Self-adaptive Differential Evolutionary Extreme Learning Machine and Its Application in Facial Age Estimation

Junhua Ku
Department of information engineering
Hainan institute of science & technology Haikou, China
E-mail: kujunhua@163.com

Kongduo Xing
Department of information engineering
Hainan institute of science & technology Haikou, China
E-mail: cogemm@163.com

Abstract—In this paper, Self-adaptive Differential Evolutionary Extreme Learning Machine (SaDE-ELM) was proposed as a new class of learning algorithm for single-hidden layer feed forward neural network (SLFN). In order to achieve good generalization performance, SaDE-ELM calculates the error on a subset of testing data for parameter optimization. Since SaDE-ELM employs extra data for validation to avoid the over fitting problem, more samples are needed for model training. In this paper, the cross-validation strategy is proposed to be embedded into the training phase so as to solve the overtraining problem. Experimental results demonstrate that the proposed algorithms are efficient for Facial Age Estimation.

Keywords-Extreme learning machines; Differential evolution extreme learning machines; Self-adaptive differential evolution extreme learning machines; Facial age estimation

I. INTRODUCTION

Automated age estimation from facial images is one of the most difficult challenges in face analysis [1,2]. It can be very favorable in a number of real life applications such as age-based authorization systems, demographic data mining, business intelligence and video surveillance systems. The difficulty of this task originates from many reasons such as the lack of enough labeled samples to model the aging patterns of subjects, as well as uncontrolled conditions in data collection such as illumination, pose, occlusions and other environmental variables. Aging process is also known to be very subject-dependent, i.e. subjects might differ in terms of aging patterns, resulting in high variations within the samples from the same age.

Recently, a new fast learning neural algorithm for SLFNs, named extreme learning machine (ELM) [3,4], was developed to improve the efficiency of SLFNs. Different from the conventional learning algorithms for neural networks (such as BP algorithms[5]), which may face difficulties in manually tuning control parameters (learning rate, learn-ing epochs, etc.) and/or local minima, ELM is fully auto-matically implemented without iterative tuning, and in theory, no intervention is required from users. Further-more, It was popular for its fast training speed by means of utilizing random hidden node parameters and calculating the output weights with least square algorithm [6-10]. However, in ELM, the number of hidden nodes is assigned a priori, the hidden node parameters are randomly chosen and they remain unchanged during the training phase. Many non-optimal nodes may exist and contribute less in minimizing the cost function. Moreover, in [11] Huang et al. pointed out that ELM tends to require more hidden nodes than conventional tuning-based algorithms [12, 13] in many cases.

Differential evolution (DE) [14] which is a simple but powerful population-based stochastic direct searching technique is a frequently used method for selecting the network parameters [15–17]. In [15], DE is directly adopt-ed as a training algorithm for feed forward networks where all the network parameters are encoded into one population vector and the error function between the network approximate output and the expected output is used as the fitness function to evaluate all the populations. However, Subudhi and Jena [16] have pointed out that using the DE approach alone for the network training may yield a slow convergence speed. Therefore, in [17], a new algorithm named evolutionary extreme learning machine (DE-ELM) based on DE and ELM has been developed for SLFNs. Using the DE method to optimize the network input parameters and the ELM algorithm to calculate the network output weights, DE-ELM has shown several promising features. It not only ensures a more compact network size than ELM, but also has better generalization performance.

However, in the above DE based neural network training algorithms, the trial vector generation strategies and the control parameters in DE have to be manually chosen. For example, the control parameters in DE-ELM are manually selected according to an empirical suggestion and the simple random generation method is adopted to produce the trial vector. As pointed out by many researchers, the performance of the DE algorithm highly depends on the chosen trial vector generation strategy and the control parameters, and inappropriate choices of strategies and control parameters may result in premature convergence or stagnation. Therefore, we propose a novel learning algorithm named self-adaptive evolutionary extreme learning machine (SaDE-ELM) for SLFNs. In SaDE-ELM, the hidden node learning parameters are optimized by the self-adaptive differential evolution algorithm.

The rest of the paper is organized as follows. In section II, a brief introduction to ELM and SaDE are given. In Section III, we introduce model of proposed SaDE-ELM algorithm in detail. In Section IV, we present Performance Evaluation. In section V, we conclude and summarize our results.

