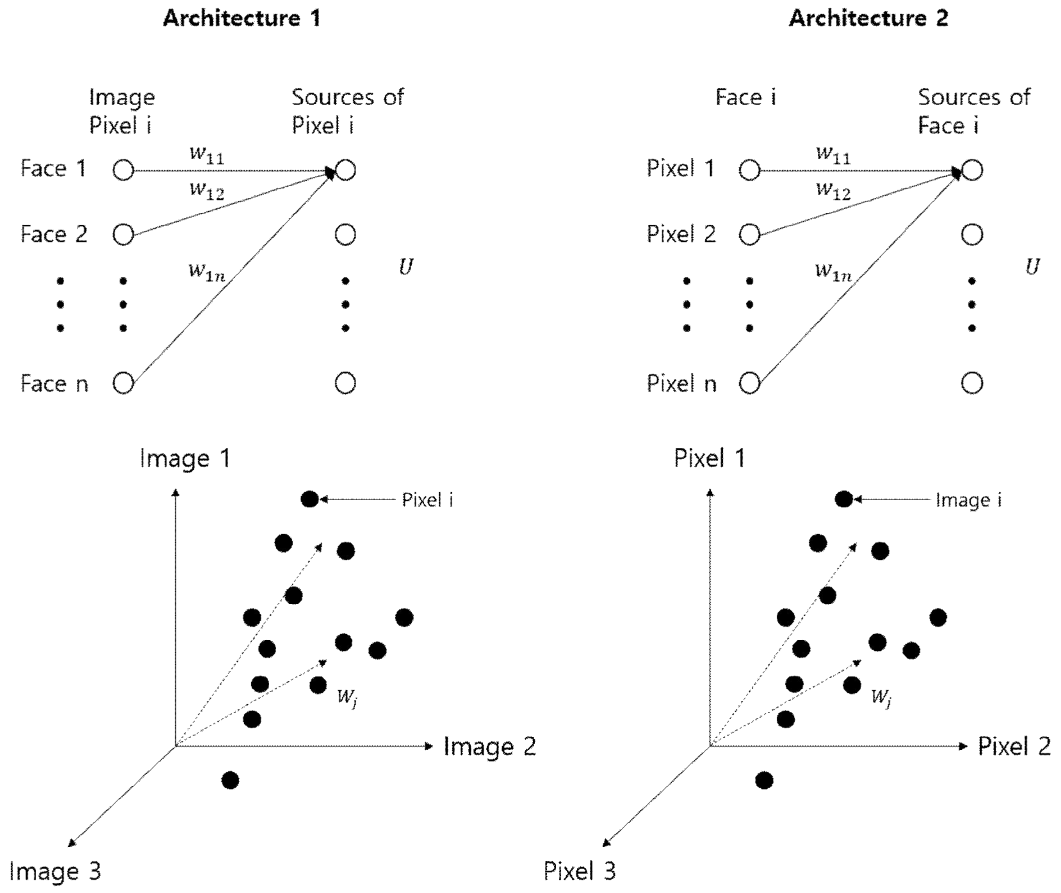


Figure 3.10. Two architectures for performing ICA on images.

3.2.1 First Stage of ICANet2

The first stage of ICANet2 is similar to the first stage of ICANet1, but there is a difference in the third step.

【Step 1】 The first patch sliding process.

Before sliding the images are padded to $I_i^{'} \in R^{(m+k_1-1)\times(n+k_2-1)}$ out-of-range, input pixels are assumed zero. This ensures that all weights in the filters reach the entire images. We use a patch of the size $k_1 \times k_2$ to slide each pixel of the ith image $I_i^{'} \in R^{(m+k_1-1)\times(n+k_2-1)}$, and subsequently reshape each $k_1 \times k_2$ matrix into a column vector, followed by concatenation to obtain a matrix:

$$X_i = [x_{i,1}\, x_{i,2} \cdots x_{i,mn}] \in R^{k_1 k_2 \times mn} \tag{72}$$

where $x_{i,j}$ denotes the jth vectorized patch in I_i. Thus, for all input training images $I_i, i = i = 1,2,\ldots,N)$, we can obtain the following matrix.

$$X = [X_1\, X_2 \cdots X_N] \in R^{k_1 k_2 \times Nmn} \tag{73}$$

$$X = [X_1\, X_2 \cdots X_N] \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)N} \tag{74}$$

【Step 2】 The first mean remove process.

In this step, we subtract the patch mean from each patch and obtain the following:

$$patch\overline{X_i} = [\overline{X}_{i,1}\, \overline{X}_{i,2} \cdots \overline{X}_{i,mn}] \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)N} \tag{75}$$

By subtracting each patch image from the above equation, we can obtain the following expression.

$$\overline{X} = patchX - patch\overline{X_i} \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)} \tag{76}$$

【Step 3】 The first PCA process with ICA

In this step, the Eigenvalues and Eigenvectors of $\overline{X}$ are calculated from equation (76) using the PCA algorithm. The covariance is obtained by repeating the above equation from all patches and images

$$\overline{X_{Cov}} = \sum_{N=1}^{NL} (\overline{X} \times \overline{X}^T) \tag{77}$$

where N is the number of images and L is the number of filters. The Eigenvector can be obtained according to the following equation.

$$\overline{X_{Cov}} \times \overline{X_{Cov}}^T v_{i=} \lambda_i v_i \tag{78}$$

$$\overline{X_{Cov}}^T \times \overline{X_{Cov}} (\overline{X_{Cov}}^T v_i) = \lambda_i (\overline{X_{Cov}}^T v_i) \tag{79}$$

$$v_i = \overline{X}^T v'_i, \quad s.t. \ \ v'_i = \overline{X}^T v^i \tag{80}$$

$$V_i = \frac{\overline{X_{cov}}}{N} v_i \tag{81}$$

$$V_i = normalized(V_i) \tag{82}$$

$$FI = \frac{\overline{X_{Cov}}}{N} V_i \tag{83}$$

$$U = W_{ICA} FI_{i(PCA)} \tag{84}$$

$FI_{i(PCA)}$ means whitened process and centered feature vector. Let the number of L be chosen as the Eigenvector obtained above. To apply the ICA algorithm to estimate the independent component, we apply it to ICA based on the feature vector. The first ICA centering and the whitening algorithm are applied, and then the data normalization is performed. Then, the initial weight is set randomly, and the initial weight and the data obtained from the whitening are linearly combined. There are two methods of ICA (Kurtosis, Negentropy) and Kurtosis method. Therefore, the expression is as follows. Therefore, the expression is as follows.

$$kurt(y) = Ey^4| - 3(Ey^2)^2 \tag{85}$$

The values obtained above and whitening data are linearly combined and normalized. Singular value decomposition is performed to remove the correlation. Then, the delta value is obtained by carrying out the dot product operation with the weight obtained from the singular value decomposition and the initial weight obtained randomly and subtracting the value from 1. If the delta value is greater than 0.000001, the data obtained from the final weight and whitening are linearized, and this value is called independent components. A linear combination of the obtained independent components U and the center image of the zero-padded original image are performed.

$$I^s i,l = U_i X_{padding} \tag{86}$$

s is the stage and the first stage is one. l means independent elements 1 to l. By subtracting each patch image from the above equation, we can obtain the following expression. For each input image $I_i \in R^{m\times n}$, we obtain L_1 output images $I^s_{i,l}\, l = 1,2,...L_1, I^s_{i,l} \in R^{m\times n}$ after the first stage of ICANet. We denote I^s as

$$I^s = [I^s_{1,1} \cdots I^s_{1,L_1} \cdots I^s_{N,1} \cdots I^s_{N,L_1}] \in R^{m\times NL_1 n} \tag{87}$$

3.2.2 Second Stage of ICANet2

A hashing code is generated based on the convoluted feature and the weight map. The values generated by the hashing code are also projected onto a weight map.

【Step 1】 The second patch sliding process.

Almost repeating the same process as the first stage, as shown in Figure 3.1, the second stage of ICANet1 also includes three steps: Similar to step 1, we use a patch of size $k_1 \times k_2$ to slide each pixel of the ith image $I'_{i,l} \in R^{k_1k_2\times mn}$, $l = 1,2,...,L_1$ and obtain a matrix as follows:

$$Y_{i,l} = [y_{i,l,1} \;\; y_{i,l,2} \;\; \cdots y_{i,l,mn}] \in R^{k_1k_2\times mn}, l = 1,2,...,L_1, i = 1,2,...,N \tag{88}$$

where $x_{i,j}$ denotes the jth vectorized patch in I_i. Thus, for all input training images $I_i, i = i = 1,2,...,N)$, we can obtain the following matrix.

$$Y_{i,l} = [Y_{1,l}\, Y_{2,l} \cdots X_{N,l}] \in R^{k_1 k_2 \times Nmn} \tag{89}$$

We concatenate the matrices of all the L_1 filters and obtain a matrix

$$Y = [Y_1\, Y_2 \cdots Y_{L_1}] \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)N} \tag{90}$$

【Step 2】 The second mean remove process

In this step, we subtract the patch mean from each patch and obtain the following:

$$patch\overline{X_i} = [\overline{Y}_{i,1}\overline{Y}_{i,2} \cdots \overline{Y}_{i,mn}] \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)N} \tag{91}$$

By subtracting each patch image from the above equation, we can obtain the following expression.

$$\overline{Y} = patch\,Y - patch\,\overline{Y_i} \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)} \tag{92}$$

【Step 3】 The first PCA process with ICA

In this step, the Eigenvalues and Eigenvectors of $\overline{X}$ are calculated from equation (76) using the PCA algorithm. The covariance is obtained by repeating the above equation from all patches and images

$$\overline{X_{Cov}}=\sum_{N=1}^{NL}(\overline{X}\times\overline{X}^{T}) \tag{93}$$

where N is the number of images and L is the number of filters. The Eigenvector can be obtained according to the following equation.

$$\overline{X_{Cov}}\times\overline{X_{Cov}}^{T}v_{i=}\lambda_i v_i \tag{94}$$

$$\overline{X_{Cov}}^{T}\times\overline{X_{Cov}}(\overline{X_{Cov}}^{T}v_i)=\lambda_i(\overline{X_{Cov}}^{T}v_i) \tag{95}$$

$$v_i=\overline{X}^{T}v'_i, \quad s.t.\ \ v'_i=\overline{X}^{T}v^{i} \tag{96}$$

$$V_i=\frac{\overline{X_{cov}}}{N}v_i \tag{97}$$

$$V_i=normalized(V_i) \tag{98}$$

$$FI=\frac{\overline{X_{Cov}}}{N}V_i \tag{99}$$

$$U=W_{ICA}FI_{i(PCA)} \tag{100}$$

$FI_{i(PCA)}$ means whitened process and centered feature vector. Let the number of L be chosen as the Eigenvector obtained above. To apply the ICA algorithm to estimate the independent component, we apply it to ICA based on the feature vector. The first ICA centering and the whitening algorithm are applied, and then the data normalization is performed. Then, the initial weight is set randomly, and the initial weight and the data obtained from the whitening are linearly combined. There are two methods of ICA (Kurtosis, Negentropy) and Kurtosis method. Therefore, the expression is as follows. Therefore, the expression is as follows.

$$kurt(y)=Ey^{4}|-3(Ey^{2})^{2} \tag{101}$$

The values obtained above and whitening data are linearly combined and normalized. Singular value decomposition is performed to remove the correlation. Then, the delta value is obtained by carrying out the dot product operation with the weight obtained from the singular value decomposition and the initial weight obtained randomly and subtracting the value from 1. If the delta value is greater than 0.000001, the data obtained from the final weight and whitening are linearized, and this value is called independent components. A linear combination of the obtained independent components U and the center image of the zero-padded original image are performed.

$$I^s i,l = U_i X_{padding} \tag{102}$$

s is the stage and the first stage is one. l means independent elements 1 to l. By subtracting each patch image from the above equation, we can obtain the following expression. For each input image $I_i \in R^{m \times n}$, we obtain L_1 output images $I^s_{i,l} l = 1,2,...L_1, I^s_{i,l} \in R^{m \times n}$ after the first stage of ICANet. We denote I^s as

$$I^s = [I^s_{1,1} \cdots I^s_{1,L_1} \cdots I^s_{N,1} \cdots I^s_{N,L_1}] \in R^{m \times NL_1 n} \tag{103}$$

3.2.3 Output Stage of ICANet2

Step 1 to Step 3 is the same as the previous PCANet algorithm. The following lists the advantages of ICANet. There are PCANet algorithms, but papers that have been studied with various structures of ICANet are a new challenge.

【Step 1】 Binary quantization.
In this step, we binarize the outputs of the second stage of PCANet, and obtain

$$P_{i,l,\iota} = H(I^{ss}_{i,l,\iota}), l = 1,2,...,L1;\ \iota = 1,2,...,L2;\ i - 1,2,...,N \tag{104}$$

【Step 2】 Weight and sum.
Around each pixel, we view the vector of L_2 binary bits as a decimal number. This converts the binary images $P_{i,l,\iota}$ back into

integer-valued images as follows:

$$T_{i,l} = \sum_{\iota=1}^{L2} 2^{\iota-1} P_{i,l,\iota} \tag{105}$$

We denote T as

$$T = [T_{1,1} \cdots T_{1,L_1} \cdots T_{N,1} \cdots T_{N,L_1}] \in R^{m \times NL_1 n} \tag{106}$$

【Step 3】 Block sliding.

We use a block of size $h_1 \times h_2$ to slide each of the L_1 images $T_{i,l}, l = 1, \ldots, L_1$, with overlap ratio R, and then reshape each $h_1 \times h_2$ matrix into a column vector, which is then concatenated to obtain a matrix

$$Z_{i,l} = [z_{i,l,1} \cdots z_{1,l,2} \cdots z_{i,lB}] \in R^{h_1 h_2 \times B} \tag{107}$$

We denote Z as

$$Z_i = [z_{1,1} \cdots z_{1,B} \cdots z_{N,1} \cdots z_{N,B}] \in R^{h_1 h_2 \times L_1 BN} \tag{108}$$

【Step 4】 Histogramming

We compute the histogram (with 2^{L_2} bins) of the decimal values in each column of Z_i and concatenate all the histograms into one vector and obtain

$$f_i = [Hist(z_{i,1,1}) \cdots Hist(z_{i,1,B}) \cdots Hist(z_{i,L_1,1}) \cdots Hist(z_{i,L_1,B})] \in R^{(2^{L_2})L_1 B} \tag{109}$$

which is the “feature” of the input image I_i and Hist(.) denotes the histogram operation. We denote f as

$$f_i = [f_1 \cdots f_N) \in R^{(2^{L_2})L_1BN} \qquad (110)$$

The feature vector is then sent to a classifier, for example, support vector machine (SVM), Euclidean distance, etc.

ICANet2 is also called a factorial ICANet, and a feature vector is obtained through the intensive multiplication of the center matrix and the Eigenvector. Unlike ICANet1, the Eigenvectors are linearly multiplied by the covariance of the patch center image. Finally, the feature values are extracted through the linear multiplication of the sum of the patch images and the normalized Eigenvectors. This is different from ICANet1. ICANet2 has the same number of features as ICANet1. Figure 3.11 shows the first stage of ICANet2 and figure 3.12 shows feature CU-ECG database in the first stage of ICANet2. Figure 3.13 indicates second and output stage of ICANet2 and figure 3.14 shows feature CU-ECG database in second and output stage of ICANet2.

The detailed description of the workflow is below:

(1) Perform PCA step like ICANet1.

(2) Patch sliding of all images is performed, and the average image and the Eigen vector obtained from the PCA are convoluted.

(3) The Eigenvectors are updated through the normalization step using the Eigenvectors, and then the convolution is performed using the average image of all the images and the updated Eigenvectors. This is called a feature vector.

(4) The center image is obtained from the basic data, and the zero padded image and the independent Eigen feature vector obtained in (2) are convoluted and the ICA convolution output value is extracted.

Steps (5) and (6) are the same as ICANet1.

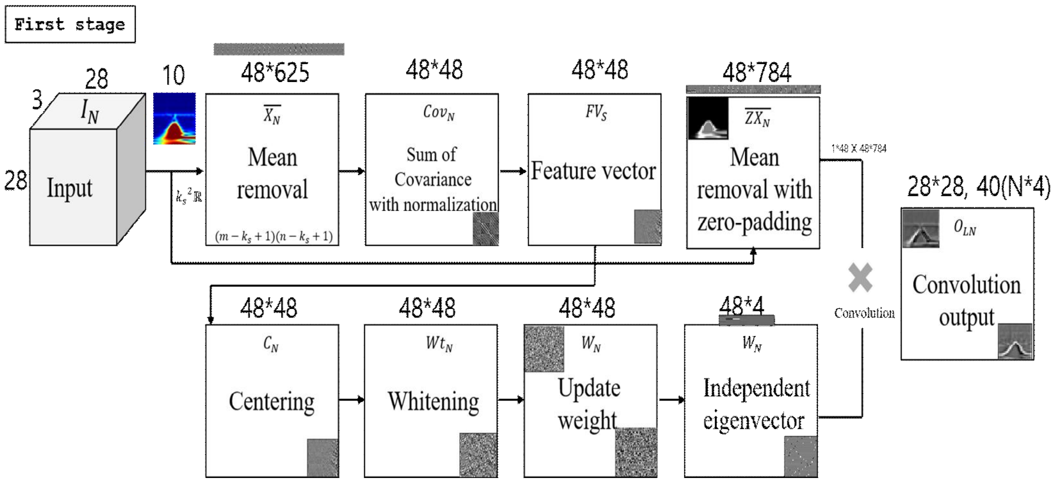


Figure 3.11. First stage of ICANet2.

