Received November 28, 2018, accepted December 7, 2018, date of publication December 11, 2018,
date of current version January 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2886294

Image Noise Level Estimation for Rice Noise Based on Extended ELM Neural Network Training Algorithm

XIAOHUI YANG[1], KAIWEI XU[1], SHAOPING XU[1], AND PETER XIAOPING LIU[1,2], (Fellow, IEEE)
[1]College of Information Engineering, Nanchang University, Nanchang 330031, China
[2]Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada

Corresponding author: Shaoping Xu (xushaoping@ncu.edu.cn)

This work was supported by the National Science Foundation of China under Grant 51765042, Grant 61463031, Grant 61662044, and Grant 61862044.

ABSTRACT The estimation of image noise level is a critical task for image denoising or super-resolution reconstruction. Mathematical methods like patch-based or model-based methods, suffer from the sensitivity of the selection of homogeneous regions or the selection of a proper statistic model, leading to inaccurate estimation, especially in signal-dependent noise cases, such as Rice noise. Ordinary, fully connected networks often suffer from the over-fitting problem, restricting their usage for realistic images. This article proposes a deep-learning-based algorithm by building a deep neural network, and train it by using the evolutionary genetic algorithm and extreme learning machine (ELM) algorithm extended into Hinton's dropout framework. By combining the evolutionary genetic algorithm and the proposed extended ELM algorithm, comparative results are obtained, showing higher accuracy and better stability than several state-of-the-art algorithms.

INDEX TERMS Noise level estimation, deep learning, convolution neural network, extreme learning machine, dropout.

I. INTRODUCTION

Image noise level estimation (NLE) is an important technique in computer image processing because of its huge importance in object detection [1], [2], denoising [3], [4], and super-resolution [5] and it has been studied for several decades [6], [7]. Normal single image denoising methods typically need pre-known arguments of the noise distribution model [8], [9], which has a huge impact on the performance of the denoising algorithm, so NLE becomes a key point. Typical NLE approaches proposed in recent years can be roughly divided into filter-based, patch-based and model-based.

As the earliest group of NLE approaches, filter-based techniques apply filters in transform domain or spatial domain, and then the noise level of a single image is obtained by exploiting the difference between filtered image and the original image.

For transform domain, Khmag Asem's work pointed out that undecimated wavelet-based image reconstruction for denoising methods may bring pseudo-Gibbs phenomena and it solved this issue by the cycle spinning technique [10]. Pyatykh *et al.* [11] presented principal component analysis (PCA) method for noise level estimation, in which the noise level was calculated by variance from covariance matrix of image block with least eigen value. In 2013, Liu and Lin [12] proposed a singular value decomposition (SVD) based algorithm, which reduced the signal impact on NLE problems by taking tail singular values of an image into account. C. Varon proposed a model selection based on distance distributions by using the incomplete Cholesky factorization to solve large-scale kernel principal components analysis problem (kPCA) [13].

For spatial domain, the method proposed in [14] used 3×3 mean and median filters to obtain the image signal and subtracted it from the noisy image to estimate the noise variance from the local variances of the obtained map. The method mentioned in Rank's work [15] first suppressed the effects of the original image signal by filtering the noisy image using difference operators, then all local variances were obtained from the suppressed image and their histogram was computed, the noise variance was finally estimated by statistically evaluating the histogram of local variances.

2169-3536 © 2018 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

In a similar procedure, the method proposed by Corner and his team [16] applied Laplacian and gradient masks to the noisy image to find image edges, then it subtracted the edge map from the filtered image and calculated local variances from the obtained image, as the final step, it took the maximum noise variance as the estimated one.

In patch-based methods, an image was divided into small blocks or patches and the noise variance are then evaluated from homogeneous blocks which contain the lowest variances. Hence, noise estimation performance depends on the input image and the noise level. In other words, these methods may overestimate low noise levels or underestimate the higher ones. To tackle this problem, recent patch-based methods apply principal component analysis (PCA) to the low-rank patches of images and image textures. Methods described in [17]–[22] follow this general idea: by selecting the patches which are considered to be homogeneous patches, signal influence on noise in these patches can be regarded as negligible, then noise distribution parameters can be estimated. For the case of additive white Gaussian noise (AWGN), the selection of homogeneous patches significantly affects the accuracy of noise variance estimation. Tai and Yang [21] tackled this issue by using Sobel filter to find structure edges separating homogeneous patches. However, this edge detection method doesn't work very well with fine-structures and actually sensitive to noise model and level. The method of [19] adaptively combines a filter-based approach with a patch-based scheme to update estimated noise variance

Model-based methods studied the statistical characteristics of data in the pixel domain or the transform domain of an image and described the image by an appropriate statistical model. In [23] and [24], the image data was first transformed into a low-dimensional representation by using subspace learning algorithms, the noise could then be effectively approximated and reduced in the projection space. Work described in [25] was based on the assumption that wavelet coefficients in the diagonal sub-band are dominated by noise, but the estimation results achieved by this approach were usually higher than ground-truths because coefficients in diagonal sub-band were dominated not only by noise but also by image details.

Although most work about noise level estimation target on Gaussian model, which is signal-independent, there is still an important case, Rice noise, which is a more realistic signal-dependent noise model in medical images, being studied for a few years. A work proposed by Lauwers Lieve trying to tackle estimation problem of Rice distribution using Bayesian approach [17]. In the same year, Maitra Ranjan and his team proposed a method to estimate the noise under the mixed model of Rice distributions with common noise parameter [20]. In 2013, Liu *et al.* [22] tried to solve NLE problem of SAR images, the model used in this paper was Rayleigh distribution, which is a specialized case of Rice noise. Peng and Zhao [26] extended their work into more complex Cauchy-Rayleigh mixture model using the expectation-maximization(EM) algorithm. In the next year, Zanetti *et al.* [27] used a restricted version of EM to fit into NLE problem under the Rayleigh-Rice mixture noise model. With the model getting more and more complex, the mathematical analysis also gets more complex and it's much harder to deal with NLE problems. So another methodology comes into play, the machine learning methods, which learn from big samples of pre-given images and noise levels to estimate the noise level of unknown images.

With recent development in machine learning technology, lots of almost-not-solvable problems are now studied. As for NLE problems, many denoising techniques were performed by the artificial neural network, which could produce high-quality results with fast real-time operation [28]–[30]. In [28], the evolutionary algorithm was introduced into NLE problems with accurate and stable results. Work described in [31] introduced a multi-layer perceptron based noise level estimation method using SVD as its network's input, demonstrating the ability of neural networks to estimate noise. But this method could only be applied to Gaussian noise and it only used singular values as the network's input, which led to unacceptable information loss and poor performance for images with fine features. To tackle these issues, techniques described in [32]–[34] used convolution network and showed promising results, while they still suffered from the over-fitting problem. The dropout layer introduced into network structure was a simple but efficient tool used to reduce over-fitting but coming with drawbacks: 2-3 times longer training time [35], [36]. L. Wan proposed a generalized framework called ''DropoutConnect'' for large fully-connected neural network [37] Another algorithm used to avoid over-fitting is the ELM algorithm, which had shown its effectiveness and time-saving [38]–[40].