II. BACKGROUND

As a novel training algorithm for SLFNs, ELM is very efficient and effective. In this section, we will give a brief review of ELM. In this section, we briefly review ELM and SaDE-ELM approach for ELM is the foundation of SaDE-ELM.

A. *Extreme Learning Machine (ELM)*

For N arbitrary distinct samples $(\mathrm{x}_j, \mathrm{t}_j)$, where $\mathrm{x}_j = [x_{j1}, x_{j2}, \cdots, x_{jn}]^T \in \mathbb{R}^n$, $\mathrm{t}_j = [t_{j1}, t_{j2}, \cdots, t_{jm}]^T \in \mathbb{R}^m$, SLFNs with L hidden nodes and activation function $g(x)$ are

$$\sum_{i=1}^{L} \beta_i \, \mathrm{g}_i(\mathrm{x}_j) = \sum_{i=1}^{L} \beta_i \, \mathrm{g}_i(\mathrm{w}_i \cdot \mathrm{x}_j + b_j) = \mathrm{o}_j \qquad (j = 1, 2, ..., N) \tag{1}$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, ..., w_{in}]T$ is the weight vector connecting the ith hidden node and the input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, ..., \beta_{im}]^T$ is the weight vector connecting the ith hidden node and the output nodes, b_i is the threshold of the ith hidden node, $\mathbf{w}_i \cdot \mathbf{x}_j$ denotes the inner product of w_i and x_j, $g(x)$ is activation function and Sigmoid, Sine, Hardlim and other functions are commonly used. The output nodes are chosen linear in this paper, and $\mathrm{o}_j = [o_{j1}, o_{j2}, ..., o_{jm}]^T$ is the jth output vector of the SLFNs [22].

The SLFNs with L hidden nodes and activation function g(x) can approximate these N samples with zero error. It means $\sum_{j=1}^{L} \left\| \mathrm{o}_j - \mathrm{t}_j \right\| = 0$ and there exist β_i, $\mathbf{w}_i$ and b_i such that

$$\sum_{i=1}^{L} \beta_i \, \mathrm{g}_i(\mathrm{x}_j) = \sum_{i=1}^{L} \beta_i \, \mathrm{g}_i(\mathrm{w}_i \cdot \mathrm{x}_j + b_j) = \mathrm{t}_j \qquad (j = 1, 2, ..., N) \tag{2}$$

The equation above can be expressed compactly as follows:

$$\mathbf{H}\beta = \mathbf{T} \tag{3}$$

Where $\mathrm{H}(\mathrm{w}_1, \mathrm{w}_2, \cdots, \mathrm{w}_L, b_1, b_2, \cdots, b_L, \mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_L)$

$$= [h_{ij}] = \begin{bmatrix} \mathrm{g}(\mathrm{w}_1 \cdot \mathrm{x}_1 + b_1) & \mathrm{g}(\mathrm{w}_1 \cdot \mathrm{x}_1 + b_2) & \cdots & \mathrm{g}(\mathrm{w}_1 \cdot \mathrm{x}_1 + b_L) \\ \mathrm{g}(\mathrm{w}_1 \cdot \mathrm{x}_2 + b_1) & \mathrm{g}(\mathrm{w}_2 \cdot \mathrm{x}_2 + b_2) & \cdots & \mathrm{g}(\mathrm{w}_L \cdot \mathrm{x}_2 + b_L) \\ \vdots & \vdots & & \vdots \\ \mathrm{g}(\mathrm{w}_1 \cdot \mathrm{x}_N + b_1) & \mathrm{g}(\mathrm{w}_2 \cdot \mathrm{x}_N + b_2) & \cdots & \mathrm{g}(\mathrm{w}_L \cdot \mathrm{x}_N + b_L) \end{bmatrix}_{N \times L}$$

$$\beta = \begin{pmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1m} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{L1} & \beta_{L2} & \cdots & \beta_{Lm} \end{pmatrix} \text{ and } \mathrm{T} = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ t_{21} & t_{22} & \cdots & t_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ t_{N1} & t_{N2} & \cdots & t_{Nm} \end{pmatrix}$$

The matric $\mathbf{H}$ is called the hidden layer output matrix of the neural network and the ith column of $\mathbf{H}$ is the ith hidden node output with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$.