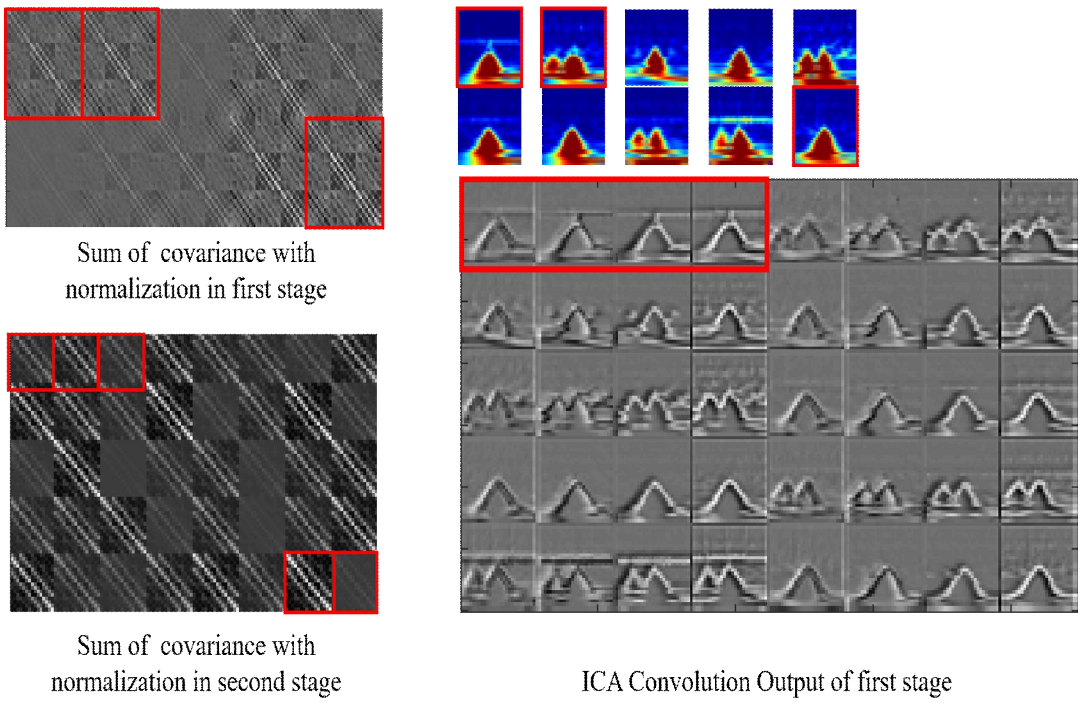


Figure 3.12. Feature CU-ECG database in first stage of ICANet2.

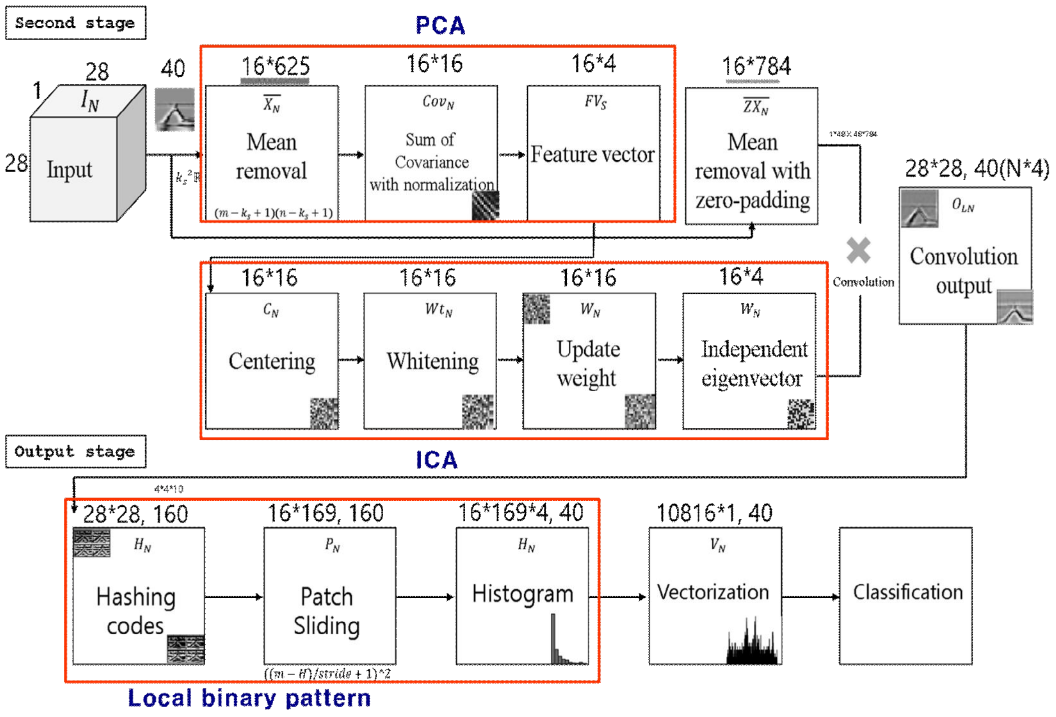


Figure 3.13. Second and output stage of ICANet2.

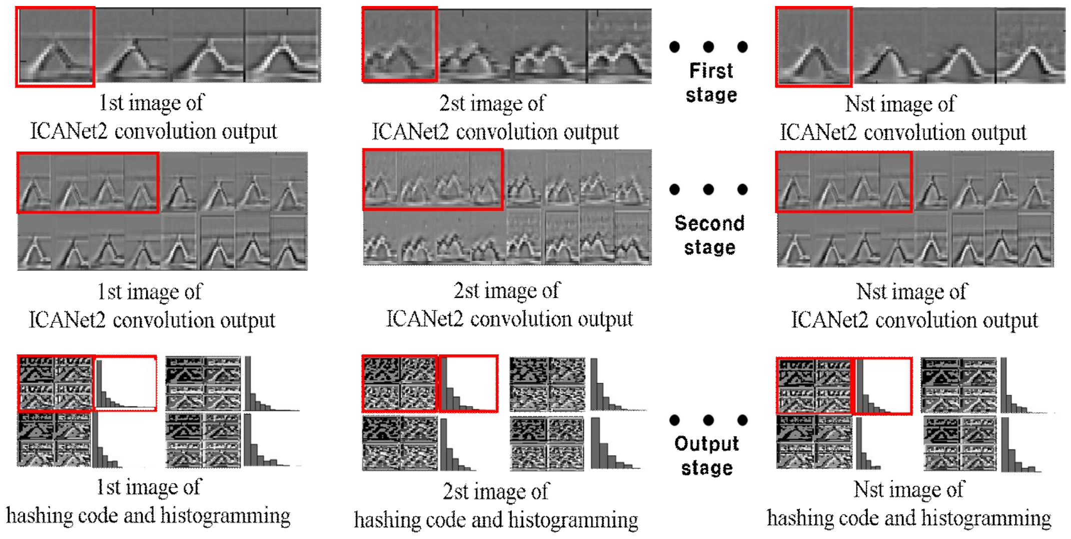

1st image of ICANet2 convolution output
2st image of ICANet2 convolution output
First stage
Nst image of ICANet2 convolution output
1st image of ICANet2 convolution output
2st image of ICANet2 convolution output
Second stage
Nst image of ICANet2 convolution output
1st image of hashing code and histogramming
2st image of hashing code and histogramming
Output stage
Nst image of hashing code and histogramming

Figure 3.14. Feature CU-ECG database in second and output stage of ICANet2.

```
Algorithm 4 ICANet2
1. Reading: Image data or signal data

2. Initialization of variable

3. Stage 1 of ICANet2
   - Patch mean-removal : $\overline{Y} = patch\,Y - patch\,\overline{Y_i} \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)}$

   - Compute Eigenvector using PCA:
         $v_i = \overline{Y}^T v'_i, \quad s.t. \; v'_i = \overline{Y}^T v^i$

   - Compute PCA feature vector using PCA:
         $V_i = \frac{\overline{X_{cov}}}{N} v_i \; , \; V_i = normalized(V_i), \quad FI = \frac{\overline{X_{Cov}}}{N} V_i$

   - Compute convolution using ICA:
         $U = W_{ICA} FI_{i(PCA)}$

   - Output : $I^s i, l = UX_{padding}$

4. Stage 2 of ICANet2
   - Patch mean-removal: $\overline{Y} = patch\,Y - patch\,\overline{Y_i} \in R^{k_s^2 \times (m-k_s+1)(n-k_s+1)}$

   - Compute PCA feature vector using PCA:
         $V_i = \frac{\overline{X_{cov}}}{N} v_i \; , \; V_i = normalized(V_i), \quad FI = \frac{\overline{X_{Cov}}}{N} V_i$

   - Compute convolution using ICA:
         $U = W_{ICA} FI_{i(PCA)}$

   - Output : $I^s i, l = UX_{padding}$

5. Output layer
   Binary hashing: compute the decimal-valued image
         $P_{i,l,\iota} = H(I^{ss}_{i,l,\iota}), l = 1,2,...,L1; \; \iota = 1,2,...,L2; \; i - 1,2,...,N$

   Histogram:
   $f_i = [Hist(z_{i,1,1}) \cdots Hist(z_{i,1,B}) \cdots Hist(z_{i,L_1,1}) \cdots Hist(z_{i,L_1,B})] \in R^{(2^{L_2})L_1 B}$
   Output : $f_i = [f_1 \cdots f_N) \in R^{(2^{L_2})L_1 BN}$
```

Ⅳ. Experiment and Results

This section describes the data acquisition process and environment, evaluates the data, and examines the similarities. In addition, we present the performance assessment and the effectiveness of the ICANet.

4.1 Database

In this paper, we use databases obtained from various systems to determine the suitability of the algorithm.

4.1.1 CU-ECG database

The CU-ECG DB is the data from Chosun University, Korea. The data contains 100 people. The measurement time is 10 s for one measurement; 60 times was measured during three days for each individual. Measurements were made when the participants were sitting in the chair in a relaxed state with the data-sampling rate of 500,000 Hz. All acquired ECG are Lead1 type taken with wet electrodes. We used a processor, an amplifier, a band-pass filter, and a low-pass filter as the primary board and the sensor. The processor is Atmega8; Analog to Digital (AD) converter uses 10-bit resolution. For communication, we use USB to serial. The gain of the amplifier was 1000 times, and the signal was measured using a 5-V positive power source. To acquire ECG data, the patient is seated in the chair in a relaxed state. The patch shown in the picture below is attached to the arm and the data is acquired. The acquired data is stored in the computer using universal synchronous/asynchronous receiver/transmitter (USART) communication. Figure 4.1 shows a data acquisition environment for ECG biometrics and figure 4.2 indicates mainboard and experiment environment. Figure 4.3 shows preprocessing using CU-ECG database and figure 4.4 indicates samples of CU-ECG database with noise CU-ECG. Figure 4.5 shows samples of CU-ECG database with size scale. Noise (White Gaussian) was used to determine the robustness of ICA. In addition, additive white Gaussian noise (AWGN) is a basic noise model used in Information theory to mimic the effect of many random processes that occur in nature.

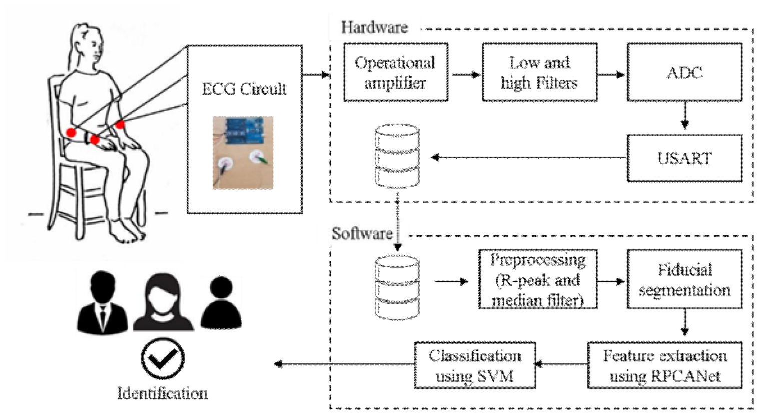


Figure 4.1. Data acquisition environment for ECG biometrics

Figure 4.2. Mainboard and experiment environment

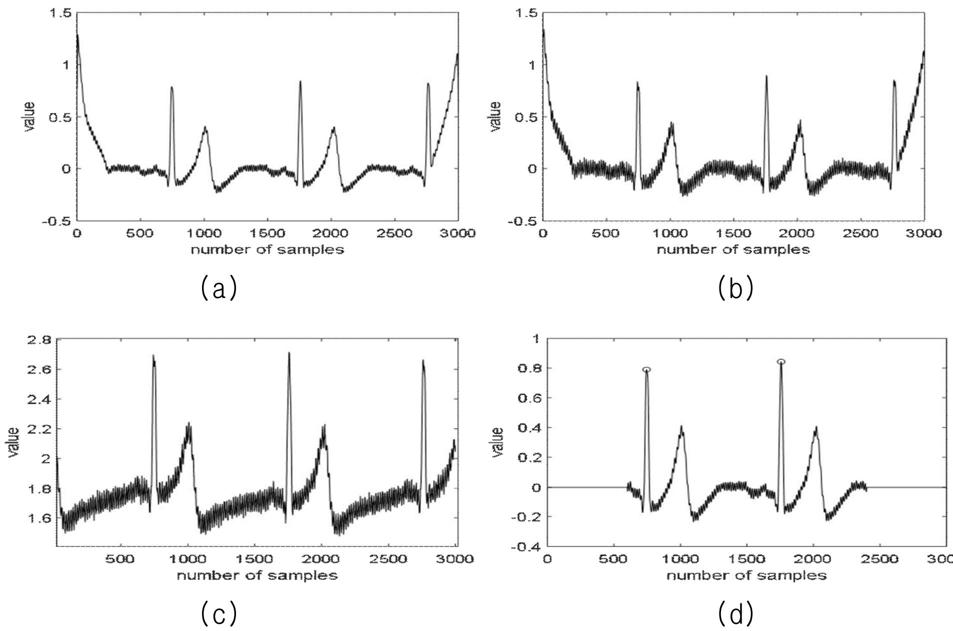
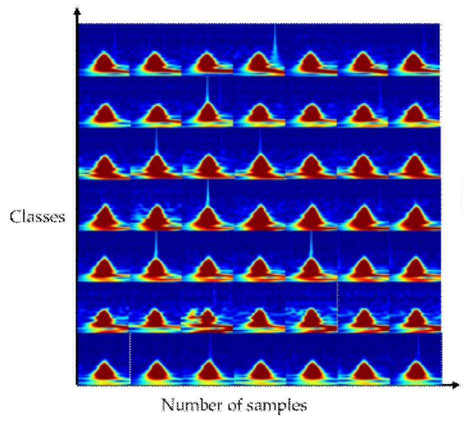
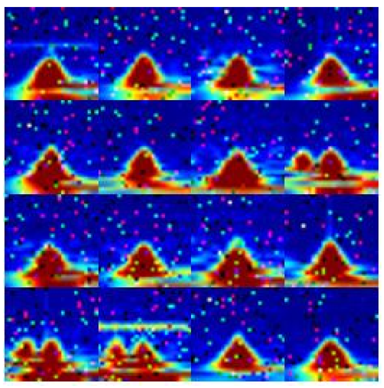


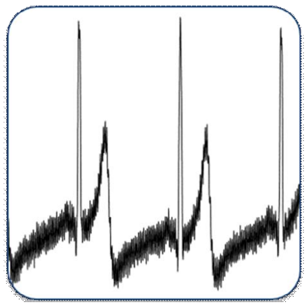
Figure 4.3. Preprocessing using CU ECG database: (a) raw signal; (b) mean variance; (c) spike removal; (d) R peak detection.