As talked above, mathematical analysis based method including patch and model-based NLE algorithms severely rely on the similarity between image patches or accurate statistical model of image noise. These conditions often can't be achieved in real life because real-life images are not always simple textures and image noise produced by real-life cameras don't exactly follow the assumed distribution. Machine learning based methods don't rely on strict mathematical analysis but on the training data, if the training data doesn't have too much diversity or of small-scale, it will suffer the over-fitting problem. In order to take advantage of not relying on specific noise model but reduce over-fitting, we build a deep neural network with both convolution and dropout layers(dropout layer resides between the output node and a fully-connected layer), which extract and process features from noisy images to get more accurate and stable results. The method we used for network training is also composed of two parts, the first part is evolutionary genetic algorithm, which is used to train convolution kernels of the convolution layer, the second part is the Extended-ELM algorithm, which is based on the basic ELM algorithm but can apply to the proposed network with dropout layer.

The remain part of this paper is organized as follows: Section II describes the noise model; The design of network's structure is introduced in Section III; The proposed training algorithm is presented in Section IV; Comparative experiments are done and results compared with other state-of-art methods are shown in Section V; We discuss the data we collect in Section VI; conclusions are given in Section VII.

II. NOISE MODEL

For every pixel x in an $M \times N$ image, luminance of this pixel $z(x)$ can be considered as a random variable. Especially in magnetic resonance images, pixel data follow the Rice distribution,

$$z(x) \sim Rice(\nu, \sigma) \tag{1}$$

where $x \in [0, M-1] \times [0, N-1] \subset Z^2$. The probability density function(PDF) of Rice distribution can be written as

$$p(z) = \frac{z}{\sigma^2} e^{-\frac{z^2+\nu^2}{2\sigma^2}} I_0(\frac{z\nu}{\sigma^2}) \tag{2}$$

where ν and σ are parameters of PDF and $I_0 = \sum_{m=0}^{\infty} \frac{(x/2)^{2m}}{m!\Gamma(m+1)}$.

Noticing that pixels don't follow the same distribution because the distribution argument varies with pixel location, which is, in fact, a function of the noiseless value of the given pixel.

The mean μ and variance s of this random variable can be written as

$$\mu = \sigma\sqrt{\frac{\pi}{2}} L(-\frac{\nu^2}{2\sigma^2}) \tag{3}$$

$$s^2 = 2\sigma^2 + \nu^2 - \frac{\pi\sigma^2}{2} L^2(-\frac{\nu^2}{2\sigma^2}) \tag{4}$$

where $L(x) = e^{\frac{x}{2}}[(1-x)I_0(-\frac{x}{2}) - xI_1(-\frac{x}{2})]$.

As formula (3) and (4) show, the variance is dependent on noiseless data, which makes traditional PCA or SVD based methods hard to use. In this paper, we try to build a deep neural network to get the parameter σ from one single given image.

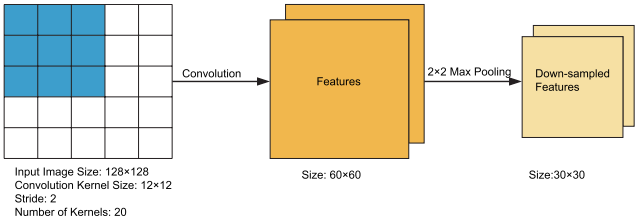


FIGURE 1. Convolution neural network structure.

III. NEURAL NETWORK STRUCTURE DESIGN

The proposed network is composed of a convolution network followed by a fully-connected network with dropout layer. The convolution part is plotted in Fig 1, the fully-connected parted is plotted in Fig 2.

As shown in Fig 1, the input image is convoluted to extract features, the number and size of features we get depend on the size of input images, number of kernels used, size of kernels and the stride, here we use 20 kernels of size 12×12, with stride equals to 2. Here we only use one convolution layer because more layers lead to more computation time but don't give much observable improvement. The pooled features are then flattened and then applied by ReLU as activation function:

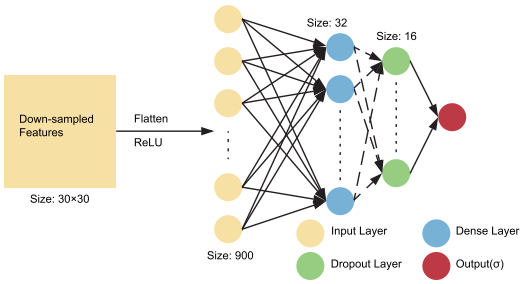


FIGURE 2. Fully-connected network with dropout.

$$ReLU(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{5}$$

Here we don't use the more common sigmoid function to avoid vanishing gradient problem during the training progress. The result is fed into the fully-connected network, as shown in Fig 2. Depending on the image size of the dataset used for training, the number of input neurons can vary. The input layer is fully connected to the dense layer shown in Fig 2 as blue color, this layer can be set to use any number of neurons, here we set it to 32 for 128×128 images. The dense layer is followed by a dropout layer, in which each neuron only has a probability(less than 1) to connect to a neuron of the former dense layer. For 128×128 images, we use 16 neurons for this layer. The detailed analysis of this dropout layer will be discussed in section IV.

IV. TRAINING ALGORITHMS

A. EVOLUTIONARY GENETIC ALGORITHM

Consider a convolution neural network with $H \times L$ input size with N number of $k \times k$ kernels, then for each input image $X_j \in R^{H \times L}$ the output of the convolution(and pooling) layer can be written as:

$$Y_j = N(X_j, K) = Flatten(X_j * K_1, X_j * K_2, \cdots, X_j * K_N) \tag{6}$$

where $K_i \in R^{k \times k} (i = 1, 2, \cdots, N)$ are the kernels used for convolution. *Flatten* is a function that fatten two-dimension images into one-dimension vectors row by row.

Suppose we have M images as training input, then $j = 1, 2, \cdots, M$

The goal of the training algorithm is to minimize this function by the parameter K.

$$K = \min_K E(X, Y) = \min_K \sum_{j=1}^{M} (N(X_j, K) - Y_j)^2 \tag{7}$$

For the training of the convolution part of our network, we use the evolutionary genetic algorithm. Genetic algorithms simulate the process of biological evolution of chromosomes. Chromosomes here means possible values of the network parameter. As we are using this algorithm for convolution network, chromosome values represent the convolution kernel values used for the convolution operation on images. Firstly, randomly initialize C number of possible W (flattened all the kernel values as the chromosome) and set P_s, r as selection and mutation rate, which will be used in the algorithm, we train the kernels of the convolution layer by the following steps:

Step 1: Set *iter* to 0.

Step 2: For every possible W, calculate a fitness f_g for it using:

$$f_g = -\sum_{j=1}^{M}(N(\vec{X}_j, W) - \vec{y}_j)^2 \tag{8}$$

where λ is a positive argument for tweaking our algorithm.

Step 3: The selection process. Choose $(1 - P_s) \times N_c$ chromosomes with the biggest fitness value, these chromosomes will be used as next generation without any change. For every two of the rest chromosomes $\overrightarrow{z}$ and $\overrightarrow{z'}$, we do a crossover using:

$$\widetilde{z}_i = a_i z_i + (1 - a_i) z'_i \tag{9}$$

$$\widetilde{z'}_i = a_i z'_i + (1 - a_i) z_i \tag{10}$$

where a_i are random numbers in [0,1] (crossover rates are random).

Step 4: The mutation process. For each chromosome, elements of it will has a small probability to slightly change its value, that is $\widetilde{z}_i = z_i + 0.05 z_i(2r - 1)$, where r is a random number in [0,1] and generated once for every element of each chromosome.