By simply randomly choosing hidden nodes and then adjusting the output weights, single hidden layer feedforward networks (SLFNs) work as universal approximators with any bounded non-linear piecewise continuous functions for additive nodes [23]. ELM algorithm claims that the hidden node parameters can be randomly assigned [3,4], then the system equation becomes a linear model and the network output weights can be analytically determined by finding a least-square solution of this linear system as follow

$$\widehat{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{4}$$

Where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$. Then the output function of ELM can be modeled as follows.

$$\phi(\xi) = \gamma(\xi)\beta = \gamma(\xi)\mathbf{H}^{\dagger}\mathbf{T} \tag{5}$$

Moreover, it should be noted that many nonlinear activation and kernel functions can be used in ELM.

B. *Self-adaptive Differential Evolution*

Differential evolution (DE), proposed by Storn and Price in 1995, is a simple yet powerful evolutionary algorithm (EA) [24]. There are three parameters in DE algorithm, which are the population size *NP*, mutation scaling factor *F* and crossover rate *CR*. *NP* is a problem-dependent parameter, while *F* and *CR* are very sensitive to the performance at different stages of evolution. To overcome the limitations of choosing the parameters in DE, Brest et al. proposed a parameter adaptation technique to choose the mutation scaling factor *F* and crossover rate *CR* namely SADE-ELM algorithm which performs better than the basic DE algorithm. In general, SADE algorithm is composed of three main steps: mutation, crossover, and selection [26].

We consider the following optimization problem:

Minimize $f(x_i)$, $x_i \in R_D$

where $x_i = [x_{i1}, x_{i2}, \cdots, x_{iD}]^T, i = 1, 2, \cdots, NP$ is a target vector of D decision variables. During the mutation operation, mutant vector v_i is generated by mutation strategy in the current population:

$$v_i = x_{r1} + F \cdot (x_{r2} - x_{r3}) \tag{6}$$

where $r1, r2, r3$ are mutually exclusive integers randomly chosen in the range [1,*NP*], and $r1 \neq r2 \neq r3 \neq i$. Following mutation, trial vector u_i is generated between x_i and v_i during crossover operation where the most widely used operator is the binomial crossover performed as follows:

$$u_{ij} = \begin{cases} v_{ij}, & if\ (\mathrm{rndreal}(0,1) < \mathrm{CR} \ \text{ or } \ j = j_{\mathrm{rand}}), \\ x_{ij}, & otherwise \end{cases} \tag{7}$$

Where j_{rand} is a integer randomly chosen in the range [1,*D*], and rndreal(0, 1) is a real number randomly generated in (0, 1). Finally, to keep the population size constant during the evolution, the selection operation is used to determine whether the trial or the target vector survives to the next generation according to one-to-one selection:

$$x_i = \begin{cases} u_{i,} & if\,(f(u_i)\,f(x_i)) \\ x_i, & otherwise \end{cases} \tag{8}$$

Where $f(x)$ is the optimized objective function. During the evolution, F and CR are adaptively tuned to improve the performance of DE for each individual

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \cdot F_u & if\,(rand_2 < \tau_1) \\ F_{i,G} & \text{otherwise} \end{cases} \tag{9}$$

$$CR_{i,G+1} = \begin{cases} rand_3 & if\,(rand_4 < \tau_2) \\ CR_{i,G} & \text{otherwise} \end{cases} \tag{10}$$

Where $F_{i;G+1}$ and $CR_{i;G+1}$ are the mutation scaling factor and crossover rate for i individual in G generation respectively, *randj*=1;2;3;4 are randomly chosen from (0, 1), τ_1 and τ_2 both valued 0.1 which is used to control the generation of F and CR, Fl valued 0.1 and Fu is valued 0.9. In the first generation, F and CR are initialized to 0.5.

C. Model of Proposed SADE-ELM Algorithm

Since the ELM generates the input weights and hidden biases arbitrarily which are the basic of calculating the output weights, it may not reach the optimal result in classification or regression. Thus, a hybrid approach integrated self-adaptive differential evolution algorithm and extreme learning machine namely SADE-ELM algorithm to optimize the input weights and hidden biases is able to obtain better generalization performance than ELM algorithm [17].