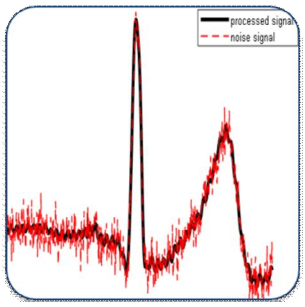


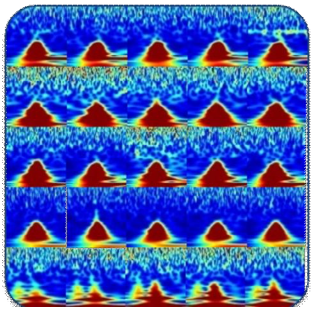
(a)



(b)

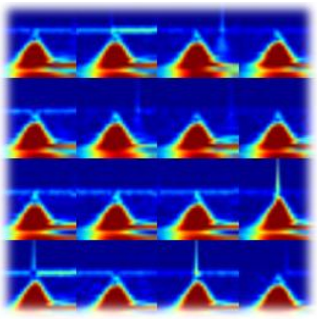


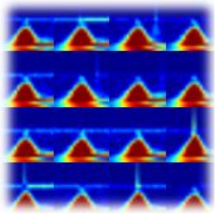


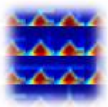


(c) (d) (e)

Figure 4.4. Samples of CU-ECG database with noise CU-ECG. (a) ECG scalogram. (b) salt noise with ECG scalogram. (c) 1D raw-ECG signal. (d) noise ECG signal. (e) Gaussian noise with ECG scalogram







(a) (b) (c)

Figure 4.5. Samples of CU-ECG database with size scale. (a) 28 × 28. (b) 20 × 20. (c) 12 × 12

4.1.2 FERET face database

FERET DB is configured for the development of automatic face recognition capabilities. It is a face image of 14051 8-bit grayscales. It also includes facial images of various poses and presents the outlines of alternative facial expressions and other illuminations. For some people, they include face images with glasses and other hair lengths. Nevertheless, this DB is one of the most known face databases since it contains many face images of many people. Images of an individual were acquired in sets of 5 to 11 images, collected under relatively unconstrained conditions. Two frontal views were taken (Fa and Fb); a different facial expression was requested for the second frontal image. For 200 sets of images, a third frontal image was taken with a different camera and different lighting (this is referred to as the Fc image). Figure 4.6 shows example of different categories of FERET images. Table 4.1 shows data information of FERET.

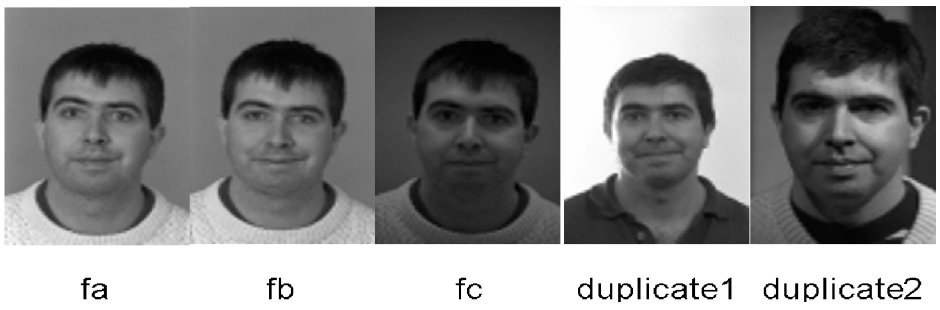


Figure 4.6. Example of different categories of FERET images. The duplicate 1 image was taken within one year of the Fa image and the duplicate 2 and Fc images were taken at least one year apart.

Table 4.1. Data information of FERET

Features / Database	Data Size	Number of data per class
Fa	13500 X 1196	1 (1196 class)
Fb	13500 X 1195	1 (1196 class)
Fc	13500 X 194	1 (1196 class)
Duplicate1	13500 X 722	from 2 to 10 (243 class)
Duplicate2	13500 X 234	from 2 to 10 (75 class)

4.1.3 CU-Face Database

Deep learning is generally known for algorithms that use big data. Therefore, the correlation between ICANet and small data can be confirmed by applying the CU-Face database that is small data. The CU-Face data is data acquired from Chosun University. The total number of participants is five, and the training data is composed of 10 pieces per person and the testing data is composed of 80 pieces of data. Therefore, verification data is 8 times more than learning. Moreover, the size of the learning data is 1800 * 50, and the size of the verification data is 1800 * 400. In order to show the robustness of ICANet, we also experimented with data mixed with noise (Gaussian noise). Figure 4.7 shows example of CU-Face data with noise.

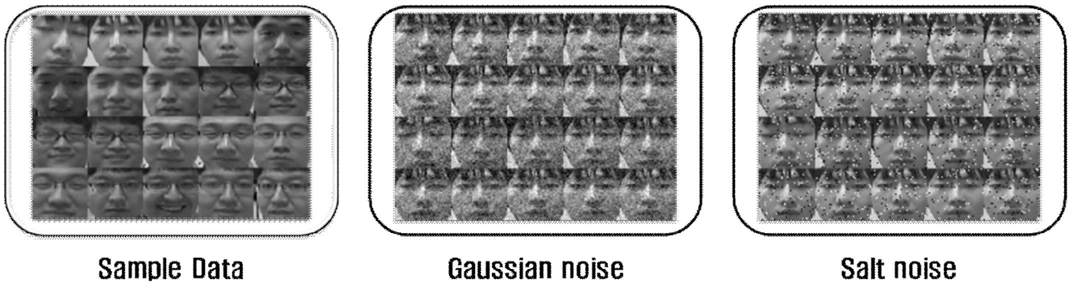


Figure 4.7. Example of CU-Face data with noise

4.2 Experimental Results and Discussion

In this section, we show the classifier used for experiments and the experimental results.

4.2.1 Classifier

Distance measurement is a method of calculating the distance between two observations. In this paper, we show performance using (Euclidean, squared Euclidean, city-block, Minkowski, cosine, correlation, and Spearman) distance and SVM. Table 4.2 shows mathematical expression of classier.

Table 4.2. Mathematical expression of classier

Classifier	Mathematical expression
Euclidean distance (ED)	$d_{st}^2 = (x_s - y_t)(x_s - y_t)'$
Squared Euclidean distance (SED)	$d_{st}^2 = (x_s - y_t)^2 (x_s - y_t)^2$
City-block distance (CBD)	$d_{st} = \sum_{j=1}^{n} \lvert x_{sj} - y_{tj} \rvert$
Minkowski distance (MKD)	$d_{st} = \sqrt[p]{\sum_{j=1}^{n} \lvert x_{sj} - y_{tj} \rvert^P}$
Cosine distance (CSD)	$d_{st} = (1 - \frac{x_s y' t}{\sqrt{x_s x' s)(y_t y' t)}})$
Correlation distance (CLD)	$d_{st} = (1 - \frac{(x_s - \overline{x_s})(y_t - \overline{y}_t)'}{\sqrt{(x_s - \overline{x}_s)(x_s - \overline{x}_s)'}\sqrt{(y_t - \overline{y}_t)(y_t - \overline{y}_t)'}}$ $s.t.\ \overline{x_s} = \frac{1}{n}\sum_j x_{sj}$
Spearman distance (SMD)	$d_{st} = 1 - \frac{(r_s - \overline{r_s})(r_t - \overline{r}_t)'}{\sqrt{(r_s - \overline{r}_s)(r_s - \overline{r}_s)'}\sqrt{(r_t - \overline{r}_t)(r_t - \overline{r}_t)'}})$ $s.t.\ \overline{r_s} = \frac{1}{n}\sum_j r_{sj} = \frac{(n+1)}{2}$

4.2.2 Performance evaluation

TP is the number of correct predictions for the positive samples, TN is the number of correct predictions for the negative samples, FN is the number of incorrect predictions for the positive samples, and FP is the number of incorrect predictions for the negative samples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{111}$$

4.2.3 Setting parameters

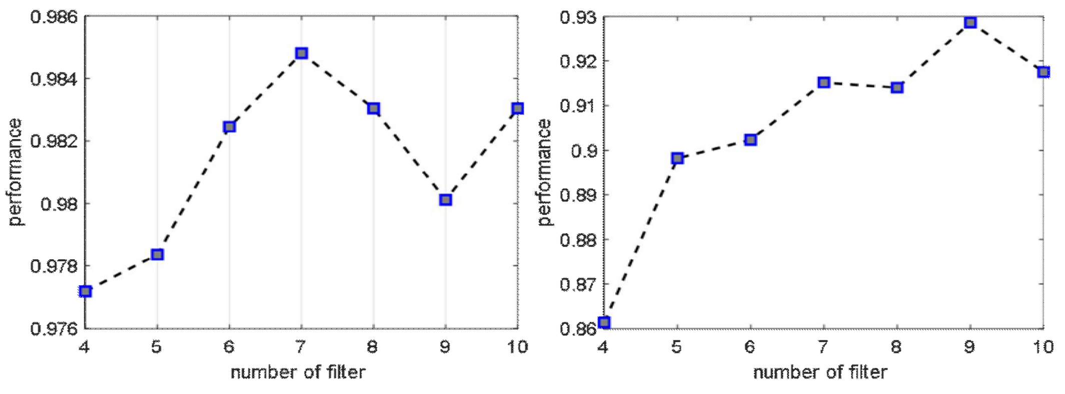
FERET data based parameters are set from three to five in terms of L, K is set to two to eight, and H is set to 4 or 8 because of memory problem.

4.2.4 Experimental results and performance analysis

To show the performance of the proposed algorithm, performance is shown through various classifiers. The feature extractor is PCANet, ICANet1 to ICANet2. The classifiers are the Euclidean distance, the squared Euclidean distance, the city-block distance, the Minkowski distance, the cosine distance, the correlation distance, the spearman distance, SVM was used. Performance analysis using FERET data showed similar performance to PCANet when using Fb data. However, as the data was changed to Fc, Dup1, and Dup2, it showed higher performance in ICANet1 and 2 than PCANet. If the training data and the testing data are similar, the PCANet can have a good effect, but if the non-similar of real-life data is used, the ICANet algorithm is robust. Especially, when the Dup2 and noise ECG data is used, the robustness of the ICANet2 algorithm is verified. In addition, the classifier has been confirmed that the SVM algorithm works well. In addition, verification time is important for ECG personal authentication. The verification time of PCANet was about 0.2 seconds, ICANet1 was 0.2 seconds, similar to PCANet, and ICANet2 was 0.15 seconds. Therefore, the effectiveness of the ICANet algorithm for personal authentication has been demonstrated in terms of classification time.

● Influence on the number of filters

The number of filters is an important factor for the recognition performance in the convolution layer on ICANet, because there is the amount of information increased when the number of filters in the first stage gradually increased. Figure 4.8 shows the influence on the number of filters in ICANet.

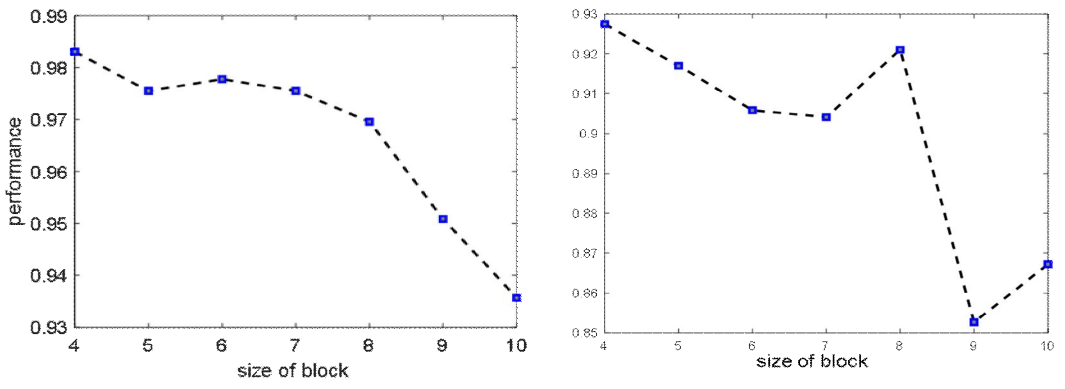


(a) (b)

Figure 4.8. The influence on the number of filters in ICANet. (a) CU-ECG database. (b) noise CU-ECG database.

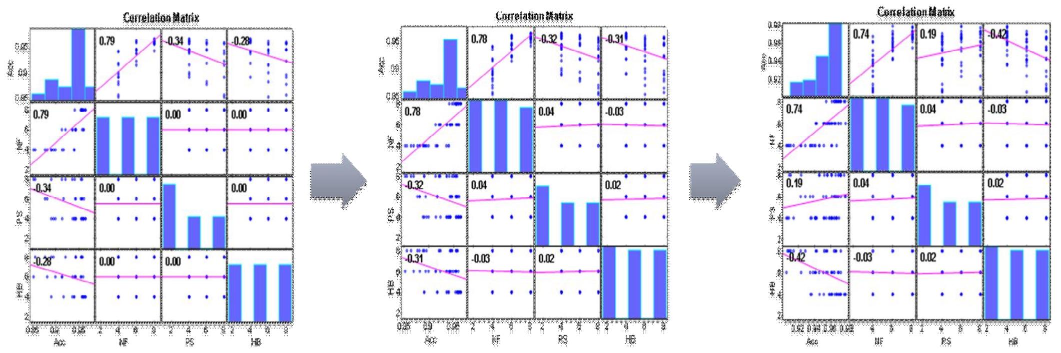
● Influence on block size

Compared to the number of filters, the performance tends to decrease as the block size increases. As the block size increases, the spacing of the strides becomes wider, resulting in information loss of data and degradation of performance. Figure 4.9 shows the influence on the block size in ICANet and figure 4.10 shows correlation of PCANet, ICANet1, and ICANet2.



(a) (b)

Figure 4.9. The influence on the block size in ICANet. (a) CU-ECG database. (b) noise CU-ECG database.



(a) (b) ©

Figure 4.10. Correlation of PCANet, ICANet1, and ICANet2. (a) PCANet1. (b) ICANet1. (c) ICANet2

To illustrate ICANet's feature distribution, Figure 4.11 shows a distribution chart of feature in ICANet. Figure 4.12 shows performances of CU-ECG using SVM and figure 4.13 performances of Gaussian noise CU-ECG using SVM. When using CU-ECG data, it is similar to PCANet's performance, but ICANet's robustness is proved when using noise mixed with white Gaussian (signal to noise ratio = 30). Figure 4.14 to 4.17 shows a comparison of face recognition rates on various methods on FERET (Fb, Fc, Dup1, and Dup2) database. In the case of Fb data, the performance of PCANet 's algorithm was high, but the performance of ICANet was higher than PCANet as Fc and Dup1 and Dup2 data. The reason is that the performance of ICA is improved by removing the correlation between the original vector, feature vector, and covariance through the independent component analysis of ICA. Figure 4.18 shows performance of CU-Face database and figure 4.19 indicates performance of CU-ECG. In addition, Figure 4.20 shows performance of CU-Face and figure 4.21 presents performance of FERET. In addition to figure 4.22 shows performance of CU-ECG and figure 4.23 indicates performance of scale CU-ECG.

As shown in Figure 4.22, the processed data shows similar performance to PCANet. However, the PCANet using Eigenvectors is weak against noise, and the feature vector using statistical independence is robust against noise. In addition, as shown in Figure 4.23, ICANet's classification rate did not significantly decrease compared to PCANet even if information of data was lost. In addition, as shown in Figure 4.19, the performance was higher than the basic deep learning of auto-encoder and the classification time was about 0.05 seconds faster than PCANet. In addition, as shown in Figure 4.21, high performance was expected in face data as well as electrocardiogram data, and it is shown that it is better than PCANet in small data as shown in Figure 4.21. As a result of performance, ICANet showed higher performance than PCANet when face data and ECG data were used. Especially, it showed stronger than PCANet in noise data. Table 4.3 to 4.8 shows performance of (PCANet, ICANet1, ICANet2) using CU-ECG and noise CU-ECG database. Table 4.9 to 4.24 shows Performance of (PCANet, ICANet1, ICANet2) using FERET database (Fb, Fc, Dup1, Dup2).