Step 5: increase *iter* by 1.

Step 6: If iter get to the maximum iteration count, then the chromosome with the greatest fitness value within current generation is used as the final argument of the fully-connected network, else we repeat from step2.

B. ASSUMPTION OF ELM ALGORITHM

After the genetic approach mentioned above, ELM algorithms are used for training of the fully-connected part, dropout technology is used to overcome the slow-computation and trapped in local minima problem [13].

Let us denote by $\{x_i, c_i\}_{i=1,\ldots,N}$ a set of N vectors $x_i \in \mathbb{R}^D$ and the corresponding class labels $c_i \in \{1, \ldots, C\}$ that can be used to train a single layer feed-forward neural network(SLFN) network consisting of D input (equal to the dimensionality of x_i), L hidden and C output (equal to the number of classes involved in the classification problem) neurons. The number L of hidden layer neurons is much larger than C, i.e., $L \gg C$. In ELM-based approaches, the network input weights $W_{in} \in \mathbb{R}^{D\times L}$ and the hidden layer bias values $b \in \mathbb{R}^L$ are randomly chosen, while the network output weights $W_{out} \in \mathbb{R}^{L\times C}$ are analytically calculated, as subsequently described.

Let us denote by v_j, w_k, w_{kj} the j-th column of W_{in}, the k-th row of W_{out} and the j-th element of w_k, respectively. Given an activation function f for the network hidden layer and using a linear activation function for the network output layer, the response $o_i = [o_{i1}, \ldots, o_{iC}]^T$ of the network corresponding to x_i is calculated by:

$$o_{ik} = \sum_{j=1}^{L} w_{kj} f(v_j, b_j, x_i) \tag{11}$$

By storing the network hidden layer outputs $\phi_i \in \mathbb{R}^L$ corresponding to all the training vectors $x_i, i = 1, \ldots, N$ in a matrix $\Phi = [\phi_1, \ldots, \phi_N]$, the network response for all the training data $O \in \mathbb{R}^{C\times N}$ can be expressed in a matrix form as:

$$O = W_{out}^T \Phi \tag{12}$$

where

$$\Phi = \begin{bmatrix} f(v_1, b_1, x_1) & \cdots & f(v_1, b_1, x_N) \\ \vdots & \ddots & \vdots \\ f(v_L, b_L, x_1) & \cdots & f(v_L, b_L, x_N) \end{bmatrix}$$

C. REGULARIZED EXTREME LEARNING MACHINE

A regularized version of the ELM algorithm(RELM) that allows small training errors and tries to minimize the norm of the network output weights W_{out} has been proposed in [41].

In RELM, the training problem is modeled as a constrained minimization problem of W_{out}:

$$W_{out} = \min_{W_{out}} \Im_{RELM} \tag{13}$$

where $\Im_{RELM} = 0.5\|W_{out}\|_F^2 + 0.5c\sum_{i=1}^{N}\|\xi_i\|_2^2$.

Constrained by

$$W_{out}^T \phi_i = t_i - \xi_i, \quad i = 1, \ldots, N, \tag{14}$$

where $\xi_i \in \mathbb{R}^C$ is the error vector corresponding to x_i and c is a parameter denoting the importance of the training error in the optimization problem, satisfying $c > 0$.

By substituting (14) into (13), we can get this formula:

$$W_{out} = (\Phi\Phi^T + \frac{1}{c}I)^{-1}\Phi T^T \tag{15}$$

where $I \in \mathbb{R}^{L\times L}$ is the identity matrix and $T = [t_1, \ldots, t_N]$ is the target matrix composed of target training vectors.

D. DROPOUT-ELM ALGORITHM

Now we try to introduce Hinton's Dropout-ELM algorithm [35].

In Dropout-ELM, the training problem can be considered as a minimization problem, which is the W_{out} to satisfy the following minimize model:

$$\Im_1 = \{0.5tr(W_{out}^T S W_{out}) + 0.5c\sum_{i=1}^{N}\|\xi_i\|_2^2\} \tag{16}$$

where $S \in \mathbb{R}^{L\times L}$ is a matrix describing the relationship of the training data that are subject to minimization. Here this matrix S describes the within-class scatter of the training data representation in the ELM space.

$$W_{out} = \min_{W_{out}} \mathfrak{J}_1 \tag{17}$$

constrained by:

$$W_{out}^T \phi_i = t_i - \xi_i, \quad i = 1, \ldots, N, \tag{18}$$
$$W_{out}^T \phi_i = W_{out}^T \phi_{i,t}, \quad i = 1, \ldots, N,\ t = 1, \ldots, N_T \tag{19}$$

where $\phi_{i,t}$ expresses the hidden layer output for the training vector x_i, after applying Dropout for the training epoch t.

Let's set $\tilde{\phi}_{i,t} = \phi_i - \phi_{i,t}$, this constrain can be written as $W_{out}^T(\phi_i - \phi_{i,t}) = 0$, so we can directly calculate $\tilde{\phi_{i,t}}$ by

$$\tilde{\phi}_{i,t} = \tilde{m}_{i,t} \circ \phi_i \tag{20}$$

where $\tilde{m}_{i,t} \in \mathbb{R}^L$ is a binary mask vector with each element being equal to 1 with probability $(1-p)$ and equal to 0 with probability p.

By substituting (18) into $\mathfrak{J}_1$ with respect to (19), we get

$$\mathfrak{J}_{1,D} = 0.5tr(W_{out}^T S W_{out}) + 0.5c\|W_{out}^T \Phi - T\|_F^2 + \frac{\lambda}{2N_T}\sum_{t=1}^{N_T}\|W_{out}^T \tilde{\Phi}\|_F^2 \tag{21}$$

where $\tilde{\Phi}_t = [\tilde{\phi}_{1,t}, \ldots, \tilde{\phi}_{N,t}]$ and λ is a parameter denoting the importance of the Dropout regularizer in the optimization problem, satisfying $\lambda > 0$. By deter the min point of $\mathfrak{J}_{1,D}$ with respect to W_{out}, the network output weights are written as:

$$W_{out} = (\Phi\Phi^T + \frac{1}{c}S + \frac{\lambda}{c}R_1)^{-1}\Phi T^T \tag{22}$$

where $R_1 = \frac{1}{N_T}\sum_{t=1}^{N_T}\tilde{\Phi}_t\tilde{\Phi}_t^T$.