In SaDE-ELM, we proposed SaDE-ELM for SLFNs by incorporating the self-adaptive differential evolution algorithm [25] to optimize the network input weights and hidden node biases and the extreme learning machine to derive the network output weights.

Given a set of training data and L hidden nodes with an activation function g( ), we summarize the SaDE-ELM algorithm in the following steps.

Step 1. Initialization

A set of *NP* vectors where each one includes all the network hidden node parameters are initialized as the populations of the first generation

$$\theta_{k,G} = \left[\mathrm{w}_{1,k,G}^T, \cdots, \mathrm{w}_{L,k,G}^T, b_{1,k,G}^T, \cdots, b_{L,k,G}^T \right] \tag{11}$$

where w_j and b_j (j = 1, … , L) are randomly generated, G represents the generation and k = 1, 2,…, *NP*.

Step 2. Calculations of output weights and RMSE

Calculate the network output weight matrix and root mean square error (RMSE) with respect to each population vector with the following equations, respectively.

$$\beta_{k,G} = \mathrm{H}_{k,G}^{\dagger}\mathrm{T} \tag{12}$$

$$\mathrm{RMSE}_{k,G} = \sqrt{\frac{\sum_{i=1}^{N} \left\| \sum_{j=1}^{L} \beta_j g(\mathrm{w}_{j,k,G}, b_{j,k,G}, x_i) - t_i \right\|}{m \times N}} \tag{13}$$

Then use the value of RMSE to calculate the new best population vector $\theta_{k,G+1}$ with the following equation.

$$\theta_{k,G+1} = \begin{cases} u_{k,G+1} & if\ (\mathrm{RMSE}_{\theta_{k,G}} - \mathrm{RMSE}_{u_{k,G+1}}) > \varepsilon \cdot \mathrm{RMSE}_{\theta_{k,G}} \\ u_{k,G+1} & if\ \left|\mathrm{RMSE}_{\theta_{k,G}} - \mathrm{RMSE}_{u_{k,G+1}}\right| < \varepsilon \cdot \mathrm{RMSE}_{\theta_{k,G}} \\ & and \left\|\beta_{u_{k,G+1}}\right\| < \left\|\beta_{\theta,G}\right\| \\ \theta_{i,G} & \text{otherwise} \end{cases} \tag{14}$$

where ε is the preset small positive tolerance rate. In the first generation, the population vector with the best RMSE is stored as $\theta_{\text{best},1}$ and $\mathrm{RMSE}\theta_{\text{best},1}$.

All the trial vectors $u_{k,G+1}$ generated at the (G+1)th generation are evaluated using equation(11) .The norm of the output weight $\|\beta\|$ is added as one more criteria for the trial vector selection as pointed out in [13] that the neural networks tend to have better generalization performance with smaller weights.

The three operations mutation, crossover and selection are repeated until the preset goal is met or the maximum learning iterations are completed. At last we calculate the output weigh $\beta = [\beta_{i1}\ \beta_{i2}\ \cdots\ \beta_{iL}]^T$ with equation $\beta = \mathbf{H}^{\dagger}\mathbf{T}$.

III. PERFORMANCE EVALUATION

In this section we describe the different parts of our age estimation pipeline, namely face alignment, feature extraction and model learning. The workflow of our proposed method is illustrated in Fig .1.

The first steps of a human age estimation pipeline are face detection [17, 18] and facial landmark localization]. In this work, we chose to use the Deformable Part Model (DPM) based face detector proposed by Mathias et al. [18], because it finds the location of the face bounding box and gives a good alignment without the need for facial landmark localization. The DPM face detector gives the coordinates of the bounding box (if any face is detected), as well as the detection score. We run the face detector on rotated version of the original image between -60◦ and 60◦ in 5◦ increments, in order to eliminate in-plane rotation. Since some of the images are rotated 90° or upside down, we also try 180°, -90° and 90° rotations. We then take the output with the maximum face score. For the cases where no face is

detected, we register the whole image. ChaLearn Looking at People 2016 - Apparent Age Estimation challenge dataset [19] consists of 7,591 face images collectively labeled by multiple human annotators, therefore the mean μ and the standard deviation σ is provided for each sample. The dataset is split into 4113 training, 1500 validation and 1978 testing samples, where the testing set labels are sequestered. The three subsets have a similar age distribution. Table I presents the number of samples where the DPM face detector was able to detect a face. Table I shows the number of detections on the three subsets.