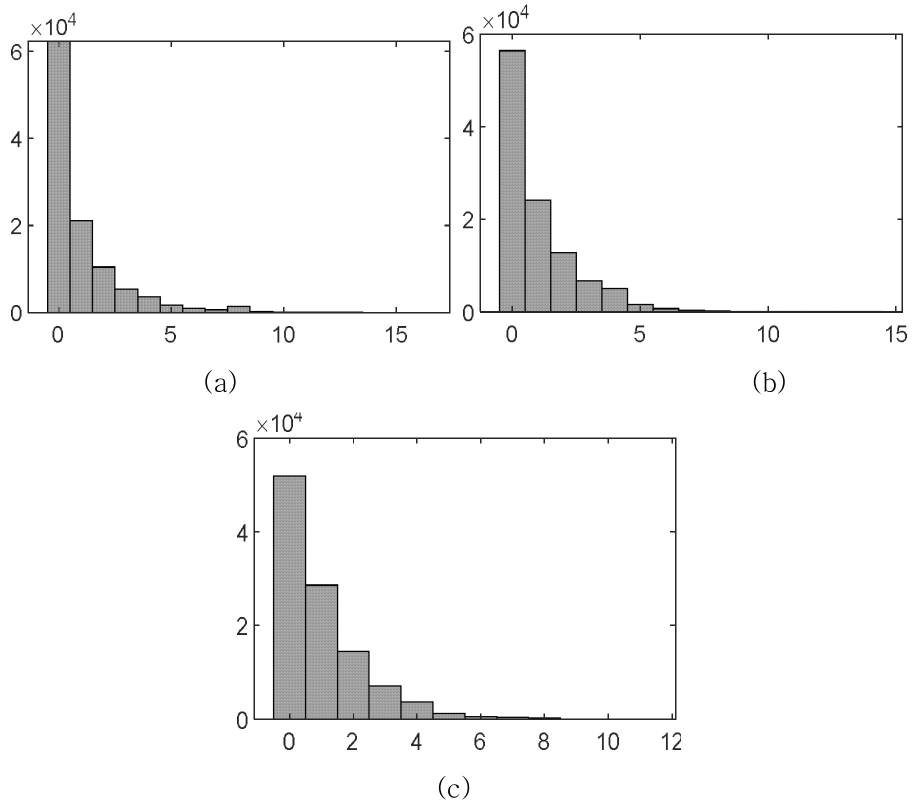


Figure 4.11. Distribution chart of feature in ICANet. (a) ICANet1. (b) ICANet2.

Table 4.3. Performance of PCANet using CU-ECG database, R=0.5

Parameters			Performance	Parameters			Performance
L	K	H		L	K	H	
8	4	10	0.9760	10	4	8	**0.9748**
9	4	4	0.9830	10	4	9	0.9795
9	4	5	0.9813	10	4	10	0.9754
9	4	6	0.9819	4	5	4	0.9825
9	4	7	0.9836	4	5	5	0.9743
9	4	8	0.9830	4	5	6	0.9772
9	4	9	0.9737	4	5	7	0.9772
9	4	10	0.9801	4	5	8	0.9784
10	4	4	0.9825	4	5	9	0.9550
10	4	5	0.9819	4	5	10	0.9585
10	4	6	0.9807	5	5	4	0.9801
10	4	7	0.9807	5	5	5	0.9784

Table 4.4. Performance of ICANet1 using CU-ECG database, R=0.5

Parameters			Performance	Parameters			Performance
L	K	H		L	K	H	
8	7	4	**0.9848**	4	9	4	**0.9848**
8	7	6	0.9825	4	9	6	0.9772
8	7	8	0.9836	4	9	8	0.9684
8	7	10	0.9807	4	9	10	0.9480
8	7	12	0.9801	4	9	12	0.9456
10	7	4	0.9830	6	9	4	0.9836
10	7	6	0.9801	6	9	6	0.9813
10	7	8	0.9789	6	9	8	0.9801
10	7	10	0.9801	6	9	10	0.9708
10	7	12	0.9789	6	9	12	0.9754
12	7	4	0.9801	8	9	4	0.9813
12	7	6	0.9825	8	9	6	0.9836

Table 4.5. Performance of ICANet2 using CU-ECG database, R=0.5

Parameters			Performance	Parameters			Performance
L	K	H		L	K	H	
8	4	10	0.9760	10	4	8	**0.9848**
9	4	4	0.9830	10	4	9	0.9795
9	4	5	0.9813	10	4	10	0.9754
9	4	6	0.9819	4	5	4	0.9825
9	4	7	0.9836	4	5	5	0.9743
9	4	8	0.9830	4	5	6	0.9772
9	4	9	0.9737	4	5	7	0.9772
9	4	10	0.9801	4	5	8	0.9784
10	4	4	0.9825	4	5	9	0.9550
10	4	5	0.9819	4	5	10	0.9585
10	4	6	0.9807	5	5	4	0.9801
10	4	7	0.9807	5	5	5	0.9784

Table 4.6. Performance of PCANet using Gaussian noise CU-ECG database, R=0.5

Parameters			Performance	Parameters			Performance
L	K	H		L	K	H	
4	6	4	0.8643	5	6	9	0.7848
4	6	5	0.8368	5	6	10	0.8023
4	6	6	0.8304	6	6	4	0.9012
4	6	7	0.8187	6	6	5	0.8789
4	6	8	0.8316	6	6	6	0.8871
4	6	9	0.7269	6	6	7	0.8830
4	6	10	0.7351	6	6	8	0.8930
5	6	4	0.8942	6	6	9	0.8105
5	6	5	0.8667	6	6	10	0.8345
5	6	6	0.8690	7	6	4	**0.9094**
5	6	7	0.8596	7	6	5	0.8830
5	6	8	0.8637	7	6	6	0.8930

Table 4.7. Performance of ICANet1 using Gaussian noise CU-ECG database, R=0.5

Parameters			Performance	Parameters			Performance
L	K	H		L	K	H	
6	4	12	0.7327	12	8	4	**0.9351**
7	4	4	0.9275	12	8	8	0.9339
7	4	5	0.9170	12	8	12	0.9058
7	4	6	0.9058	4	12	4	0.8386
7	4	7	0.9041	4	12	8	0.8281
7	4	8	0.9211	4	12	12	0.7281
7	4	9	0.8526	8	12	4	0.9345
7	4	10	0.8673	8	12	8	0.9035
7	4	11	0.7468	8	12	12	0.8164
7	4	12	0.8018	12	12	4	0.9316
8	4	4	0.9187	12	12	8	0.9327
8	4	5	0.9170	12	12	12	0.9006

Table 4.8. Performance of ICANet2 using Gaussian noise CU-ECG database, R=0.5

Parameters			Performance	Parameters			Performance
L	K	H		L	K	H	
4	4	8	0.8363	8	8	6	**0.9363**
4	4	12	0.8070	8	8	7	0.9257
4	4	16	0.7632	8	8	8	0.9123
8	4	4	0.9234	8	8	9	0.9082
8	4	8	0.9082	8	8	10	0.9140
8	4	12	0.8959	9	8	4	0.9322
8	4	16	0.8965	9	8	5	0.9345
12	4	4	0.9281	9	8	6	0.9211
12	4	8	0.9117	9	8	7	0.9146
12	4	12	0.9105	9	8	8	0.9175
12	4	16	0.9129	9	8	9	0.9135
4	8	4	0.8994	9	8	10	0.9199

Table 4. 9. Performance of PCANet using FERET database (Fb), R=0.5

ED	SED	CBD	MKD	CSD	CLD	SMD	SVM	L	K	H
0.8686	0.8686	0.8586	0.8686	0.9289	0.9297	0.9297	0.9054	3	4	4
0.9690	0.9690	0.9615	0.9690	0.9816	0.9816	0.9808	0.9649	3	4	8
0.8117	0.8117	0.8000	0.8117	0.9364	0.9356	0.9448	0.9665	4	4	4
0.9556	0.9556	0.9356	0.9556	0.9874	0.9883	0.9874	0.9732	4	4	8
0.8887	0.8887	0.8828	0.8887	0.9464	0.9473	0.9456	0.9741	3	6	4
0.9690	0.9690	0.9623	0.9690	0.9849	0.9849	0.9791	0.9766	3	6	8
0.8653	0.8653	0.8619	0.8653	0.9490	0.9490	0.9531	0.9766	4	6	4
0.9598	0.9598	0.9623	0.9598	0.9841	0.9858	0.9858	0.9808	4	6	8
0.9029	0.9029	0.8937	0.9029	0.9531	0.9548	0.9531	0.9816	3	8	4
0.9640	0.9640	0.9640	0.9640	0.9791	0.9791	0.9791	0.9824	3	8	8
0.8820	0.8820	0.8711	0.8820	0.9640	0.9640	0.9623	0.9824	4	8	4
0.9649	0.9649	0.9649	0.9649	0.9883	0.9874	0.9883	0.9841	4	8	8

Table 4.10. Performance of PCANet using FERET database (Fc), R=0.5

ED	SED	CBD	MKD	CSD	CLD	SMD	SVM
0.8918	0.8918	0.8711	0.8918	0.9536	0.9588	0.9536	0.9381
0.9794	0.9794	0.9639	0.9794	0.9845	0.9845	0.9845	0.9794
0.8505	0.8505	0.8402	0.8505	0.9588	0.9588	0.9639	0.9794
0.9742	0.9742	0.9639	0.9742	0.9897	0.9897	0.9897	0.9794
0.8814	0.8814	0.8763	0.8814	0.9485	0.9485	0.9485	0.9794
0.9433	0.9433	0.9330	0.9433	0.9742	0.9691	0.9742	0.9794
0.8660	0.8660	0.8608	0.8660	0.9639	0.9639	0.9588	0.9845
0.9691	0.9691	0.9639	0.9691	0.9845	0.9845	0.9845	0.9845
0.8608	0.8608	0.8454	0.8608	0.9330	0.9330	0.9330	0.9845
0.9227	0.9227	0.9021	0.9227	0.9639	0.9639	0.9588	0.9845
0.8505	0.8505	0.8454	0.8505	0.9639	0.9639	0.9639	0.9845
0.9278	0.9278	0.9227	0.9278	0.9742	0.9742	0.9794	0.9845

Table 4.11. Performance of PCANet using FERET database (Dup1), R=0.5

ED	SED	CBD	MKD	CSD	CLD	SMD	SVM
0.7382	0.7382	0.7175	0.7382	0.8587	0.8560	0.8587	0.8393
0.8421	0.8421	0.8338	0.8421	0.9086	0.9072	0.8934	0.8975
0.6274	0.6274	0.6011	0.6274	0.8837	0.8837	0.8726	0.9072
0.7895	0.7895	0.7452	0.7895	0.9349	0.9349	0.9363	0.9307
0.7271	0.7271	0.7161	0.7271	0.8615	0.8601	0.8587	0.9321
0.8407	0.8407	0.8199	0.8407	0.9017	0.9017	0.9030	0.9349
0.7050	0.7050	0.6856	0.7050	0.8837	0.8837	0.8767	0.9377
0.8089	0.8089	0.8006	0.8089	0.9224	0.9224	0.9141	0.9391
0.7105	0.7105	0.6994	0.7105	0.8518	0.8518	0.8504	0.9391
0.8047	0.8047	0.7950	0.8047	0.8809	0.8837	0.8864	0.9391
0.6842	0.6842	0.6704	0.6842	0.8767	0.8753	0.8740	0.9391
0.7936	0.7936	0.7687	0.7936	0.9141	0.9155	0.9086	0.9391

Table 4.12. Performance of PCANet using FERET database (Dup2), R=0.5

ED	SED	CBD	MKD	CSD	CLD	SMD	SVM
0.7137	0.7137	0.6838	0.7137	0.7991	0.7991	0.7991	0.7906
0.8077	0.8077	0.7949	0.8077	0.8761	0.8718	0.8462	0.8462
0.5812	0.5812	0.5470	0.5812	0.8376	0.8376	0.8333	0.8547
0.7521	0.7521	0.7094	0.7521	0.9017	0.9017	0.9103	0.8974
0.6923	0.6923	0.6752	0.6923	0.8120	0.8120	0.8120	0.9017
0.8120	0.8120	0.7906	0.8120	0.8590	0.8590	0.8547	0.9060
0.6581	0.6581	0.6624	0.6581	0.8376	0.8376	0.8333	0.9145
0.7821	0.7821	0.7521	0.7821	0.8803	0.8803	0.8675	0.9145
0.6709	0.6709	0.6581	0.6709	0.8248	0.8248	0.8205	0.9145
0.7991	0.7991	0.7778	0.7991	0.8419	0.8376	0.8504	0.9145
0.6538	0.6538	0.6410	0.6538	0.8419	0.8419	0.8419	0.9145
0.7821	0.7821	0.7650	0.7821	0.8889	0.8932	0.8889	0.9145

Table 4.13. Performance of ICANet1 using FERET database (Fb), R=0.5

ED	SED	CBD	MKD	CSD	CLD	SMD	SVM
0.7799	0.7799	0.7791	0.7799	0.9013	0.9013	0.9004	0.8762
0.9582	0.9582	0.9448	0.9582	0.9724	0.9732	0.9724	0.9456
0.7046	0.7046	0.6862	0.7046	0.9280	0.9297	0.9389	0.9490
0.9481	0.9481	0.9406	0.9481	0.9816	0.9816	0.9824	0.9682
0.7331	0.7331	0.7272	0.7331	0.9322	0.9314	0.9414	0.9682
0.9456	0.9456	0.9297	0.9456	0.9824	0.9824	0.9841	0.9782
0.8477	0.8477	0.8326	0.8477	0.9230	0.9238	0.9247	0.9782
0.9607	0.9607	0.9506	0.9607	0.9799	0.9799	0.9799	0.9782
0.7674	0.7674	0.7715	0.7674	0.9381	0.9381	0.9381	0.9782
0.9540	0.9540	0.9448	0.9540	0.9808	0.9808	0.9808	0.9782
0.8100	0.8100	0.7941	0.8100	0.9456	0.9456	0.9490	0.9782
0.9331	0.9331	0.9222	0.9331	0.9824	0.9824	0.9849	0.9782
0.8393	0.8393	0.8360	0.8393	0.9397	0.9397	0.9372	0.9782
0.9556	0.9556	0.9464	0.9556	0.9766	0.9757	0.9732	0.9782
0.8460	0.8460	0.8469	0.8460	0.9423	0.9431	0.9456	0.9782
0.9389	0.9389	0.9339	0.9389	0.9808	0.9808	0.9791	0.9791
0.7657	0.7657	0.7615	0.7657	0.9456	0.9456	0.9473	0.9791
0.9238	0.9238	0.9205	0.9238	0.9816	0.9816	0.9833	0.9791

Table 4.14. Performance of ICANet1 using FERET database (Fc), R=0.5

ED	SED	CBD	MKD	CSD	CLD	SMD	SVM
0.8247	0.8247	0.8196	0.8247	0.9536	0.9536	0.9485	0.9227
0.9742	0.9742	0.9485	0.9742	0.9845	0.9845	0.9845	0.9588
0.7680	0.7680	0.7680	0.7680	0.9536	0.9485	0.9691	0.9691
0.9948	0.9948	0.9948	0.9948	0.9948	0.9948	0.9948	0.9845
0.6804	0.6804	0.6598	0.6804	0.9742	0.9742	0.9742	0.9845
0.9381	0.9381	0.8866	0.9381	0.9948	0.9948	0.9948	0.9897
0.8866	0.8866	0.8557	0.8866	0.9381	0.9330	0.9227	0.9897
0.9794	0.9794	0.9845	0.9794	0.9948	0.9948	0.9948	0.9948
0.8711	0.8711	0.8505	0.8711	0.9742	0.9691	0.9691	0.9948
0.9536	0.9536	0.9381	0.9536	0.9845	0.9845	0.9897	0.9948
0.7165	0.7165	0.7165	0.7165	0.9794	0.9794	0.9794	0.9948
0.9124	0.9124	0.9021	0.9124	0.9948	0.9948	0.9948	0.9948
0.7990	0.7990	0.7784	0.7990	0.9485	0.9485	0.9433	0.9948
0.9639	0.9639	0.9536	0.9639	0.9897	0.9897	0.9845	0.9948
0.7268	0.7268	0.7062	0.7268	0.9433	0.9433	0.9485	0.9948
0.9175	0.9175	0.9072	0.9175	0.9948	0.9948	0.9897	0.9948
0.7268	0.7268	0.7165	0.7268	0.9433	0.9433	0.9536	0.9948
0.8711	0.8711	0.8608	0.8711	0.9948	0.9948	0.9897	0.9948

Table 4.15. Performance of ICANet1 using FERET database (Dup1), R=0.5

ED	SED	CBD	MKD	CSD	CLD	SMD	SVM
0.6247	0.6247	0.6039	0.6247	0.8546	0.8546	0.8504	0.8296
0.8380	0.8380	0.8186	0.8380	0.9058	0.9072	0.9003	0.8989
0.4931	0.4931	0.4612	0.4931	0.8753	0.8753	0.8684	0.9030
0.7812	0.7812	0.7535	0.7812	0.9280	0.9280	0.9238	0.9197
0.5249	0.5249	0.5028	0.5249	0.8435	0.8407	0.8518	0.9197
0.7396	0.7396	0.7091	0.7396	0.9294	0.9294	0.9155	0.9349
0.6136	0.6136	0.5873	0.6136	0.8227	0.8227	0.8172	0.9349
0.8338	0.8338	0.8116	0.8338	0.8961	0.8961	0.8892	0.9418
0.5235	0.5235	0.4958	0.5235	0.8837	0.8823	0.8726	0.9418
0.7742	0.7742	0.7604	0.7742	0.9252	0.9252	0.9155	0.9446
0.4792	0.4792	0.4765	0.4792	0.8670	0.8684	0.8670	0.9460
0.7355	0.7355	0.7230	0.7355	0.9169	0.9169	0.9086	0.9460
0.6537	0.6537	0.6385	0.6537	0.8407	0.8380	0.8241	0.9460
0.7659	0.7659	0.7576	0.7659	0.8864	0.8878	0.8795	0.9460
0.5360	0.5360	0.5166	0.5360	0.8476	0.8463	0.8546	0.9460
0.7078	0.7078	0.6939	0.7078	0.9114	0.9100	0.8989	0.9460
0.4086	0.4086	0.4086	0.4086	0.8504	0.8504	0.8463	0.9460
0.7299	0.7299	0.6994	0.7299	0.9030	0.9017	0.8906	0.9460