Consider the extreme case that training process has enough (e.g. $N_T > 100$) steps, we consider N_T as $N_T \to \infty$, then R_1 will converge to its expection:

$$R_1 = E[\frac{1}{N_T}\sum_{t=1}^{N_T}\tilde{\Phi}_t\tilde{\Phi}_t^T] \tag{23}$$

Let $p = [(1-p), \cdots, (1-p)]^T \in R^L$ denoting a vector having elements expressing the probability that the j-th element of ϕ_i will not survive, then R_1 can be written as:

$$R_1 = (\Phi\Phi^T) \circ P \tag{24}$$

where $P = [(pp^T) \circ (11^T - I)] + [(p1^T) \circ I]$ and $1 \in R^L$ is the vector of 1. Noting that since $R_1 \in R^{L\times L}$, overall computational complexity of $O(L^3)$. Finally, by combining (24) and (22) W_{out} can be given by following formula:

$$W_{out} = \left(\left[\Phi\Phi^T\right] \circ \left[I + \frac{\lambda}{c}P\right] + \frac{1}{c}S\right)^{-1}\Phi T^T \tag{25}$$

E. PROPOSED EXTENDED-ELM ALGORITHM

The proposed training algorithm is called Extended-ELM, which is a more complex optimization problem. Compared to dropout-ELM, the expression $\mathfrak{J}_2$ and W_{out} argument to be minimalized here actually stay the same, but constriction is changed in Extended-ELM. Let's re-write them:

$$\mathfrak{J}_2 = \{0.5tr(W_{out}^T S W_{out}) + 0.5c\sum_{i=1}^{N}\|\xi_i\|_2^2\} \tag{26}$$

Our goal is to get the W_{out}

$$W_{out} = \min_{W_{out}} \mathfrak{J}_2 \tag{27}$$

Our W_{out} is constrained by (28) and (29), notice that (29) is different from (19).

$$W_{out}^T \phi_i = t_i - \xi_i, \quad i = 1, \ldots, N, \tag{28}$$
$$W_{out}^T \phi_i = (M_{i,t} \circ W_{out})^T \phi_i, \quad i = 1, \ldots, N,\ t = 1, \ldots, N_T \tag{29}$$

Like we did in last section, we substitute (28) in $\mathfrak{J}_2$ and taking the equivalent dual problem with respect to (29), we obtain:

$$\mathfrak{J}_{2,D} = 0.5tr(W_{out}^T S W_{out}) + 0.5c\|W_{out} - T\|_F^2 + \frac{\lambda}{2N_T}\sum_{t=1}^{N_T}\sum_{i=1}N\|(M_{i,t} \circ W_{out})^T\phi_i\|_2^2 \tag{30}$$

When we have enough training steps($N_T \to \infty$), we have

$$\frac{1}{N_T}\sum_{t=1}^{N_T}(M_{i,t} \circ W_{out})(M_{i,t} \circ W_{out})^T = (W_{out}W_{out}) \circ P \tag{31}$$

By (31), we have

$$\mathfrak{J}_{2,D} = 0.5tr(W_{out}^T S W_{out}) + 0.5c\|W_{out} - T\|_F^2 + \frac{\lambda}{2}\sum_{t=1}^{N_T}tr(D_{\phi_i}PD_{\phi_i}W_{out}W_{out}^T) \tag{32}$$

where $D_{\phi_i} = diag(\phi_i)$.

By determining the saddle point of $\mathfrak{J}_{2,D}$ with respect to W_{out}, the network output weights are given by:

$$W_{out} = (\Phi\Phi^T + \frac{1}{c}S + \frac{\lambda}{c}R_2)^{-1}\Phi T^T \tag{33}$$

where $R_2 = \sum_{i=1}^{N} tr(D_{\phi_i}PD_{\phi_i}W_{out})$.

After we get the network output weights W_{out}, the response of a given input vector $x_l \in \mathbb{R}^D$ is given by:

$$o_l = W_{out}\phi_l \tag{34}$$

where ϕ_l is the hidden layer for x_l.

V. EXPERIMENT AND RESULT ANALYSIS

A. NETWORK TRAINING PROCESS

The proposed neural network is built and initialized randomly in MATLAB R2017b, learning process is conducted by parallel computing system using 4 cores of Intel Core i7-6700 and 8GB memory on Windows 7(No GPU is involved). Learning base rate is constantly set to 0.1 to reduce computational cost.

Due to the large scale of training dataset, images are not sent to the network as a whole big matrix but sent batch by batch. A batch is fed into the network to get rough parameters of network, after this, we use the roughly trained network to train the next batch to improve the arguments of the network, so that we can reduce the computation cost of the training process.

In order to enforce the relation between batches, we take $4 \sim 5\%$ images from the previous batch from the previous batch into the current batch. The training process is composed of the genetic evolution algorithm for the convolution part and then the proposed Extended-ELM algorithm [35] for optimizing the arguments of our network.

B. DATASETS USED FOR TRAINING

The proposed method was trained using USPTex(surface textures, 128×128 pixel size) [42], STL-10(real-life object pictures, 96×96 pixel size) [43] and AVIRIS(huge-size aerial pictures, all images reshaped to 512×512 pixel size) [44] datasets and tested on them. Each of the datasets is composed of a larger training subset and a smaller testing subset. Training data are generated by adding Rice noise with random noise level($\sigma = 5 \sim 30$, continuously random) to every image of the training subset to form repeat-ability.

For USPTex dataset, it contains 191 textures, each ships with 12 varieties with different luminance and slightly different position offset (2292 images in total). We set 500 images aside as testing dataset for validation and do our training on the else.

For AVRIS dataset, it contains many texture details and is very useful to test our NLE algorithm on application scenarios like pattern recognition or super-resolution reconstruction because of its relatively big size. It contains 10000 images, and we use 7500 of them for training and the rest for testing.

For STL-10 dataset, it's taken from real-life pictures and unlike commonly used CIFAR-10, large enough in size to show a more realistic and objective result of NLE methods. It comes with 8000 images in the training set and 5000 images in the testing set.

Some sample images from training datasets and their noisy version with different noise level are shown in Fig 3.

C. TRAINING CURVES AND VALIDATION TEST

Diagram of the training process of the training accuracy and loss function with iteration number is plotted in Fig 4 with orange dotted line representing accuracy and blue solid line representing the loss function.

FIGURE 3. Fully-connected network with dropout.

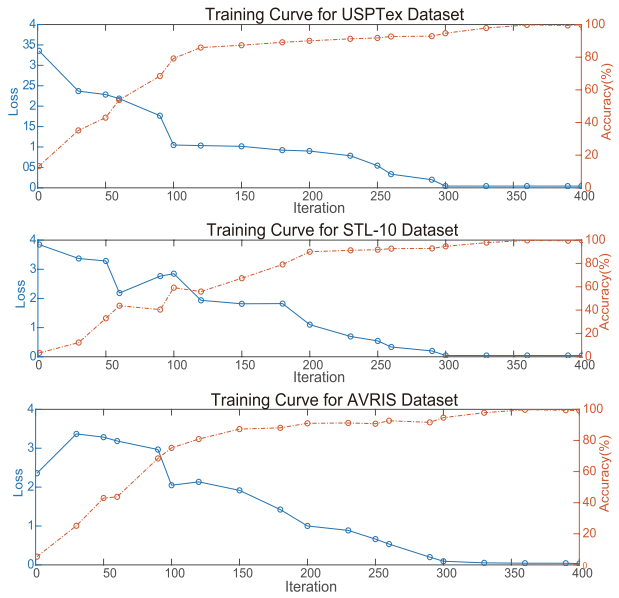


FIGURE 4. Diagram of network training process.

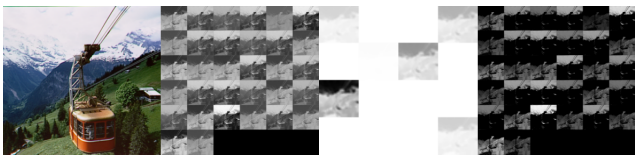


FIGURE 5. Example of feature extraction.

This cable car image is fed into the trained network as an example to show extracted features by different layers of our proposed neural network, as shown in Fig 5 below.

We did some comparative experiments with four patch- or model-based methods, which are LiuWei's image singular value based analysis method SVD [12], C.Varon's kernel principal component analysis method kPCA [13], Shin's adaptive Gaussian filtering based method AGF [19], Tai's adaptive edge detection approach AED [21], Aditya's LR algorithm [45] and Mostafa's DCT-DWT [46] method.