TABLE I. FACE ALIGNMENT SUMMARY

#	Train	Val	Test
Given	4113	1500	1978
Detected	4016	1462	1920

We used the deep network to extract CNN features from aligned images. The VGG- Face network consists of 37 layers, the final one being a 2622-dimensional softmax layer, trained for the face recognition task. We tried the performance of the final layers and found that the 33rd layer, which is the first (earliest) 4096-dimensional convolution layer, was the most informative one. Therefore we used only the features from this layer in model learning. The baseline regression performances (without any grouping) of the best layers are shown in Table II

TABLE II. COMPARISON OF DIFFERENT LAYERS OF VGG-FACE

Layer	Num. features	ε_{val}	MAE_{val}
32	25088	0.4284	4.68
33	4096	0.4021	4.35
35	4096	0.4150	4.48
37	2622	0.4066	4.38

We then normalize each feature vector by dividing it to its Euclidean norm. We have tried various normalization options prior to L2 normalization and saw that none of them was improving the normal score; therefore we decided to use only L2 normalization for the final system. Performance with various normalization options for the best layers is shown in Table III.

In our experiments, we tried combinations of alternative feature normalization methods, including the sigmoid function, power normalization by 2 (i.e. setting the absolute value of each feature to its square root), min-max normalization of each feature to [−1, 1] among samples, and z- normalization. For min-max and z-normalization, we learn the parameters from training folds and apply them to the test fold.

TABLE III. Validation set performance with different normalization options

Norm. Type	Lay r 33		Lay r 37	
	ε	MAE	ε	MAE
Nonorm	0.4487	4.91	0.4403	4.79
L2	0.4021	4.35	0.4066	4.38
Pow. + L2	0.4028	4.32	0.4079	4.44
Sig. + L2	0.4152	4.49	0.4137	4.46
MM+L2	0.4355	4.77	0.4301	4.63
Z+L2	0.4102	4.48	0.4036	4.33
MM +Sig. +	0.4861	5.46	0.4652	5.12
Z + Sig. + L2	0.4220	4.59	0.4164	4.51
M 1 + Pow. + L2	0.4565	5.01	0.4438	4.88
Z + Pow. + L2	0.4083	4.43	0.4078	4.37

Now, we introduce the evaluation criteria in our experiments as following.

Mean Absolute Error (MAE): A standard way of measuring the accuracy of a regressor is to average the absolute deviation of each sample's label from its estimated value. More formally, MAE of a given dataset is calculated as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} | x_i - \hat{x}_i | \tag{15}$$

where x_i is the true label i.e. the average of apparent age annotations for sample i, $\hat{x}_i$ is the predicted value, and N is the number of testing samples.

Normal Score (ε): Since the LAP-2016 dataset is labeled by multiple annotators, the performance of an age estimation system might be more accurately measured by taking into account the variance of the annotations for each sample. Therefore the ϙ-score is calculated by fitting a normal distribution with mean μ and standard deviation σ of the annotations for each sample:

$$\varepsilon = 1 - \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right) \tag{16}$$

Thus, the average ϙ-score for a dataset can change between 1 (worst case) and 0 (best case).

The system is implemented in MATLAB. Face detection takes around 2 seconds per image and rotation angle. Feature extraction from VGG-Face with MatConvNet library takes around 1 second per image. For classification and regression, we optimize the kernel parameter γ and the regularization coefficient C with a grid search where both parameters are searched in the exponential set $2^{[-2,-1,\ldots,6]}$. Training the whole system takes 12 minutes and obtaining the estimation takes around 2 seconds per test image.

According to the above conditions, we present the results of our classification and regression systems. In Table 4, we

summarize the classification accuracy and recall for the 8 overlapping age groups we used. The 9th row is the performance of the backup system, and the final row is the performance of the whole system on the validation set of LAP-2016 dataset.