Table 4.16. Performance of ICANet1 using FERET database (Dup2), R=0.5

ED	SED	CBD	MKD	CSD	CLD	SMD	SVM
0.6410	0.6410	0.6111	0.6410	0.7906	0.7906	0.7692	0.7564
0.8077	0.8077	0.7821	0.8077	0.8803	0.8803	0.8675	0.8761
0.5684	0.5684	0.5299	0.5684	0.8376	0.8376	0.8333	0.8803
0.7863	0.7863	0.7436	0.7863	0.8889	0.8889	0.8675	0.8974
0.3889	0.3889	0.3547	0.3889	0.8462	0.8462	0.8547	0.9060
0.6880	0.6880	0.6325	0.6880	0.9145	0.9103	0.9103	0.9145
0.6282	0.6282	0.6026	0.6282	0.8077	0.8077	0.8162	0.9188
0.7778	0.7778	0.7735	0.7778	0.8590	0.8590	0.8504	0.9188
0.4530	0.4530	0.4487	0.4530	0.8162	0.8120	0.8034	0.9188
0.7863	0.7863	0.7692	0.7863	0.8761	0.8761	0.8590	0.9231
0.3974	0.3974	0.3718	0.3974	0.8504	0.8504	0.8376	0.9231
0.6923	0.6923	0.6880	0.6923	0.8718	0.8761	0.8718	0.9274
0.6239	0.6239	0.6111	0.6239	0.8077	0.8077	0.8034	0.9274
0.7692	0.7692	0.7564	0.7692	0.8504	0.8504	0.8376	0.9274
0.4744	0.4744	0.4701	0.4744	0.8248	0.8248	0.8205	0.9274
0.7051	0.7051	0.6880	0.7051	0.8675	0.8675	0.8675	0.9274
0.3547	0.3547	0.3675	0.3547	0.8291	0.8291	0.8376	0.9274
0.6325	0.6325	0.6282	0.6325	0.8889	0.8889	0.8803	0.9274

Table 4.17. Performance of ICANet2 using FERET database (Fb), R=0.5

ED	SED	CBD	MKD	CSD	CLD	SMD	SVM
0.8151	0.8151	0.8017	0.8151	0.9130	0.9121	0.9155	0.8795
0.9632	0.9632	0.9473	0.9632	0.9782	0.9782	0.9774	0.9573
0.8276	0.8276	0.8226	0.8276	0.9339	0.9322	0.9364	0.9598
0.9540	0.9540	0.9490	0.9540	0.9833	0.9833	0.9833	0.9690
0.7607	0.7607	0.7331	0.7607	0.9364	0.9364	0.9456	0.9699
0.9464	0.9464	0.9347	0.9464	0.9808	0.9799	0.9841	0.9741
0.8561	0.8561	0.8477	0.8561	0.9222	0.9222	0.9213	0.9741
0.9439	0.9439	0.9389	0.9439	0.9590	0.9598	0.9531	0.9749
0.7858	0.7858	0.7732	0.7858	0.9431	0.9431	0.9473	0.9749
0.9565	0.9565	0.9473	0.9565	0.9799	0.9799	0.9816	0.9757
0.7205	0.7205	0.7088	0.7205	0.9431	0.9431	0.9490	0.9757
0.9423	0.9423	0.9322	0.9423	0.9808	0.9808	0.9841	0.9766
0.8586	0.8586	0.8494	0.8586	0.9230	0.9213	0.9180	0.9766
0.9473	0.9473	0.9397	0.9473	0.9724	0.9724	0.9749	0.9766
0.7799	0.7799	0.7766	0.7799	0.9456	0.9456	0.9448	0.9766
0.9515	0.9515	0.9498	0.9515	0.9816	0.9816	0.9824	0.9774
0.7464	0.7464	0.7339	0.7464	0.9314	0.9314	0.9389	0.9774
0.9130	0.9130	0.9138	0.9130	0.9833	0.9833	0.9833	0.9782

Table 4.18. Performance of ICANet2 using FERET database (Fc), R=0.5

ED	SED	CBD	MKD	CSD	CLD	SMD	SVM
0.8505	0.8505	0.8402	0.8505	0.9536	0.9536	0.9433	0.9330
0.9742	0.9742	0.9536	0.9742	0.9845	0.9845	0.9794	0.9845
0.8608	0.8608	0.8196	0.8608	0.9639	0.9639	0.9588	0.9897
0.9948	0.9948	0.9742	0.9948	0.9948	0.9948	0.9948	0.9948
0.7938	0.7938	0.7629	0.7938	0.9588	0.9588	0.9742	0.9948
0.9742	0.9742	0.9536	0.9742	0.9948	0.9948	0.9948	0.9948
0.8196	0.8196	0.8093	0.8196	0.9433	0.9433	0.9381	0.9948
0.9742	0.9742	0.9588	0.9742	0.9897	0.9897	0.9897	0.9948
0.8299	0.8299	0.7938	0.8299	0.9485	0.9485	0.9588	0.9948
0.9536	0.9536	0.9433	0.9536	0.9845	0.9845	0.9845	0.9948
0.7680	0.7680	0.7629	0.7680	0.9536	0.9536	0.9588	0.9948
0.8814	0.8814	0.8660	0.8814	0.9897	0.9897	0.9897	0.9948
0.7784	0.7784	0.7784	0.7784	0.9330	0.9330	0.9278	0.9948
0.9588	0.9588	0.9330	0.9588	0.9845	0.9794	0.9691	0.9948
0.8041	0.8041	0.8041	0.8041	0.9433	0.9433	0.9433	0.9948
0.9330	0.9330	0.9227	0.9330	0.9897	0.9897	0.9845	0.9948
0.6907	0.6907	0.6495	0.6907	0.9433	0.9433	0.9433	0.9948
0.7216	0.7216	0.7165	0.7216	0.9845	0.9794	0.9794	0.9948

Table 4.19. Performance of ICANet2 using FERET database (Dup1), R=0.5

ED	SED	CBD	MKD	CSD	CLD	SMD	SVM
0.6759	0.6759	0.6524	0.6759	0.8560	0.8546	0.8504	0.8283
0.8075	0.8075	0.7978	0.8075	0.8975	0.8975	0.8837	0.8934
0.5776	0.5776	0.5485	0.5776	0.8670	0.8670	0.8657	0.8989
0.8075	0.8075	0.7881	0.8075	0.9141	0.9141	0.9155	0.9224
0.4224	0.4224	0.4169	0.4224	0.8684	0.8712	0.8767	0.9224
0.6842	0.6842	0.6662	0.6842	0.9391	0.9391	0.9377	0.9377
0.6828	0.6828	0.6648	0.6828	0.8296	0.8310	0.8296	0.9377
0.8061	0.8061	0.7936	0.8061	0.8920	0.8920	0.8823	0.9391
0.5166	0.5166	0.4931	0.5166	0.8781	0.8781	0.8629	0.9391
0.7618	0.7618	0.7424	0.7618	0.9224	0.9224	0.9183	0.9418
0.4349	0.4349	0.4197	0.4349	0.8698	0.8670	0.8670	0.9418
0.7271	0.7271	0.7064	0.7271	0.9169	0.9169	0.9127	0.9432
0.5554	0.5554	0.5443	0.5554	0.8463	0.8476	0.8407	0.9432
0.7770	0.7770	0.7645	0.7770	0.8947	0.8947	0.8864	0.9432
0.5485	0.5485	0.5499	0.5485	0.8449	0.8435	0.8352	0.9446
0.7244	0.7244	0.7188	0.7244	0.9017	0.9017	0.9030	0.9446
0.4058	0.4058	0.3947	0.4058	0.8560	0.8532	0.8560	0.9446
0.7036	0.7036	0.6898	0.7036	0.9100	0.9114	0.8906	0.9446

Table 4.20. Performance of ICANet2 using FERET database (Dup2), R=0.5

ED	SED	CBD	MKD	CSD	CLD	SMD	SVM
0.5214	0.5214	0.5000	0.5214	0.7906	0.7906	0.7949	0.7692
0.7991	0.7991	0.7650	0.7991	0.8590	0.8590	0.8462	0.8718
0.5342	0.5342	0.5427	0.5342	0.8333	0.8333	0.8162	0.8761
0.7564	0.7564	0.7479	0.7564	0.8932	0.8974	0.8761	0.9359
0.4530	0.4530	0.4188	0.4530	0.8248	0.8248	0.8376	0.9359
0.7051	0.7051	0.6538	0.7051	0.9103	0.9103	0.9017	0.9444
0.6496	0.6496	0.6325	0.6496	0.8034	0.8034	0.8162	0.9444
0.8120	0.8120	0.7821	0.8120	0.8632	0.8632	0.8632	0.9444
0.5598	0.5598	0.5256	0.5598	0.8248	0.8248	0.8333	0.9444
0.7479	0.7479	0.7222	0.7479	0.8889	0.8889	0.8846	0.9444
0.3889	0.3889	0.3846	0.3889	0.8333	0.8333	0.8419	0.9444
0.7479	0.7479	0.7350	0.7479	0.9017	0.9017	0.8974	0.9444
0.5470	0.5470	0.5427	0.5470	0.8248	0.8248	0.8034	0.9444
0.7607	0.7607	0.7436	0.7607	0.8376	0.8376	0.8376	0.9444
0.5342	0.5342	0.5128	0.5342	0.8162	0.8162	0.8333	0.9444
0.7265	0.7265	0.6966	0.7265	0.8632	0.8675	0.8632	0.9444
0.3590	0.3590	0.3504	0.3590	0.8205	0.8205	0.8205	0.9444
0.6752	0.6752	0.6581	0.6752	0.8803	0.8803	0.8889	0.9444

Table 4.21. Performance of algorithms using FERET database (Fb), R=0.5

Networks / classifier	PCANet	ICANet1	ICANet2
ED	0.9690	0.9607	0.9632
SED	0.9690	0.9607	0.9632
CBD	0.9649	0.9506	0.9498
MKD	0.9690	0.9607	0.9632
CSD	0.9883	0.9824	0.9833
CLD	0.9883	0.9824	0.9833
SMD	0.9883	0.9849	0.9841
SVM	0.9841	0.9791	0.9782

Table 4.22. Performance of algorithms using FERET database (Fc), R=0.5

Networks / classifier	PCANet	ICANet1	ICANet2
ED	0.9794	0.9948	0.9948
SED	0.9794	0.9948	0.9948
CBD	0.9639	0.9948	0.9742
MKD	0.9794	0.9948	0.9948
CSD	0.9897	0.9948	0.9948
CLD	0.9897	0.9948	0.9948
SMD	0.9897	0.9948	0.9948
SVM	0.9845	0.9948	0.9948

Table 4.23. Performance of algorithms using FERET database (Dup2), R=0.5

Networks / classifier	PCANet	ICANet1	ICANet2
ED	0.8421	0.8380	0.8075
SED	0.8421	0.8380	0.8075
CBD	0.8338	0.8186	0.7978
MKD	0.8421	0.8380	0.8075
CSD	0.9349	0.9294	0.9391
CLD	0.9349	0.9294	0.9391
SMD	0.9363	0.9238	0.9377
SVM	0.9391	0.9460	0.9446

Table 4.24. Performance of algorithms using FERET database (Dup2), R=0.5

Networks / classifier	PCANet	ICANet1	ICANet2
ED	0.8120	0.8077	0.8120
SED	0.8120	0.8077	0.8120
CBD	0.7949	0.7821	0.7821
MKD	0.8120	0.8077	0.8120
CSD	0.9017	0.9145	0.9103
CLD	0.9017	0.9103	0.9103
SMD	0.9103	0.9103	0.9017
SVM	0.9145	0.9274	0.9444