To verify the performance of those comparative methods and our proposed method, we use the performance metrics defined as following. Since we are solving a NLE problem, we add Rice noise with $\sigma = 5 \sim 30$ to each image in the original dataset, each image in the dataset will be repeatedly used for r times, so we will have r different levels of noise for

the same image, which is meant to provide more reliability of our method. After doing this, we get W different images with different noise levels out of a dataset, then we can define the mean and standard deviation of the noise level errors as the following:

$$\mu_{err} = \frac{1}{W \times H} \sum_{w=1}^{W} \sum_{h=1}^{H} err(w, h), \tag{35}$$

$$\sigma_{err} = \sqrt{\frac{1}{W \times H} \sum_{w=1}^{W} \sum_{h=1}^{H} (err\,(w, h) - \mu_{err})^2}, \tag{36}$$

where $err(w, h) = \sigma - \sigma_{est}$

The μ_{err} and σ_{err} can be used to evaluate the accuracy and stability of a certain noise level estimation method. We also calculate $\Psi_{err} = \sqrt{\mu_{err}^2 + \sigma_{err}^2}$ as the overall measurement of an NLE algorithm.

To see the performance of NLE algorithms more meticulously and accurately, we also compare peak signal-to-noise ratio (PSNR) error function based metrics between NLE algorithms, which are defined as:

$$\mu_{psnr} = \frac{1}{W \times H} \sum_{w=1}^{W} \sum_{h=1}^{H} psnr(w, h) \tag{37}$$

$$\sigma_{psnr} = \sqrt{\frac{1}{W \times H} \sum_{w=1}^{W} \sum_{h=1}^{H} \left(psnr\,(w, h) - \mu_{psnr}\right)^2} \tag{38}$$

$$\Psi_{psnr} = \sqrt{\mu_{psnr}^2 + \sigma_{psnr}^2} \tag{39}$$

where $psnr(w, h) = 10 \times \left(lg\frac{255^2}{\sigma^2} - lg\frac{255^2}{\sigma_{est}^2}\right)$.

TABLE 1. Test result on USPTex dataset.

True noise level	Method	μ_{err}	σ_{err}	Ψ_{err}	μ_{psnr}	σ_{psnr}	Ψ_{psnr}
5	SVD [12]	0.389	0.564	0.685	0.669	0.851	1.082
	kPCA [13]	0.401	0.956	1.037	0.791	0.459	0.915
	AGF [19]	0.675	1.115	1.303	0.811	0.965	1.261
	AED [21]	0.953	0.496	1.074	1.311	1.290	1.839
	LR [45]	0.389	0.479	0.617	0.549	0.494	0.739
	DCT-DWT [46]	0.392	0.487	0.625	0.543	0.474	0.721
	Proposed	**0.385**	**0.477**	**0.613**	**0.534**	**0.451**	**0.699**
10	SVD [12]	0.322	0.521	0.612	**0.510**	0.774	0.927
	kPCA [13]	0.365	0.877	0.950	0.652	0.593	0.881
	AGF [19]	0.665	1.009	1.208	0.962	1.001	1.388
	AED [21]	1.012	0.503	1.130	0.896	1.035	1.369
	LR [45]	0.327	0.521	0.615	0.522	0.546	0.755
	DCT-DWT [46]	0.335	0.518	0.617	0.532	0.548	0.764
	Proposed	**0.318**	**0.502**	**0.594**	0.521	**0.502**	**0.723**
15	SVD [12]	0.295	0.522	0.600	0.312	0.638	0.710
	kPCA [13]	0.302	0.711	0.772	0.551	0.502	0.745
	AGF [19]	0.562	0.853	1.021	0.729	0.862	1.129
	AED [21]	0.853	0.512	0.995	0.775	0.891	1.181
	LR [45]	0.266	0.467	0.537	0.241	0.501	0.556
	DCT-DWT [46]	0.278	0.500	0.572	0.249	0.498	0.557
	Proposed	**0.260**	**0.459**	**0.528**	**0.240**	**0.473**	**0.530**
20	SVD [12]	0.121	0.432	0.449	0.298	0.594	0.665
	kPCA [13]	0.253	0.641	0.689	0.502	0.517	0.721
	AGF [19]	0.501	0.546	0.741	0.608	0.776	0.986
	AED [21]	0.631	0.455	0.778	0.276	0.469	0.544
	LR [45]	0.122	0.334	0.356	0.285	0.322	0.430
	DCT-DWT [46]	0.121	0.341	0.362	0.287	**0.312**	0.424
	Proposed	**0.119**	**0.331**	**0.352**	**0.271**	0.318	**0.418**
25	SVD [12]	0.375	0.395	0.545	0.113	0.445	0.459
	kPCA [13]	-0.205	0.524	0.563	-0.121	0.264	0.290
	AGF [19]	0.499	0.426	0.656	0.233	0.509	0.560
	AED [21]	-0.521	0.402	0.658	-0.113	0.268	0.291
	LR [45]	0.235	0.231	0.330	0.111	0.197	0.226
	DCT-DWT [46]	0.222	0.237	0.325	0.107	0.190	0.218
	Proposed	**0.211**	**0.229**	**0.311**	**0.097**	**0.189**	**0.212**
30	SVD [12]	-0.102	0.256	0.276	-0.149	0.202	0.251
	kPCA [13]	-0.111	0.442	0.456	-0.127	0.301	0.327
	AGF [19]	0.311	0.326	0.451	0.186	0.362	0.407
	AED [21]	0.679	0.369	0.773	0.179	0.145	0.230
	LR [45]	0.064	0.229	0.238	0.110	0.132	0.172
	DCT-DWT [46]	0.055	0.231	0.237	0.099	0.129	0.163
	Proposed	**-0.054**	**0.217**	**0.224**	**-0.096**	**0.123**	**0.156**

We record these performance metrics in Table 1-3. Bold font emphasis the best result among the tested methods.

TABLE 2. Test result on AVIRIS dataset.