Table Ⅳ shows that the ensemble of local regressors yield smaller MAE for younger age groups. As the age progresses, within-group variance increases with it, making the apparent age estimation task harder. Finally, since younger subjects are usually annotated with less variance, the ϙ-score behaves almost inversely to MAE score, as the ϙ-score is more tolerant for the errors in the older subjects. We display the estimation results on samples from the validation set in Fig. 2, which shows the invariance of CNN features to common difficulties such as blur, pose and occlusions.

TABLE IV. CLASSIFICATION ACCURACY, RECALL AND REGRESSION PERFORMANCE ON VALIDATION SET WITH DIFFERENT AGE GROUPS. N DENOTES THE NUMBER OF SAMPLES

Group	N_{tr}	N_{val}	Acc.	Rec.	ε	MAE
0-15	860	152	0.96	0.78	0.45	2.46
10-25	2366	436	0.84	0.65	0.31	2.90
15-30	3686	662	0.84	0.83	0.31	3.19
20-35	4072	705	0.81	0.86	0.33	3.52
30-40	1764	311	0.81	0.35	0.34	3.82
35-50	1568	288	0.85	0.45	0.34	4.26
45-60	976	184	0.91	0.48	0.28	3.87
55-∞	554	106	0.96	0.57	0.28	4.36
0-∞	8032	1462	-	-	0.40	4.35
Overall	8032	1462	-	-	0.33	3.85

IV. CONCLUSION

In this paper, we propose an apparent age estimation system with the use of SaDE-ELM Algorithm. We show that the performance of local regressors are better than the global regressor for almost all groups. However, we give equal weight to each group a sample is assigned to, whereas weighing the decisions with a membership score can result in more accurate estimation. CNNs are robust to common difficulties in image processing such as pose and illumination differences as well as occlusions. Therefore our system works with a very coarse alignment system, however we believe that obtaining a finer alignment with the help of a landmark detection system will further improve the estimation accuracy.We make use of transfer learning by using the features from a deep network that is trained on a face recognition task and directly employing them in age estimation.

ACKNOWLEDGMENT

This research was supported by science research project of Education Department of Hainan province (Hnky2017ZD-20).

REFERENCES

[1] A. Lanitis, C. Draganova, and C. Christodoulou. Comparing different classifiers for automatic age estimation. IEEE Trans. Syst. Man Cybern. B, 34(1):621–628, 2004.

[2] A. Lanitis, C. J. Taylor, and T. F. Cootes. Toward automatic simulation of aging effects on face images. TPAMI, 24(4):442–455, 2002.

[3] Huang GB, Zhu QY, Siew CK. Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of international joint conference on neural networks (IJCNN2004), vol 2, no 25–29, pp 985–990.

[4] Huang GB, Zhu QY, Siew CK. Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501.

[5] Espana-Boquera S, Zamora-Mart ínez F, Castro-Bleda M J, et al. Efficient BP algorithms for general feedforward neural networks[C]//International Work-Conference on the Interplay Between Natural and Artificial Computation. Springer Berlin Heidelberg, 2007: 327-336.

[6] G. Thatte, U. Mitra, and J. Heidemann, "Parametric methods for anomaly detection in aggregate traffic," IEEE/ACM Transactions on Networking, vol. 19, no. 2, pp. 512–525, April 2011.

[7] M. Qin and K. Hwang, "Frequent episode rules for internet anomaly detection," in Proceedings of the Network Computing and Applications, Third IEEE International Symposium. Washington, DC, USA: IEEE Computer Society, 2004, pp. 161–168.

[8] X. He, C. Papadopoulos, J. Heidemann, U. Mitra, and U. Riaz, "Remote detection of bottleneck links using spectral and statistical methods," Computer Networks, vol. 53, pp. 279–298, February 2009.

[9] W. W. Streilein, R. K. Cunningham, and S. E. Webster, "Improved detec- tion of low-profile probe and denial-of-service attacks," in Proceedings of the 2001 Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection, June 2001.

[10] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, pp. 273–297, 1995.