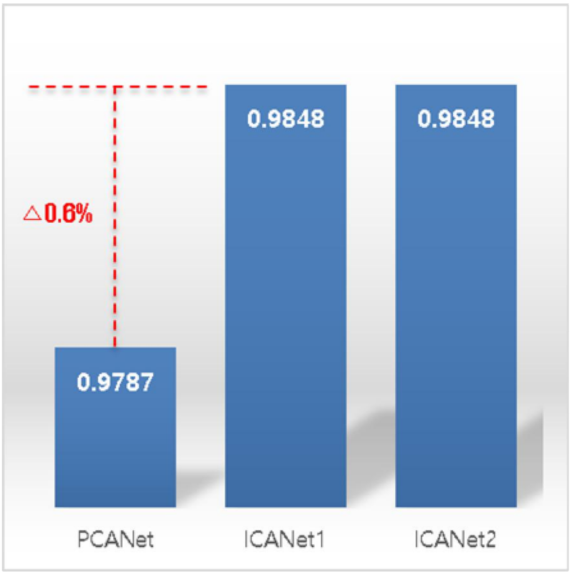


Figure 4.12. Performances of CU-ECG using SVM

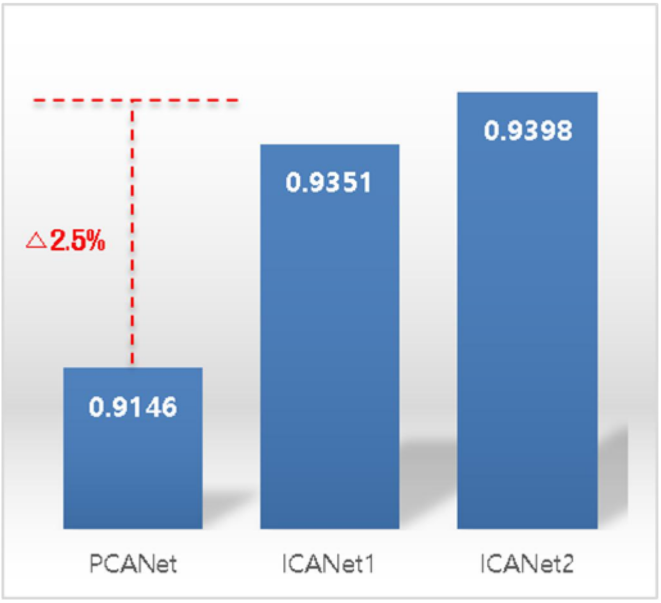


Figure 4.13. Performances of Gaussian noise CU-ECG using SVM

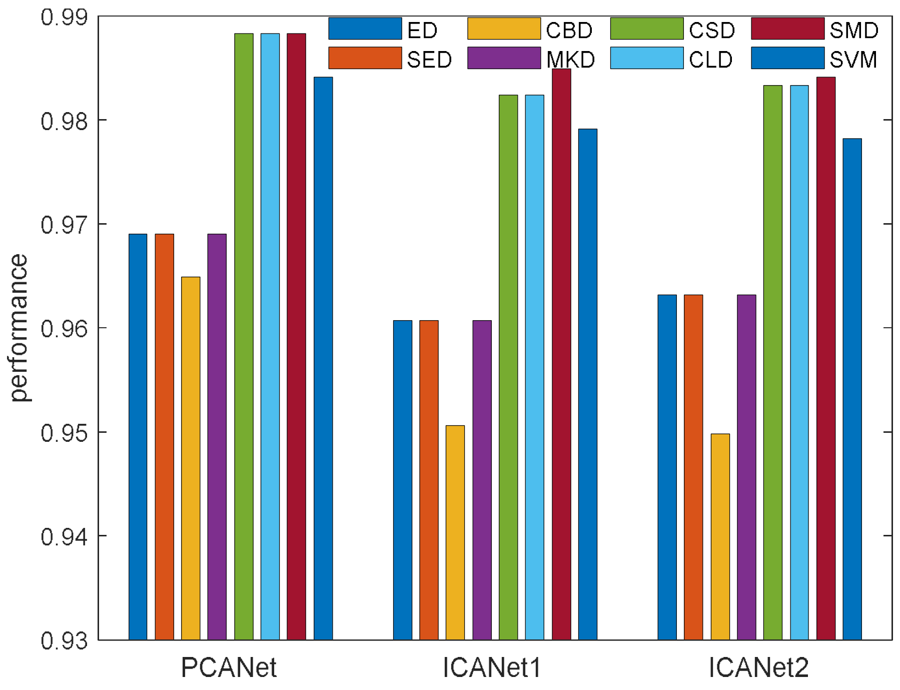


Figure 4.14. Performance of FERET database (Fb), R=0.5

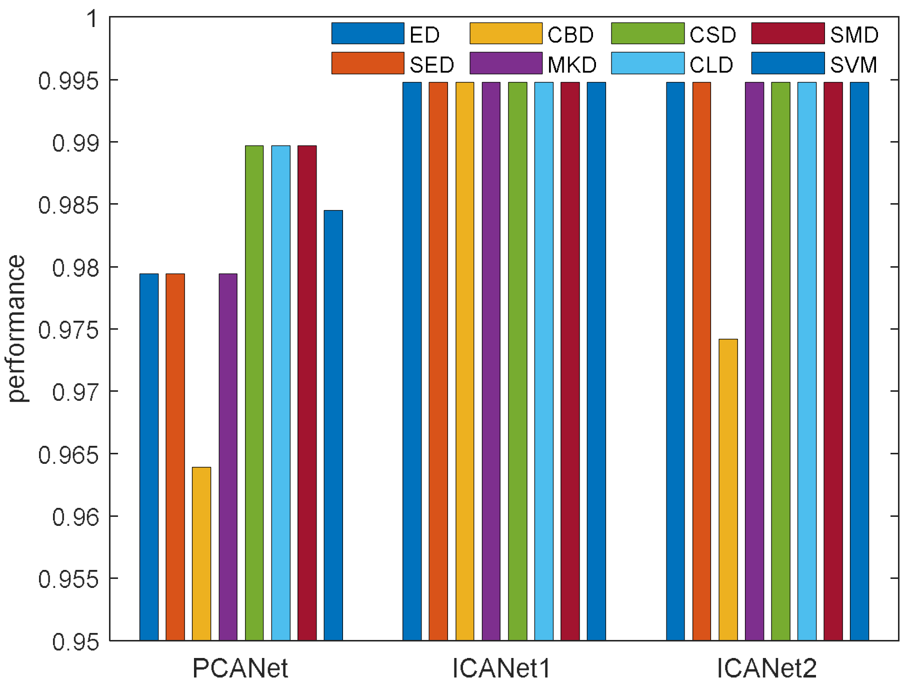


Figure 4.15. Performance of FERET database (Fc), R=0.5

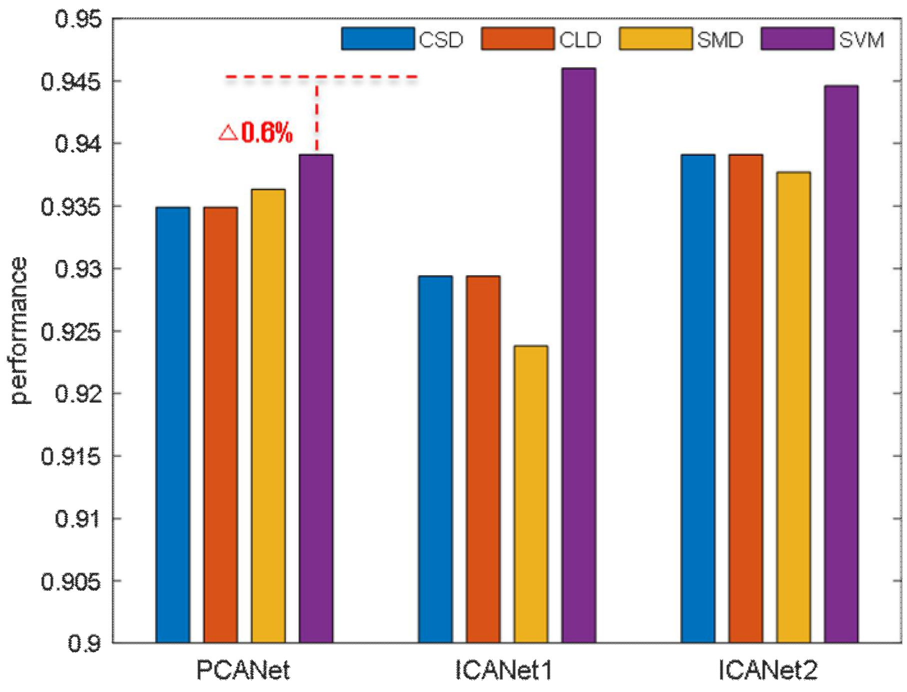


Figure 4.16. Performance of FERET database (Dup1), R=0.5

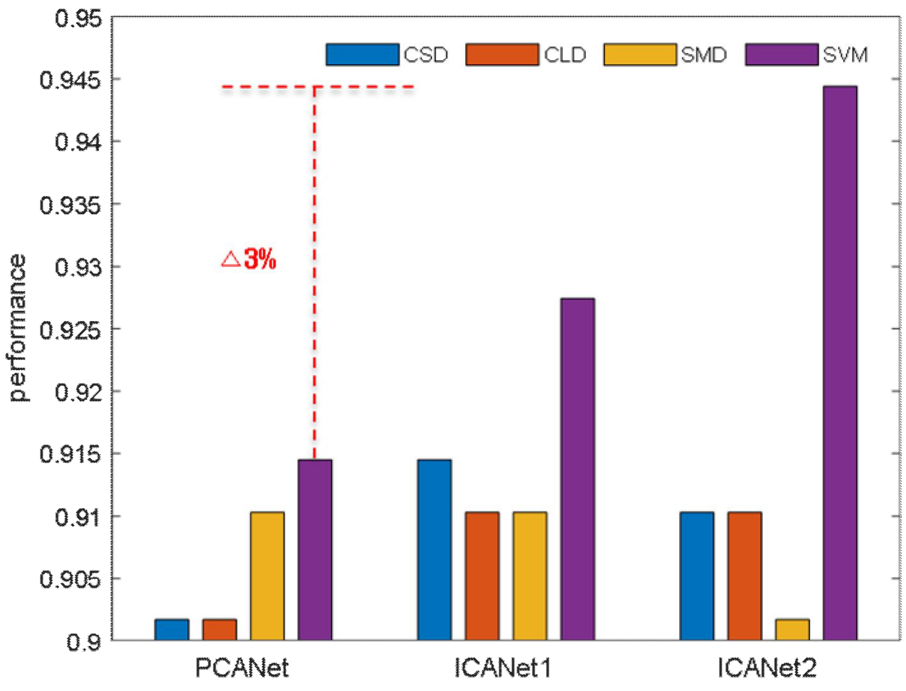


Figure 4.17. Performance of FERET database (Dup2), R=0.5

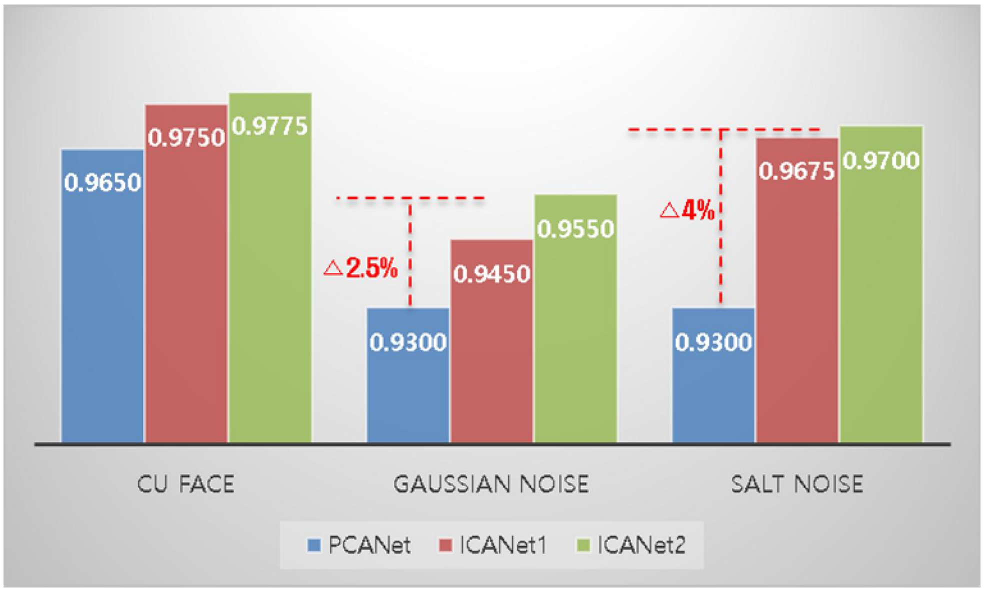


Figure 4.18. Performance of CU-Face database

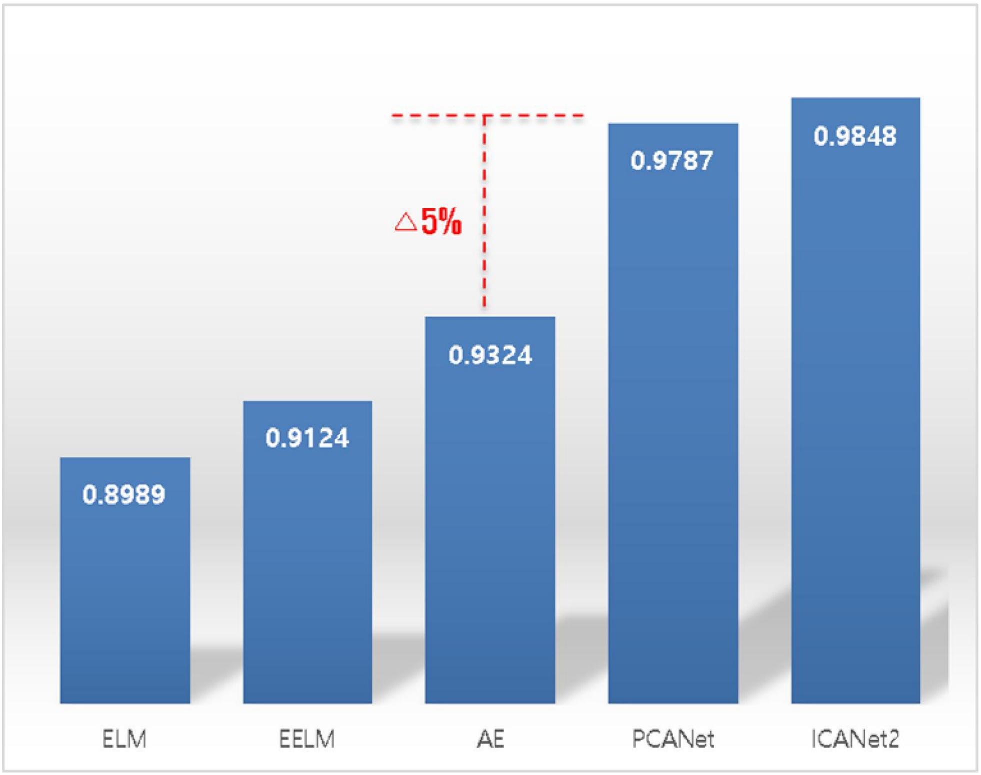


Figure 4.19. Performance of CU-ECG

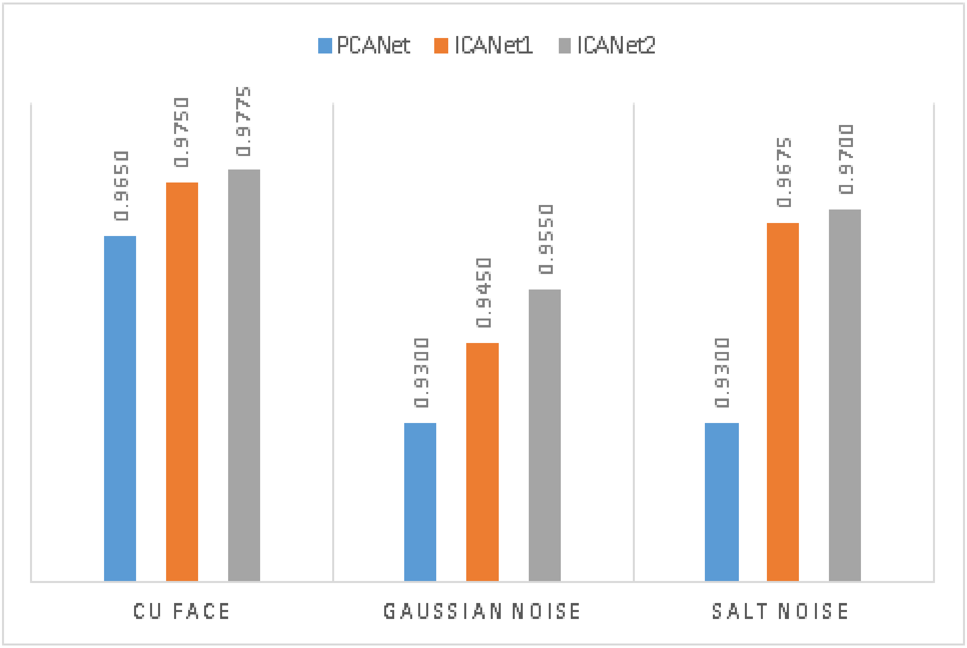


Figure 4.20. Performance of CU-Face

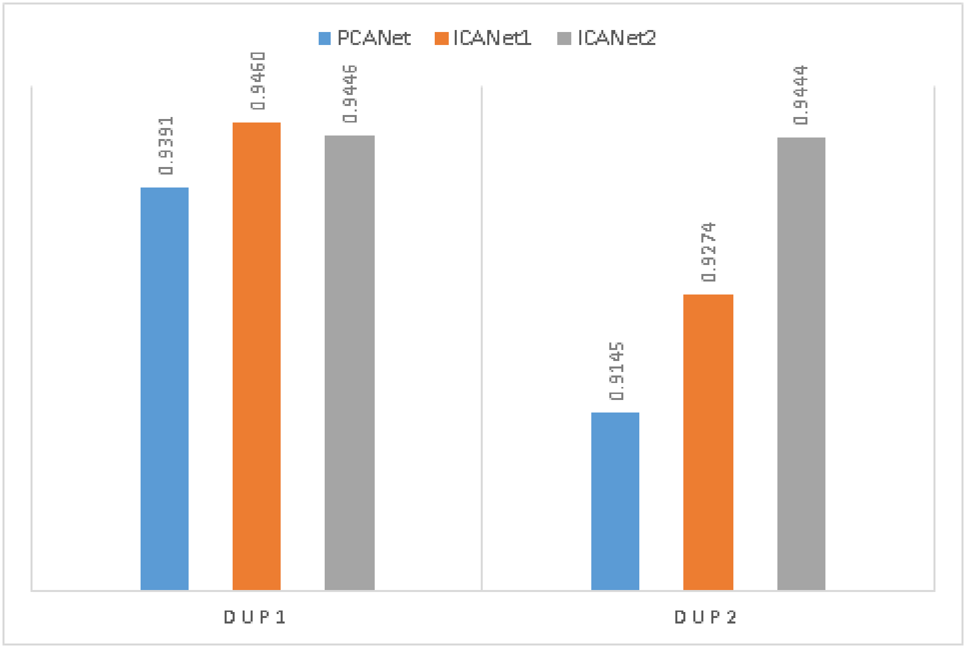


Figure 4.21. Performance of FERET

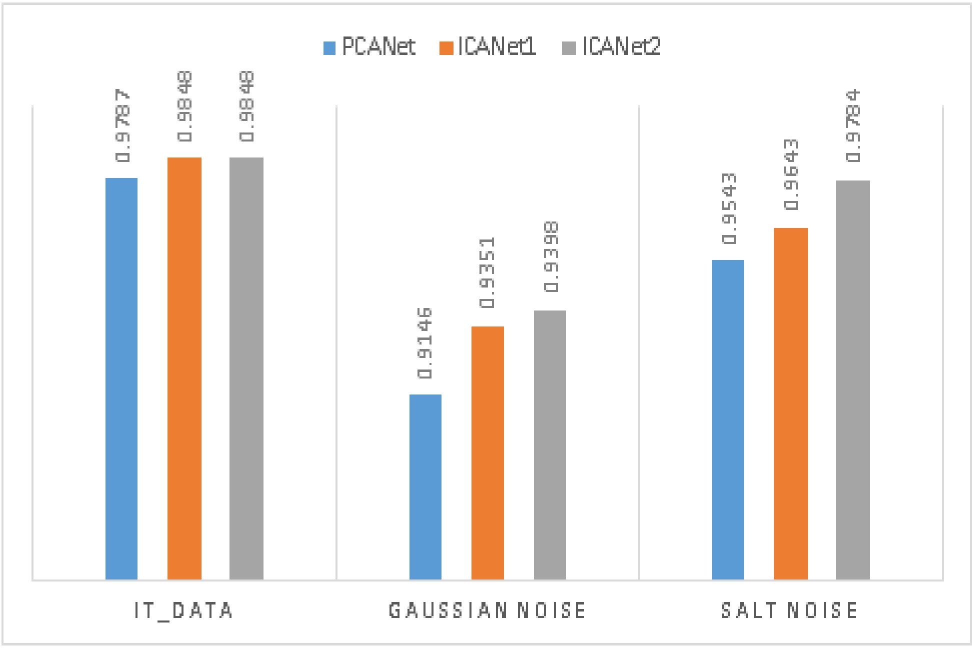


Figure 4.22. Performance of CU-ECG

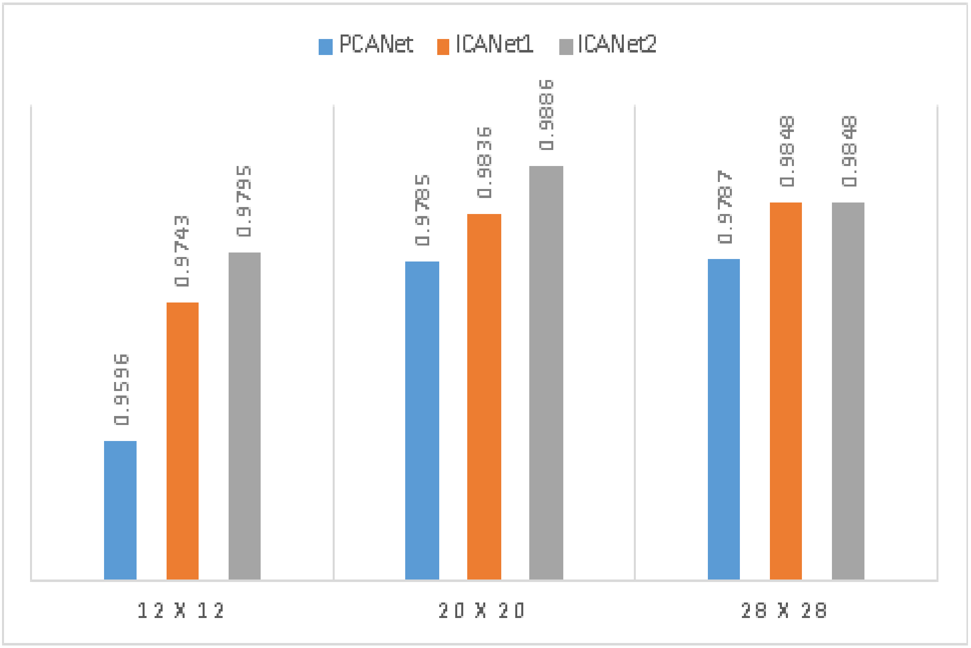


Figure 4.23. Performance of scale CU-ECG

Ⅴ. Conclusion

In this paper, we have organized the ICANet1 algorithm to improve the disadvantages of ICANet, and studied the new ICANet2 model. Experimental results show that PCANet algorithm has low performance when using noise data and small feature data. Normally, when processed data is used, it shows similar performance to PCANet. However, the PCANet using Eigenvectors is weak against noise, and the feature vector using statistical independence is robust against noise. In addition, ICANet showed robustness compared to PCANet, even with loss of data information. In addition, the performance was better than the basic deep learning of auto-encoders and the classification time was about 0.05 seconds faster than PCANet. We could expect to improve the performance of face data as well as electrocardiogram data. ICANet was superior to PCANet when all data were used. Especially, ICANet showed higher performance than PCANet when using noise data. Finally, we used ECG and facial data for personal identification security problem. ICANet algorithm has been studied to improve the problem of deep learning (computational complexity and composition method). ICANet 's effectiveness has been shown to be superior to computational weight and time although the recognition rate can be reduced due to shallow learning.

References

[1] H. Nguyen, L.M. Kieu, T. Wen, and C. Cai, "Deep learning methods in transportation domain: a review," *IET Intell. Transp. Syst,* vol. 12, pp. 998-1004, 2018.

[2] R. Mu, X. Zeng, and L. Han, "A Survey of recommender systems based on deep learning," *IEEE Access*, vol. 6, pp. 69009-69022, 2018.

[3] F. Celesti, A. Celesti, J. Wan, and M. Villari, "Why deep learning is changing the way to approach NGS data processing: a review," *IEEE Rev. Biomed. Eng,* vol. 11, pp. 68-76, 2018.

[4] X. W. Chen and X. Lin, "Big data deep learning: challenges and perspectives," *IEEE Access,* vol. 2, pp. 514-525, 2014.

[5] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *J. Big Data,* vol. 2, no. 1, pp. 1-21, 2015.

[6] P. Angelov and A. Sperduti, "Challenges in deep learning," *Proc. Eur. Symp. Artif. Neural Networks*, no. 1, pp. 489-496, 2016.

[7] I. T. Jolliffe, "Principal component analysis. Second Edition," *Springer Ser. Stat,* vol. 98, pp. 487, 2002.

[8] N. N. Bui, S. H. Min, and J. Y. Kim, "Human detection in video using poselet with articulated pose estimation and edge-based RPCA foreground extraction," *Korean Society of Information Technology Association*, vol. 12, no. 3, pp. 51-60, 2014.

[9] Y. Byeon, W. Lim, J. Lee, H. Jeong, H. Han, and C. Kwak, "Individual identification on electrocardiogram using third-order tensor-based MPCA," *Journal of advanced engineering and technology*, vol. 11, pp. 151-157, 2018.

[10] M. S. Park and S. Oh, "Nonlinear feature extraction using class-augmented kernel PCA," *International Conference on Artificial*

Neural Networks, 2011.

[11] B. Kim, S. Oh, and J. Kim, “Design of digit recognition system realized with the aid of fuzzy RBFNNs and incremental,” *Korean Association of Intelligence Systems*, vol. 1, no. 1, pp. 56-63, 2016.

[12] M. Jin, R. Li, J. Jiang, and B. Qin, “Extracting contrast-filled vessels in X-ray angiography by graduated RPCA with motion coherency constraint,” *Pattern Recognit,* vol. 63, no. 1, pp. 653-666, 2017.

[13] Y. Lu, Z. Lai, Z. Fan, J. Cui, and Q. Zhu, “Manifold discriminant regression learning for image classification,” *Neurocomputing*, vol. 166, pp. 475-486, 2015.

[14] Y. Li, G. Liu, Q. Liu, Y. Sun, and S. Chen, “Moving object detection via segmentation and saliency constrained RPCA,” *Neurocomputing*, vol. 323, no. 1, pp. 352-362, 2019.

[15] N. Vaswani, T. Bouwmans, S. Javed, and P. Narayanamurthy, “Robust subspace learning: robust PCA, robust subspace tracking, and robust subspace recovery,” *IEEE Signal Process. Mag*, vol. 35, no. 4, pp. 32-55, 2018.

[16] J. X. Liu, Y. Xu, C. H. Zheng, H. Kong, and Z. H. Lai, “RPCA-based tumor classification using gene expression data,” *IEEE/ACM Trans. Comput. Biol. Bioinforma*, vol. 12, no. 4, pp. 964-970, 2015.

[17] W. Sun and Q. Du, “Graph-regularized fast and robust principal component analysis for hyperspectral band selection,” *IEEE Trans. Geosci. Remote Sens*, vol. 56, no. 6, pp. 3185-3195, 2018.

[18] F. Zhong, L. Liu, and J. Hu, “Robust 2DLDA based on correntropy,” *Neurocomputing*, vol. 316, no. 1, pp. 399-404, 2018.

[19] M. Yin, D. Zeng, J. Gao, Z. Wu, and S. Xie, “Robust multinomial logistic regression based on RPCA,” *IEEE J. Sel. Top. Signal Process*, vol. 12, no. 6, pp. 1144-1154, 2018.

[20] F. Kuang, W. Xu, and S. Zhang, “A novel hybrid KPCA and SVM with GA model for intrusion detection,” *Appl. Soft Comput. J*, vol. 18, pp. 178-184, 2014.

[21] J. Liu, C. Zhang, and C. Zheng, “EEG-based estimation of mental fatigue by using KPCA-HMM and complexity parameters,” *Biomed. Signal Process. Control*, vol. 5, no. 2, pp. 124-130, 2010.

[22] A. Vinay, V. S. Shekhar, K. N. B. Murthy, and S. Natarajan, “Face recognition using gabor wavelet features with PCA and KPCA - a comparative study,” *Procedia Comput. Sci*, vol. 57, pp. 650-659, 2015.

[23] Q. Wang, “Kernel principal component analysis and its applications in face recognition and active shape models,” *Computer Vision and Pattern Recognition,* vol. 21, no. 10, pp. 1-9, 2012.

[24] A. Romero, C. Gatta, and G. Camps-Valls, “Unsupervised deep feature extraction for remote sensing image classification,” *IEEE Trans. Geosci. Remote Sens*, vol. 54, no. 3, pp. 1349-1362, 2016.

[25] J. Xia, N. Falco, J. A. Benediktsson, P. Du, and J. Chanussot, “Hyperspectral image classification with rotation random forest via KPCA,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens*, vol. 10, no. 4, pp. 1601-1609, 2017.