True noise level	Method	μ_{err}	σ_{err}	Ψ_{err}	μ_{psnr}	σ_{psnr}	Ψ_{psnr}
5	SVD [12]	0.433	0.554	0.703	0.651	0.842	1.064
	kPCA [13]	0.393	0.948	1.026	0.785	**0.447**	0.903
	AGF [19]	0.665	1.124	1.306	0.822	0.971	1.272
	AED [21]	0.961	0.493	1.080	1.320	1.250	1.818
	LR [45]	0.402	0.587	0.711	0.658	0.594	0.886
	DCT-DWT [46]	0.382	0.500	0.629	0.745	0.553	0.928
	Proposed	**0.379**	**0.492**	**0.621**	**0.647**	0.459	**0.793**
10	SVD [12]	0.310	0.527	0.611	**0.513**	0.768	0.924
	kPCA [13]	0.359	0.881	0.951	0.649	0.604	0.887
	AGF [19]	0.661	0.995	1.195	0.969	1.003	1.395
	AED [21]	1.009	**0.501**	1.127	0.901	1.041	1.377
	LR [45]	0.403	0.567	0.696	0.533	0.511	0.738
	DCT-DWT [46]	0.328	0.517	0.612	0.528	0.509	0.733
	Proposed	**0.303**	0.505	**0.589**	0.519	**0.497**	**0.719**
15	SVD [12]	0.306	0.522	0.605	0.312	0.638	0.710
	kPCA [13]	0.304	0.716	0.778	0.561	0.512	0.760
	AGF [19]	0.571	0.851	1.025	0.736	0.858	1.130
	AED [21]	0.867	0.509	1.005	0.774	0.903	1.189
	LR [45]	0.299	0.477	0.563	0.266	0.473	0.543
	DCT-DWT [46]	0.284	0.500	0.575	0.261	0.489	0.554
	Proposed	**0.276**	**0.466**	**0.542**	**0.253**	**0.471**	**0.535**
20	SVD [12]	0.128	0.448	0.466	**0.276**	0.616	0.675
	kPCA [13]	0.249	0.682	0.726	0.486	0.508	0.703
	AGF [19]	0.513	0.555	0.756	0.612	0.781	0.992
	AED [21]	0.627	0.461	0.778	0.285	0.473	0.552
	LR [45]	0.126	0.401	0.420	0.293	0.356	0.461
	DCT-DWT [46]	0.127	0.358	0.380	0.287	0.349	0.452
	Proposed	**0.122**	**0.346**	**0.367**	0.286	**0.337**	**0.442**
25	SVD [12]	0.211	0.388	0.442	0.135	0.436	0.456
	kPCA [13]	-0.210	0.524	0.565	-0.125	0.267	0.295
	AGF [19]	-0.495	0.419	0.649	-0.240	0.503	0.557
	AED [21]	0.517	0.402	0.655	0.121	0.261	0.288
	LR [45]	0.210	0.256	0.331	0.094	0.199	0.220
	DCT-DWT [46]	0.213	0.249	0.328	0.087	0.197	0.215
	Proposed	**0.208**	**0.247**	**0.323**	**0.084**	**0.194**	**0.211**
30	SVD [12]	-0.109	0.274	0.295	-0.150	0.211	0.259
	kPCA [13]	0.114	0.455	0.469	0.131	0.302	0.329
	AGF [19]	-0.319	0.346	0.471	-0.196	0.357	0.407
	AED [21]	0.662	0.371	0.759	0.182	0.144	0.232
	LR [45]	-0.116	0.217	0.246	-0.132	0.148	0.198
	DCT-DWT [46]	0.112	0.211	0.239	0.112	0.154	0.190
	Proposed	**-0.097**	**0.209**	**0.230**	**-0.108**	**0.143**	**0.179**

TABLE 3. Test result on STL-10 dataset.

True noise level	Method	μ_{err}	σ_{err}	Ψ_{err}	μ_{psnr}	σ_{psnr}	Ψ_{psnr}
5	SVD [12]	0.390	0.561	0.683	0.677	0.847	1.084
	kPCA [13]	0.412	0.956	1.041	0.788	0.466	0.915
	AGF [19]	0.658	1.124	1.302	0.806	0.974	1.264
	AED [21]	0.968	0.496	1.088	1.291	1.284	1.821
	LR [45]	0.402	0.587	0.711	1.023	0.894	1.359
	DCT-DWT [46]	0.388	0.500	0.633	0.962	0.753	1.222
	Proposed	**0.374**	**0.495**	**0.620**	**0.676**	**0.479**	**0.829**
10	SVD [12]	0.376	0.519	0.641	0.514	0.786	0.939
	kPCA [13]	0.359	0.877	0.948	0.659	0.584	0.881
	AGF [19]	0.674	1.001	1.207	0.972	1.016	1.406
	AED [21]	1.009	**0.489**	1.121	0.886	1.048	1.372
	LR [45]	0.412	0.576	0.708	0.521	0.894	1.035
	DCT-DWT [46]	0.384	0.504	0.634	0.955	0.753	1.216
	Proposed	**0.342**	0.502	**0.607**	**0.511**	**0.536**	**0.741**
15	SVD [12]	0.287	0.534	0.606	0.325	0.647	0.724
	kPCA [13]	0.302	0.711	0.772	0.551	0.516	0.755
	AGF [19]	0.569	0.856	1.028	0.729	0.859	1.127
	AED [21]	0.841	0.509	0.983	0.769	0.879	1.168
	LR [45]	0.277	0.489	0.562	0.256	0.505	0.566
	DCT-DWT [46]	0.289	0.501	0.578	0.244	0.513	0.568
	Proposed	**0.257**	**0.461**	**0.528**	**0.239**	**0.495**	**0.550**
20	SVD [12]	0.135	0.432	0.453	0.284	0.583	0.648
	kPCA [13]	0.248	0.641	0.687	0.513	0.524	0.733
	AGF [19]	0.496	0.546	0.738	0.617	0.702	0.935
	AED [21]	0.762	0.455	0.888	0.296	0.451	0.539
	LR [45]	0.130	0.350	0.373	0.288	0.399	0.492
	DCT-DWT [46]	0.129	0.377	0.398	0.283	0.375	0.470
	Proposed	**0.124**	**0.348**	**0.369**	**0.266**	**0.310**	**0.408**
25	SVD [12]	0.351	0.397	0.530	0.128	0.432	0.451
	kPCA [13]	-0.211	0.536	0.576	-0.127	0.259	0.288
	AGF [19]	0.518	0.431	0.674	0.239	0.502	0.556
	AED [21]	-0.521	0.419	0.669	-0.115	0.254	0.279
	LR [45]	0.223	0.290	0.366	0.119	0.209	0.241
	DCT-DWT [46]	0.239	0.328	0.406	0.128	0.199	0.237
	Proposed	**0.203**	**0.237**	**0.312**	**0.104**	**0.183**	**0.210**
30	SVD [12]	-0.099	0.253	0.272	-0.151	0.215	0.263
	kPCA [13]	-0.118	0.432	0.448	-0.097	0.306	0.321
	AGF [19]	0.325	0.337	0.468	0.152	0.358	0.389
	AED [21]	0.681	0.371	0.776	0.167	0.141	0.219
	LR [45]	0.087	0.238	0.253	0.102	0.126	0.162
	DCT-DWT [46]	0.098	0.228	0.248	0.101	0.123	0.159
	Proposed	**-0.056**	**0.224**	**0.231**	**-0.093**	**0.124**	**0.155**

VI. DISCUSSION

As we can see from the results above, the following conclusions can be summarized:

In terms of noise level estimation algorithm accuracy, which is measured by the μ_{err} and μ_{psnr} evaluation indicators, direct difference between real noise level and estimated value are used instead of absolute difference to show the difference between under-estimation and over-estimation, thus some data in the result tables is negative, indicating the under-estimation. We can see that average estimation errors

generally decrease along with the increase of noise level. Let's take STL-10 dataset as example, our proposed method successfully decrease the μ_{err} by 3.7%($\sigma = 5$), 9.0%($\sigma = 10$), 10.4%($\sigma = 15$), 3.8%($\sigma = 20$), 3.8%($\sigma = 25$), 35.6%($\sigma = 30$) compared to the second-best method, and it also decrease μ_{psnr} by 0.14%($\sigma = 5$), 0.58%($\sigma = 10$) 2.1%($\sigma = 15$), 9.6%($\sigma = 25$), 4.1%($\sigma = 30$) compared to the second-best method. We also notice that in AVIRIS and USPTex datasets, there're rare cases that the proposed method is not always the best, even by those cases, the proposed method still has the second-best result. For images with relatively high noise level, LR [45] and DCT-DWT [46] provide comparable second-best results while they almost always over-estimate.