[11] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," International Journal of Machine Leaning and Cybernetics, vol. 2, no. 2, pp. 107–122, 2011.

[12] G. Tandon, "Weighting versus pruning in rule validation for detecting network and host anomalies," in In Proceedings of the 13th ACM SIGKDD international. ACM Press, 2007.

[13] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," Computers & Security, vol. 25, pp. 439–448, 2002.

[14] Storn R, Price K (2004) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359

[15] Ilonen J, Kamarainen JI, Lampinen J (2003) Differential evolution training algorithm for feedforward neural networks. Neural Process Lett 17:93–105

[16] Subudhi B, Jena D (2008) Differential evolution and levenberg marquardt trained neural network scheme for nonlinear system identification. Neural Process Lett 27:285–296.

[17] P. Viola and M. J. Jones. Robust real-time face detection. IJCV, 57(2):137–154, 2004.

[18] [18] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In ECCV, 2014, pages 720–735.

[19] S. Escalera, M. Torres, B. Martnez, X. Bar, H. J. Escalante, I. Guyon, G. Tzimiropoulos, C. Corneanu, M. Oliu, M. A. Bagheri, and M. Valstar. Chalearn looking at people and faces of the world: Face analysis workshop and challenge 2016. In ChaLearn Looking at People and Faces of the World, CVPRW, 2016

.

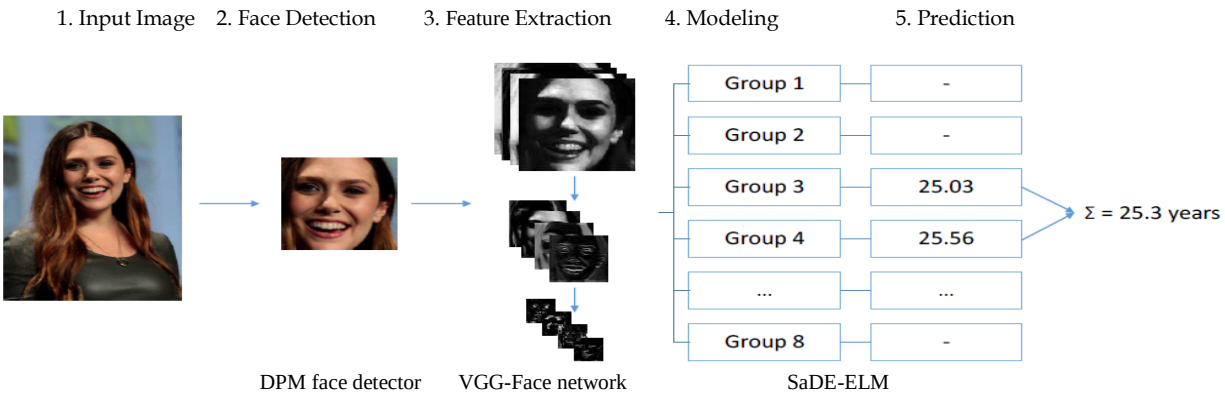

Figure 1. Pipeline of the proposed system

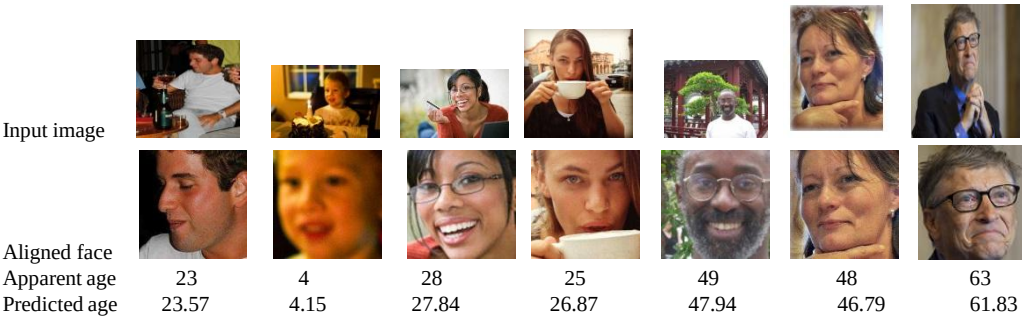

Figure 2. Application in facial age estimation based on sade-elm from the validation set