[26] F. Yuan, X. Xia, J. Shi, H. Li, and G. Li, “Non-linear dimensionality reduction and gaussian process based classification method for smoke detection,” *IEEE Access*, vol. 5, pp. 6833-6841, 2017.

[27] K. Hotta, “Local co-occurrence features in subspace obtained by KPCA of local blob visual words for scene classification,” *Pattern Recognit*, vol. 45, no. 10, pp. 3687-3694, 2012.

[28] Y. Xiao, H. Wang, W. Xu, and J. Zhou, “L1 norm based KPCA for novelty detection,” *Pattern Recognit*, vol. 46, no. 1, pp. 389-396, 2013.

[29] S. Chaib, Y. Gu, H. Yao, and S. Zhao, “A VHR scene classification method integrating sparse PCA and saliency computing,” *Int. Geosci. Remote Sens. Symp*, vol. 2016, pp. 2742-2745, 2016.

[30] A. Mishra, N. S. Rajput, D. Singh, “An object linked intelligent

classification method for hyperspectral images," *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 3345-3348.

[31] P. H. Hsu and Y. H. Tseng, "Feature extraction for hyperspectral Image," *Proc. 20th Asian Conf. Remote Sens*, pp. 405-410, 1999.

[32] X. Xiao and Y. Zhou, "Two-Dimensional quaternion PCA and Sparse PCA," *IEEE Trans. Neural Networks Learn. Syst*, pp. 1-15, 2018.

[33] Z. Hu, G. Pan, Y. Wang, and Z. Wu, "Sparse principal component analysis via rotation and truncation," *Adv. Princ. Compon. Anal. Res. Dev*, vol. 27, no. 4, pp. 1-18, 2017.

[34] A. De Pierrefeu, "Structured sparse principal components analysis with the TV-Elastic net penalty," *IEEE Trans. Med. Imaging*, vol. 37, no. 2, pp. 396-407, 2018.

[35] L. Wang, X. Xie, W. Li, Q. Du, and G. Li, "Sparse feature extraction for hyperspectral image classification," *2015 IEEE China Summit Int. Conf. Signal Inf. Process*, pp. 1067-1070, 2015.

[36] S. Chaib, Y. Gu, and H. Yao, "An informative feature selection method based on sparse PCA for VHR scene classification," *IEEE Geosci. Remote Sens. Lett*, vol. 13, no. 2, pp. 147-151, 2016.

[37] F. Li, A. Wong, and D. A. Clausi, "Comparative study of feature space projection methods for hyperspectral image classification," *Int. Geosci. Remote Sens. Symp*, pp. 3438-3441, 2014.

[38] S. B. Liu, Z. Y. Yuan, J. H. Zhao, and X. L. Wang, "An approach to face recognition based on wavelet decomposition, spca and svm," *Proc. 2009 Int. Conf. Mach. Learn. Cybern*, vol. 2, no. 2, July, pp. 989-993, 2009.

[39] Y. T. Xi and P. J. Ramadge, "Separable PCA for image classification," ICASSP, *IEEE Int. Conf. Acoust. Speech Signal Process*, pp. 1805-1808, 2009.

[40] I. Dagher and R. Nachar, "Face recognition using IPCA-ICA algorithm," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 28, no. 6,

pp. 996-1000, 2006.

[41] A. Rozza, G. Lombardi, and E. Casiraghi, “Novel IPCA-based classifiers and their application to spam filtering,” *Int. Conf. Intell. Syst. Des. Appl*, pp. 797-802, 2009.

[42] Y. Yuan, Y. Pang, J. Pan, and X. Li, “Scene segmentation based on IPCA for visual surveillance,” *Neurocomputing,* vol. 72, no. 10, pp. 2450-2454, 2009.

[43] X. Qu and M. Yao, “Adaptive subspace incremental PCA based online learning for object classification and recognition,” *Int. Congr. Image Signal Process. CISP 2011*, vol. 3, pp. 1494-1498, 2011.

[44] C. Qing, S. Zhao, and X. Xu, “The incremental PCA tracking with negative samples,” *2014 IEEE Int. Conf. Consum. Electron*. pp. 1-4, 2014.

[45] Y. Wang, J. Zhou, Z. Li, Z. Dong, and Y. Xu, “Discriminant-analysis-based single-phase earth fault protection using improved PCA in distribution systems,” *IEEE Trans. Power Deliv*, vol. 30, no. 4, pp. 1974-1982, 2015.

[46] Y. Yin, D. Xu, X. Wang, and M. Bai, “Online state-based structured SVM combined with incremental PCA for robust visual tracking,” *IEEE Trans. Cybern*, vol. 45, no. 9, pp. 1988-2000, 2015.

[47] A. A. Alorf, “Performance evaluation of the PCA versus improved PCA (IPCA) in image compression, and in face detection and recognition,” *FTC 2016 - Proc. Futur. Technol. Conf*, no. 1, pp. 537-546, 2017.

[48] M. Bengherabi, L. Mezai, F. Harizi, A. Guessoum, and M. Cheriet, “Score fusion of SVD and DCT-RLDA for face recognition,” *2008 1st Int. Work. Image Process. Theory, Tools Appl. IPTA 2008*, pp. 1-8, 2008.

[49] H. Yuan, Y. Y. Tang, Y. Lu, L. Yang, and H. Luo, “Spectral-spatial classification of hyperspectral image based on discriminant analysis,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens*, vol. 7,

no. 6, pp. 2035-2043, 2014.

[50] C. He, T. Zhuo, D. Ou, M. Liu, and M. Liao, "Nonlinear compressed sensing-based LDA topic model for polarimetric SAR image classification," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens*, vol. 7, no. 3, pp. 972-982, 2014.

[51] R. Zeng, J. Wu, L. Senhadji, and H. Shu, "Tensor object classification via multilinear discriminant analysis network," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process,* vol. 2, pp. 1971-1975, 2015.

[52] S. J. Hong and Y. H. Cho, "Image recognition by using hybrid coefficient measure of correlation and distance," *J. Korean Inst. Intell. Syst*, vol. 20, no. 3, pp. 343-347, 2011.

[53] J. S. Hwang, J. H. Kim, "A study on the estimation of modal parameters using independent component analysis method," *Architectural Institute of Korea,* vol. 27, pp. 27-35, 2011.

[54] S. C. Baek, J. S. Hong, "Image classification based on multi-resolution characteristic and independent component analysis," *Proceedings of KIIT Conference*, pp. 455-460, 2012.

[55] K. H. Kwon, J. D. Lim, "Dried pepper sorting using independent component analysis on RGB images," *Korean society of computer and information*, vol. 1, pp. 59-65, 2012.

[56] K. S. Kang, H. J, Kim, J. S. Hwang, "Mode decomposition and output-only system identification of lateral-torsionally coupled structures using independent component analysis method," *Architectural Institute of Korea*, vol. 28, pp. 37-45, 2012.

[57] K. H. Kim, "Vibration source contribution analysis of plate structure using independent component analysis," *Journal of the Korea Maritime Engineering Association*, vol. 26, pp. 70-76, 2012.

[58] K. Kim, J. J. Jun, D. S. Cho, J. H. Kim, and H. M. Kwon, "Vibration source signal identification of structures using ICA," *J. Soc. Nav.*

Archit. Korea, vol. 49, no. 6, pp. 498-503, 2013.

[59] X. Quan, K. S. Bae, “Frequency bin alignment using covariance of power ratio of separated signals in multi-channel FD-ICA,” *Korean Association of Voice of Korea*, vol. 6, pp. 149-153, 2014.

[60] C. G. Park, “Blind source separation using ICA and masking operation in time-frequency domain,” *Korea Inst. Inf. Technol. Rev*, vol. 11, no. 10, 2014.

[61] J. S. Hwang, “A novel mode decomposition method for non-classical damping structure using acceleration responses,” *J. Archit. Inst. Korea Struct. Constr*, vol. 31, no. 1, pp. 19-26, 2015.