Regarding the stability of NLE methods, which is measured mainly by the σ_{err} and σ_{psnr} metrics, the proposed method shows its advantage in almost all cases. As seen from Table 1, in the USPTex dataset, the proposed method decrease σ_{err} by 0.41%($\sigma = 5$), 1.8%($\sigma = 10$), 1.7%($\sigma = 15$), 0.90%($\sigma = 20$), 0.86%($\sigma = 25$), 5.2%($\sigma = 30$) compared to the second-best method. The σ_{psnr} shown in the same table also decrease by 1.7%($\sigma = 5$), 8.1%($\sigma = 10$), 0.41%($\sigma = 15$), 1.8%($\sigma = 20$), 0.53%($\sigma = 25$), 4.6%($\sigma = 30$) compared to the second-best method. This result shows the great stability of our proposed method, which is an important demanding in auto continuous image processing, which may need to deal with images with wide range of noise levels. Rare cases mainly appear in AVIRIS and STL-10 datasets at low noise level cases, however in which our proposed method is still offering competitive data.

The test data of AVIRIS dataset, included in Table 2, Ψ_{err} and Ψ_{psnr} are showing the overall general good quality of the proposed method. Our proposed method decrease Ψ_{err} by 1.3%($\sigma = 5$), 3.6%($\sigma = 10$), 3.7%($\sigma = 15$), 3.4%($\sigma = 20$), 1.5%($\sigma = 25$), 3.8%($\sigma = 30$), also Ψ_{psnr} by 10.5%($\sigma = 5$), 1.9%($\sigma = 10$), 1.5%($\sigma = 15$), 2.2%($\sigma = 20$), 1.8%($\sigma = 25$), 5.8%($\sigma = 30$) compared to the second-best method. The Ψ_{err} and Ψ_{psnr} also show the best result of the proposed method, even in relatively smaller image datasets like USPTex or STL-10, based on the size of images in AVIRIS dataset(larger than 512×512), we conclude that the overall performance of the proposed method is superior in average for images of different sizes, and prove the proposed method's ability to deal with large size images from real-life.

VII. CONCLUSION

In this paper, we proposed a network with both convolution and fully-connected layers. The artificial neural network proposed in this paper use convolution layer to extract features from the input image, after pooling, feature images are flattened as input to a fully-connected network(with dropout layer). We use the evolutionary genetic algorithm to train the kernels of our convolution network, and then proposed the Extended-ELM method. This proposed method is based on Hinton's Dropout-ELM [35], but build a more accurate minimization model, then solve this model by taking dropout layer into account. The proposed neural network was tested by three datasets(USPTex, AVIRIS and STL-10), the training process is plotted with the accuracy and loss function with training steps. Experimental result is compared with several recent works, showing that the proposed artificial neural network has the ability of image noise level estimation with Rice noise of wide noise level range. Compared to patch- or model-based methods, our comparative result has shown the proposed neural network structure and proposed training algorithm have higher estimation accuracy, stability with both low and high noise level, and good overall quality.

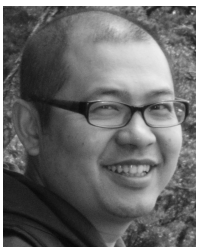
REFERENCES

[1] D. Spulak, R. Otrebski, and W. Kubinger, "Evaluation of PCA, LDA and Fisherfaces in appearance-based object detection in thermal infra-red images with incomplete data," *Procedia Eng.*, vol. 100, pp. 1167–1173, Jan. 2015.
[2] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Trans. Intell. Transp. Syst.*, vol. 1, no. 4, pp. 179–189, Dec. 2000.
[3] H. Zhong, K. Ma, and Y. Zhou, "Modified BM3D algorithm for image denoising using nonlocal centralization prior," *Signal Process.*, vol. 106, pp. 342–347, Jan. 2015.
[4] S. Gai and L. Luo, "Image denoising using normal inverse Gaussian model in quaternion wavelet domain," *Multimedia Tools Appl.*, vol. 74, no. 3, pp. 1107–1124, Feb. 2015.
[5] L. Fang, H. Zhuo, and S. Li, "Super-resolution of hyperspectral image via superpixel-based sparse representation," *Neurocomputing*, vol. 273, pp. 171–177, Jan. 2018.
[6] Y. Sun, L. Pan, C. Li, and B. Liu, "Study on image denoise based on empirical mode decomposition and total variation model," *J. Inf. Comput. Sci.*, vol. 9, no. 17, pp. 5189–5196, 2012.
[7] N. Krishnan, S. Muthukumar, S. Ravi, D. Shashikala, and P. Pasupathi, "Image restoration by using evolutionary technique to denoise Gaussian and impulse noise," in *Mining Intelligence and Knowledge Exploration*. Tamil Nadu, India: Springer, 2013, pp. 299–309. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-03844-5_31
[8] S. Xu, X. Yang, and S. Jiang, "A fast nonlocally centralized sparse representation algorithm for image denoising," *Signal Process.*, vol. 131, pp. 99–112, Feb. 2017.
[9] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 613–627, May 1995.
[10] A. Khmag, A. R. Ramli, S. Al-Haddad, S. Hashim, Z. M. Noh, and A. A. Najih, "Design of natural image denoising filter based on second-generation wavelet transformation and principle component analysis," *J. Med. Imag. Health Inform.*, vol. 5, no. 6, pp. 1–6, Aug. 2015.
[11] S. Pyatykh, J. Hesser, and L. Zheng, "Image noise level estimation by principal component analysis," *IEEE Trans. Image Process.*, vol. 22, no. 2, pp. 687–699, Feb. 2013.
[12] W. Liu and W. Lin, "Additive white Gaussian noise level estimation in SVD domain for images," *IEEE Trans. Image Process.*, vol. 22, no. 3, pp. 872–883, Mar. 2013.
[13] C. Varon, C. Alzate, and J. A. K. Suykens, "Noise level estimation for model selection in kernel PCA denoising," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2650–2663, Nov. 2015.
[14] S. I. Olsen, "Estimation of noise in images: An evaluation," *Graph. Models Image Process.*, vol. 55, no. 4, pp. 319–323, Jul. 1993.
[15] K. Rank, M. Lendl, and R. Unbehauen, "Estimation of image noise variance," *IEE Proc.-Vis., Image Signal Process.*, vol. 146, no. 2, pp. 80–84, Aug. 1999.
[16] B. Corner, R. M. Narayanan, and S. E. Reichenbach, "Noise estimation in remote sensing imagery using data masking," *Int. J. Remote Sens.*, vol. 24, no. 4, pp. 689–702, Jan. 2003.
[17] L. Lauwers, K. Barbé, W. Van Moer, and R. Pintelon, "Estimating the parameters of a Rice distribution: A Bayesian approach," in *Proc. IEEE Instrum. Meas. Technol. Conf.*, May 2009, pp. 114–117.
[18] S. B. Mohan, T. A. Raghavendiran, and R. Rajavel, "Patch based fast noise level estimation using DCT and standard deviation," in *Cluster Computing*. 2018, pp. 1–10. [Online]. Available: https://link.springer.com/article/10.1007/s10586-018-2327-4
[19] D.-H. Shin, R.-H. Park, S. Yang, and J.-H. Jung, "Block-based noise estimation using adaptive Gaussian filtering," *IEEE Trans. Consum. Electron.*, vol. 51, no. 1, pp. 218–226, Feb. 2005.