[62] J. Lee and B. Lee, “Design of filter to remove motionartifacts of photoplethysmography based on indepenent components analysis and filter banks,” *Korean Society of Information and Communication,* vol. 20, pp. 1431-1437, 2016.

[63] C. Symposium, “Designed of partial discharge pattern classifier of independent component analysis based fuzzy neural networks,” *Korean institute of electrical engineers*, vol. 2017, no. 10, pp.142-143, 2017.

[64] M. Wu, J. Zhou, and J. Sun, “Multi-scale ICA texture pattern for gender recognition,” *Electron. Lett*, vol. 48, no. 11, pp. 629, 2012.

[65] Z. L. Sun, C. H. Zheng, Q. W. Gao, J. Zhang, and D. X. Zhang, “Tumor classification using eigengene-based classifier committee learning algorithm,” *IEEE Signal Process. Lett*, vol. 19, no. 8, pp. 455-458, 2012.

[66] R. J. Martis, U. R. Acharya, and L. C. Min, “ECG beat classification using PCA, LDA, ICA and Discrete Wavelet Transform,” *Biomed. Signal Process. Control*, vol. 8, no. 5, pp. 437-448, 2013.

[67] X. Wang and X. P. Zhang, “An ICA mixture hidden conditional random field model for video event classification,” *IEEE Trans. Circuits Syst. Video Technol*. vol. 23, no. 1, pp. 46-59, 2013.

[68] C. Yu, Q. Qui, Y. Zhao, and X. Chen, “Satellite image classification using morphological component analysis of texture and cartoon layers,” *IEEE Geosci. Remote Sens. Lett,* vol. 10, no. 5, pp. 1109-1113, 2013.

[69] N. Falco, J. A. Benediktsson, and L. Bruzzone, “A study on the effectiveness of different independent component analysis algorithms for hyperspectral image classification,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens*, vol. 7, no. 6, pp. 2183-2199, 2014.

[70] A. X. Stewart, A. Nuthmann, and G. Sanguinetti, “Single-trial classification of EEG in a visual object task using ICA and machine learning,” *J. Neurosci. Methods*, vol. 228, no. 1, pp. 1-14, 2014.

[71] Y. Zhao, S. Wei, Y. Xiao, Z. Zhu, and Y. Wei, “Kernel Reconstruction ICA for Sparse Representation,” *IEEE Trans. Neural Networks Learn. Syst*, vol. 26, no. 6, pp. 1222-1232, 2014.

[72] R. H. R. Pruim, M. Mennes, D. van Rooij, A. Llera, J. K. Buitelaar, and C. F. Beckmann, “ICA-AROMA: A robust ICA-based strategy for removing motion artifacts from fMRI data,” *Neuroimage*, vol. 112, no. 1, pp. 267-277, 2015.

[73] R. Mahajan and B. I. Morshed, “Unsupervised eye blink artifact denoising of EEG data with modified multiscale sample entropy, kurtosis, and wavelet-ICA,” *IEEE J. Biomed. Heal. Informatics*, vol. 19, no. 1, pp. 158-165, 2015.

[74] J. Coloigner, “A novel classification method for prediction of rectal bleeding in prostate cancer radiotherapy based on a semi-nonnegative ICA of 3D planned dose distributions,” *IEEE J. Biomed. Heal. Informatics*, vol. 19, no. 3, pp. 1168-1177, 2015.

[75] M. Tao, F. Zhou, Y. Liu, and Z. Zhang, “Tensorial independent component analysis-based feature extraction for polarimetric SAR Data Classification,” *IEEE Trans. Geosci. Remote Sens*, vol. 53, no. 5, pp. 2481-2495, 2015.

[76] J. Bijsterbosch, “Hand classification of fMRI ICA noise

components," *Neuroimage*, vol. 154, no. 1, pp. 188-205, 2016.

[77] R. Chai, "Driver Fatigue Classification with Independent Component by Entropy Rate Bound Minimization Analysis in an EEG-Based System," *IEEE J. Biomed. Heal. Informatics*, vol. 21, no. 3, pp. 715 -724, 2017.

[78] M. Iwahashi, H. Arof, P. Cumming, N. Mokhtar, and C. Y. Sai, "Automated Classification and Removal of EEG Artifacts With SVM and Wavelet-ICA," *IEEE J. Biomed. Heal. Informatics*, vol. 22, no. 3, pp. 664-670, 2017.

[79] Y. Zhang, T. Geng, X. Wu, J. Zhou, and D. Gao, "ICANet: a simple cascade linear convolution network for face recognition," *Eurasip J. Image Video Process*, vol. 2018, no. 1, 2018.

[80] C. Zhang, X. Wu, X. Gao, Y. Wang, and Z. Lv, "An ICA-based spatial filtering approach to saccadic EOG signal recognition," *Biomed. Signal Process. Control*, vol. 43, no.1, pp. 9-17, 2018.

[81] R. Salakhutdinov, J. B. Tenenbaum, and A. Torralba, "Learning with hierarchical-deep models," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 35, no. 8, pp. 1958-1971, 2013.

[82] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens*, vol. 7, no. 6, pp. 2094-2107, 2014.

[83] T. Dobhal, V. Shitole, G. Thomas, and G. Navada, "Human Activity Recognition using Binary Motion Image and Deep Learning," *Procedia Comput. Sci*, vol. 58, no. 1, pp. 178-185, 2015.

[84] M. Hasan and A. K. Roy-Chowdhury, "A Continuous Learning Framework for Activity Recognition Using Deep Hybrid Feature Models," *IEEE Trans. Multimed*, vol. 17, no. 11, pp. 1909-1922, 2015.

[85] Y. Du, Y. Fu, and L. Wang, "Representation Learning of Temporal Dynamics for Skeleton-Based Action Recognition," *IEEE Trans. Image Process*, vol. 25, no. 7, pp. 3010-3022, 2016.

[86] N. Neverova, "Learning Human Identity from Motion Patterns," *IEEE*

Access, vol. 4, no. 1 pp. 1810-1820, 2016.

[87] P. Wang, W. Li, Z. Gao, J. Zhang, C. Tang, and P. O. Ogunbona, "Action Recognition from Depth Maps Using Deep Convolutional Neural Networks," *IEEE Trans.* Human-Machine Syst., vol. 46, no. 4, pp. 498-509, 2016.

[88] M. Koohzadi and N. M. Charkari, "Survey on deep learning methods in human action recognition," *IET Comput. Vis*, vol. 11, no. 8, pp. 623-632, 2017.

[89] Y. Kim and Y. Li, "Human Activity Classification with Transmission and Reflection Coefficients of On-Body Antennas Through Deep Convolutional Neural Networks," *IEEE Trans. Antennas Propag*, vol. 65, no. 5, pp. 2764-2768, 2017.

[90] Y. Liu, Q. Wu, L. Tang, and H. Shi, "Gaze-Assisted Multi-Stream Deep Neural Network for Action Recognition," *IEEE Access*, vol. 5, pp. 19432-19441, 2017.

[91] D. Holden, "Edinburgh Research Explorer Fast Neural Style Transfer for Motion Data," *IEEE Computer Graphics and Applications*, vol. 37. no. 4, pp. 42-47, 2017.

[92] H. Chen, J. Chen, R. Hu, C. Chen, and Z. Wang, "Action recognition with temporal scale-invariant deep learning framework," *China Commun*, vol. 14, no. 2, pp. 163-172, 2017.

[93] Z. Wu, Y. Huang, L. Wang, X. Wang, and T. Tan, "A Comprehensive Study on Cross-View Gait Based Human Identification with Deep CNNs," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 39, no. 2, pp. 209-226, 2017.

[94] B. Ni, T. Li, and X. Yang, "Learning semantic-aligned action representation," *IEEE Trans. Neural Networks Learn. Syst*, vol. 29, no. 8, pp. 3715-3725, 2018.

[95] H. Rahmani, A. Mian, and M. Shah, "Learning a Deep Model for Human Action Recognition from Novel Viewpoints," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 40, no. 3, pp. 667-681, 2018.

[96] L. Wang, Y. Xu, J. Cheng, H. Xia, J. Yin, and J. Wu, “Human Action Recognition by Learning Spatio-Temporal Features with Deep Neural Networks,” *IEEE Access*, vol. 6, pp. 17913-17922, 2018.

[97] C. Yang, N. Tavassolian, “An Independent Component Analysis Approach to Motion Noise Cancelation of,” *IEEE Trans. Biomed. Eng.*, vol. 66, no. 3, pp. 784-793, 2019.

[98] Y. Zheng and X. Hu, “Interference Removal From Electromyography Based on Independent Component Analysis,” *IEEE Trans. Neural Syst. Rehabil. Eng*, vol. 27, no. 5, pp. 887-894, 2019.

[99] T. Chan and Y. Ma, “PCANet: A Simple Deep Learning Baseline for Image Classification?,” *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017-5032, 2015.

[100] J. Wu, “PCANet: An energy perspective,” *Neurocomputing*, vol. 313, pp. 271-287, 2018.

List of publications

Paper

1. <u>**Jae-Neung Lee,**</u> Myung-Won Lee, Yeong-Hyeon Byeon, Won-Sik Lee, Keun-Chang Kwak, "Classification of horse gaits using FCM-based neuro-fuzzy classifier from the transformed data information of inertial sensor", *Sensors,* vol. 16, Article ID 664, 2016.

2. Yeong-Hyeon Byeon, Myung-Won Lee, <u>**Jae-Neung Lee,**</u> Keun-Chang Kwak, "Prediction of dinghy boom direction using intelligent predictor", *International Journal of Control Automation and Systems,* vol. 16, no. 1, pp. 368-376, 2018.

3. <u>**Jae-Neung Lee,**</u> Yeong-Hyeon Byeon and Keun-Chang Kwak, "Design of Ensemble Stacked Auto-Encoder for Classification of Horse Gaits with MEMS Inertial Sensor Technology", *Micromachines*, vol. 9, no. 8, pp. 1-17, 2018.

4. Yeong-Hyeon Byeon, Myung-Won Lee, <u>**Jae-Neung Lee,**</u> Keun-Chang Kwak, "Multilinear EigenECGs and FisherECGs for Individual Identification from Information Obtained by an Electrocardiogram Sensor", *Symmetry.* vol. 10, no. 10, pp. 1-21, 2018.

5. <u>**Jae-Neung Lee,**</u> Yeong-Hyeon Byeon, Sum Bum Pan, and Keun-Chang Kwak, An EigenECG network approach based on PCANet for personal identification from ECG signal", *Sensors.* vol. 18, no. 11, pp. 1-25, 2018.

6. <u>**Jae-Neung Lee,**</u> Keun-Chang Kwak, "Personal identification using a robust EigenECG netwrok based on time-frequency representations of ECG signals", *IEEE Access*, vol. 7, pp. 48392-48404, 2019.

7. <u>**이재능,**</u> 곽근창, "셀프코칭 시스템을 위한 웨이블릿 패킷과 오토 엔코더 기반 승마 보법분류", 한국정보기술학회논문지, vol. 15, no. 5, pp. 1-9, May 31. 2017.

List of publications

Conference

1. **<u>Jae-Neung Lee,</u>** Keun-Chang Kwak, "A Trend Analysis of Image Processing Approaches in Unmanned Aerial Vehicle", *International Conference on Image Processing, Analysis and Computer Vision (IPACV)*, pp. 271-274, Feb. 2014.

2. **<u>Jae-Neung Lee,</u>** Keun-Chang Kwak, "A Trends Analysis of Yatch Simulator Dinghy", *International Conference on Wireless and Mobile Networks (ICWMN)*, pp. 1272-1275, Feb. 2015.

3. **<u>Jae-Neung Lee,</u>** Myung-Won Lee, Jung-Su Han, Sung-Bum. Pan, Keun-Chang Kwak, "Yacht DB Construction Based on Five Essentials of Sailing", *International Conference on Soft Computing and Data Mining (ICSCDM),* pp. 1276-1279, Feb, 2015.

4. **<u>Jae-Neung Lee,</u>** Keun-Chang Kwak, "A Performance Comparison of Auto-Encoder and Its Variants for Classification", *International Conference on Signals and Systems (ICSigSys)*, pp. 207-211, May, 2017.

5. **<u>Jae-Neung Lee,</u>** Sum Bum Pan, Keun-Chang Kwak, "IIndividual identification Based on Cascaded PCANet from ECG Signal", *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1-4, Jan, 2019.

7. **<u>이재능,</u>** 곽근창, “다선형 PCA를 이용한 K-POP 댄스모션 분류”, *한국정보처리학회*, 건국대, 2018년 5월

8. **<u>이재능</u>**, 곽근창, “Stacked 오토엔코더 기반 승마 보법 분류”, *한국정보처리학회 추계학술발표대회*, 부산대, 2016년 11월 5.

9. **<u>이재능</u>**, 반성범, 곽근창, "요트 시뮬레이터와 모션 분석의 국내외 연구동향 분석", *한국정밀공학회 춘계학술대회논문집*, pp 1053-1054, 2015년 5월 13.

10. **이재능,** 곽근창, "관성센서 기반의 모션 캡쳐 슈트를 이용한 승마 모션 분석", *한국정보기술학회 하계학술대회*, 공주대, pp. 92-95, 2014년 5월.

11. **이재능,** 곽근창, "국내 무인항공기 영상처리 동향분석", *한국스마트미디어학회 추계학술대회*, 군산대, pp 221-225, 2013년 11월.

12. **이재능,** 곽근창, "스포츠 시뮬레이터의 국내외 연구개발 동향 분석", *순천 정원엑스포 ICT 합동 학술대회 논문집*, 순천대, pp. 141-143 2013년 6월

List of publications

Patent

1. Keun-Chang Kwak, Myung-Won Lee, Yeong-Hyeon Byeon, **Jae-Neung Lee**, “Real-time coaching system based on three dimensional motion analysis of horse rider“, 10-2013-0138335, 2013.
2. **Jae-Neung Lee,** Keun-Chang Kwak, Sum Bum Pan, Myung-Won Lee, Yeong-Hyeon Byeon, “Dinghy yacht coaching system and method thereof”, 10-2016-0022355, 2016.

List of publications

S/W Registration

1. 변영현, 이명원, <u>**이재능**</u>, 곽근창, "승마보법별 승마자의 자세분석 프로그램", KR, C-2013-009539 호, 2013.
2. <u>**이재능**</u>, 곽근창, "앙상블 기반 스택트 오토엔코더", KR, C-2017-033274 호, 2017.
3. <u>**이재능**</u>, 곽근창, "앙상블 기반 ELM 알고리즘", KR, C-2017-036615 호, 2017.

ABSTRACT

Deep Network Design based on Independent Component Analysis and Application of User Recognition

Lee, Jae Neung
Advisor : Prof. Kwak, Keun Chang, Ph. D.
Dept. of Control and Instrumentation Eng.,
Graduate School of Chosun University

In this paper, we propose a systematic structure of ICANet1, a deep network based on independent component analysis, and develop it into ICANet2 model. The proposed PCANet has smaller computational complexity and faster speed than deep learning. However, there is a disadvantage in that the performance is lowered according to the modification of data. The ICANet algorithm is proposed to solve the problem. There are two structures for the ICANet algorithm. The first stage of the ICANet1 algorithm is a step of mean removal and obtaining a central image. Next, the covariance is obtained by using the center image, the Eigenvector is obtained through the value, and finally, the Eigenvector is used as the input of the independent component analysis filter. Here, centering and whitening steps are performed, and an optimal independent component matrix is obtained by updating the weights. Using this matrix, a padding image obtained through the averaging step and convolution are calculated. This is the first step in ICANet2. ICANet2 does not use Eigenvectors but uses PCA feature vectors. To remove the correlation among the PCA feature vectors, the ICA algorithm is used to find that are statistically independent. The use of statistically independent vectors can help overcome the disadvantages of the PCA and improve the performance. The second stage of the ICANet is almost identical to the first stage, and because it uses the data from the first stage, only the number of dimensions is reduced. It is used as an input to the output stage using the resultant product data from Step 2. Here, the feature values of the independent component analysis network can be obtained by using the hashing code step and the histogram. The feature values obtained from the final histogram are vectorized and used as an input to the classifier. By using ICANet, the performance is lower than that of deep learning, but it is expected to be superior to recognition speed, especially for mobile. The classifier shows performance using Euclidean distance, squared Euclidean distance, city block distance, Minkowski distance, cosine distance, correlation distance, and Spearman distance, SVM. To verify the performance of the proposed method, we used the FERET face, the CU-FACE, the CU-ECG, and the noise ECG database. Experimental results show that ICANet1 and 2 have better performance than PCANet for Dup1 and Dup2 data affected by time. In addition, when noise ECG data were used, it showed better performance than PCANet. This proved the validity of the ICANet algorithm.