[20] R. Maitra and D. Faden, "Noise estimation in magnitude MR datasets," *IEEE Trans. Med. Imag.*, vol. 28, no. 10, pp. 1615–1622, Oct. 2009.
[21] S.-C. Tai and S.-M. Yang, "A fast method for image noise estimation using laplacian operator and adaptive edge detection," in *Proc. 3rd Int. Symp. Commun. Control Signal Process.*, Mar. 2008, pp. 1077–1081.
[22] T. Liu, H. Cui, T. Mao, Z. Xi, and J. Gao, "Modeling multilook polarimetric SAR images with heavy-tailed Rayleigh distribution and novel estimation based on matrix log-cumulants," *Sci. China Inf. Sci.*, vol. 56, no. 6, pp. 1–14, Jun. 2013.
[23] X. Peng, Z. Yu, Z. Yi, and H. Tang, "Constructing the L2-graph for robust subspace learning and subspace clustering," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 1053–1066, Apr. 2016.
[24] X. Peng, M. Yuan, Z. Yu, W. Y. Yau, and L. Zhang, "Semi-supervised subspace learning with L2graph," *Neurocomputing*, vol. 208, pp. 143–152, Oct. 2016.
[25] Y.-P. Liu, W. Du, J. Jin, H. Wang, and R. Liang, "Boost image denoising via noise level estimation in quaternion wavelet domain," *AEU-Int. J. Electron. Commun.*, vol. 70, no. 5, pp. 584–591, May 2016.
[26] Q. Peng and L. Zhao, "SAR image filtering based on the cauchy–Rayleigh mixture model," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 5, pp. 960–964, May 2014.
[27] M. Zanetti, F. Bovolo, and L. Bruzzone, "Rayleigh-Rice mixture parameter estimation via EM algorithm for change detection in multispectral images," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5004–5016, Dec. 2015.
[28] J. Tian and L. Chen, "Image noise estimation using a variation-adaptive evolutionary approach," *IEEE Signal Process. Lett.*, vol. 19, no. 7, pp. 395–398, Jul. 2012.
[29] T. Saba, A. Rehman, and G. Sulong, "An intelligent approach to image denoising," *J. Theor. Appl. Inf. Technol.*, vol. 17, no. 2, pp. 32–36, Jul. 2010.
[30] M. Debakla, K. Djemal, and M. Rebbah, "MRI image denoising approach based on TV and neural network filter," in *Proc. Int. Conf. Image Process. Prod. Comput. Sci. (ICIPCS)*, Istanbul, Turkey, Jun. 2015, pp. 23–30.
[31] W. Zhiming and Y. Guobin, "Image noise level estimation by neural networks," in *Proc. Int. Conf. Mater. Eng. Inf. Technol. Appl. (MEITA)*, Guilin, China, 2015. [Online]. Available: https://www.researchgate.net/publication/301432202_Image_Noise_Level_Estimation_by_Neural_Networks
[32] F. Zhang, N. Cai, J. Wu, G. Cen, H. Wang, and X. Chen, "Image denoising method based on a deep convolution neural network," *IET Image Process.*, vol. 12, no. 4, pp. 485–493, Apr. 2017.
[33] D. Cireşan, U. Meier, and J. Schmidhuber. (2012). "Multi-column deep neural networks for image classification." [Online]. Available: https://arxiv.org/abs/1202.2745
[34] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.
[35] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. (2012). "Improving neural networks by preventing co-adaptation of feature detectors." [Online]. Available: https://arxiv.org/abs/1207.0580
[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
[37] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. 30th Int. Conf. Mach. Learn.*, Feb. 2013, pp. 1058–1066.
[38] R. Zhang, Y. Lan, G.-B. Huang, and Z.-B. Xu, "Universal approximation of extreme learning machine with adaptive growth of hidden nodes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 365–371, Feb. 2012.
[39] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.
[40] A. Iosifidis, A. Tefas, and I. Pitas, "DropELM: Fast neural network regularization with Dropout and DropConnect," *Neurocomputing*, vol. 162, pp. 57–66, Aug. 2015.
[41] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.
[42] A. R. Backes, D. Casanova, and O. M. Bruno, "Color texture analysis based on fractal descriptors," *Pattern Recognit.*, vol. 45, no. 5, pp. 1984–1992, May 2012.
[43] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, Jun. 2011, pp. 215–223.
[44] Jet Propulsion Laboratory. *AVIRIS Moffett Field*. Accessed: Apr. 4, 2013. [Online]. Available: https://aviris.jpl.nasa.gov/data/free_data.html
[45] A. K. Das and J. Chandle, "Accurate noise level estimation through singular values and linear regression," in *Proc. IEEE Int. Conf. Eng. Technol. (ICETECH)*, Mar. 2016, pp. 1132–1135.
[46] M. M. Ghazi and H. Erdogan, "Image noise level estimation based on higher-order statistics," *Multimedia Tools Appl.*, vol. 76, no. 2, pp. 2379–2397, Jan. 2017.

XIAOHUI YANG received the B.Sc. M.Sc., and Ph.D. degrees from Nanchang University, Nanchang, China, in 2003, 2006, and 2015, respectively. Since 2006, he has been with the Department of Electronic Information Engineering, School of Information Engineering, Nanchang University, where he is currently an Associate Professor. He has published over 30 research articles. His current interests include intelligent control, process control, fault diagnosis, and stochastic nonlinear systems.

KAIWEI XU received the B.E. degree in material science and engineering and its automation from Nanchang University, Nanchang, China, in 2014, where he is currently pursuing the M.E. degree in electric engineering. His research interests include image processing and neural network.



SHAOPING XU received the M.S. degree in computer application from the China University of Geosciences, Wuhan, China, in 2004, and the Ph.D. degree in mechatronics engineering from the University of Nanchang, Nanchang, China, in 2010. He is currently a Professor with the Department of Computer Science, School of Information Engineering, Nanchang University. He has published more than 30 research articles. His current research interests include digital image processing and analysis, computer graphics, virtual reality, and surgery simulation. He serves as a Reviewer for several journals, including the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT.

PETER XIAOPING LIU (F'19) received the B.Sc. and M.Sc. degrees from Northern Jiaotong University, China, in 1992 and 1995, respectively, and the Ph.D. degree from the University of Alberta, Canada, in 2002. He has been with the Department of Systems and Computer Engineering, Carleton University, Canada, since 2002, and he is currently a Professor and the Canada Research Chair. He is also with the School of Information Engineering, Nanchang University, as an Adjunct Professor. His interests include interactive networked systems and teleoperation, haptics, micro-manipulation, robotics, intelligent systems, context-aware intelligent networks, and their applications to biomedical engineering. He has published more than 280 research articles. He was a recipient of the 2007 Carleton Research Achievement Award, the 2006 Province of Ontario Early Researcher Award, the 2006 Carty Research Fellowship, the Best Conference Paper Award from the 2006 IEEE International Conference on Mechatronics and Automation, and the 2003 Province of Ontario Distinguished Researcher Award. He is a Licensed Member of the Professional Engineers of Ontario (P.Eng), a Senior Member of IEEE, and a Fellow of the Engineering Institute of Canada. He serves as an Associate Editor for several journals, including the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE/ASME TRANSACTIONS ON MECHATRONICS, the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, and the IEEE ACCESS.