GABOR FEATURE EXTRACTION DRIVEN FACIAL AGE ESTIMATION USING MULTILAYER PERCEPTRON NEURAL NETWORK

Anjali A. Shejul*
Research Scholar, Department of CSE, JNTUA, Ananthapuramu, Andhra Pradesh, India
anjalishejul123@gmail.com

Kishor S. Kinage
Professor, Department of E&TC, Pimpri Chinchwad College of Engineering, India

B. Eswara Reddy
Professor, Department of CSE, JNTUA College of Engineering, Ananthapuramu, India

Abstract - Facial based human age estimation has become popular now a days due to tremendous increase in real time applications. Age estimation process comes with various challenges such as variation in lighting conditions, poses and facial expression. Performance of age estimation is evaluated with the help of measure 'Mean Absolute Error' (MAE). Main purpose of this work is improving facial based age estimation accuracy by building a Neural Network (NN). Beauty of NN is it learns nonlinear features of input data efficiently. For building NN model input images undergoes preprocessing, feature extraction, training and testing phases. NN model is built by providing training using various training algorithms. For prediction of age estimation neural network have shown superior performance for trainlm algorithm as compared to trainscg and traingdm algorithm. Experimentation are performed on partitioned training images and evaluated on testing images. We achieved low mean absolute error of 4.53 for 70 percent training images dataset.

***Keywords*:** Facial Age Estimation; Feature Extraction; Neural Network

1. Introduction

Predicting age from human face image has real time applications in surveillance, electronic vending machines, security control, forensic art, entertainment, cosmetology and many more. Facial based human age estimation has obtained huge popularity and has become interesting and attractive topic in research domain. As age progresses human face reflects remarkable facial changes [1]. Human face exhibits large amount of information such as gender, expression, age. In this work our main focus is on evaluating performance measure for facial age estimation.

Organization of paper

The remaining part of the paper is organized as follows.

Section 2 describes the related work. Section 3 presents in detail methodology of the work. This section elaborates different steps used in age estimation. Performance measures used by researchers are presented in section 4. Experimentation carried out and corresponding results are mentioned in section 5. Finally conclusion is discussed.

2. Related work

In literature most authors considered age estimation as a multi class classification approach [2][3][4][5]or a regression approach [6][7][8][9][10]. Kuang-Yu Chang et al.[11] introduced a cost sensitive ordinal hyper planes ranking algorithm for facial based human age estimation. Author applied relative order information among the age labels for rank prediction. Shengzheng Wang et al.[12] proposed that asymmetric information can be used to improve the generalizability of the trained model. Author introduced learning using privileged information (LUPI) framework, for this attributes of support vector machine (SVM+) are carefully defined. Author termed specific setting as relative attribute SVM+ (raSVM+).Sun et.al. proposed DeePID structure to extract features [13]. DeepID crops 60 patches from face image and each patch is fed to independent network [14].

3. Methodology

Estimating age and calculating mean absolute error from facial images are primary goal in this paper. Fig.1 depicts steps of age estimation process using Neural Network (NN). In step 1 Image database is fed as input, each image undergoes preprocessing operation to produce noise free image. To extract vital features and to reduce dimensionality of image gabor filtering is performed in step 2, step 3 accepts filtered image having dominant features extracted, builds neural network model by providing training using different training algorithms. Finally testing is performed on NN and MAE is calculated for all algorithms to evaluate performance of NN model. Each step is elaborated in below subsection.

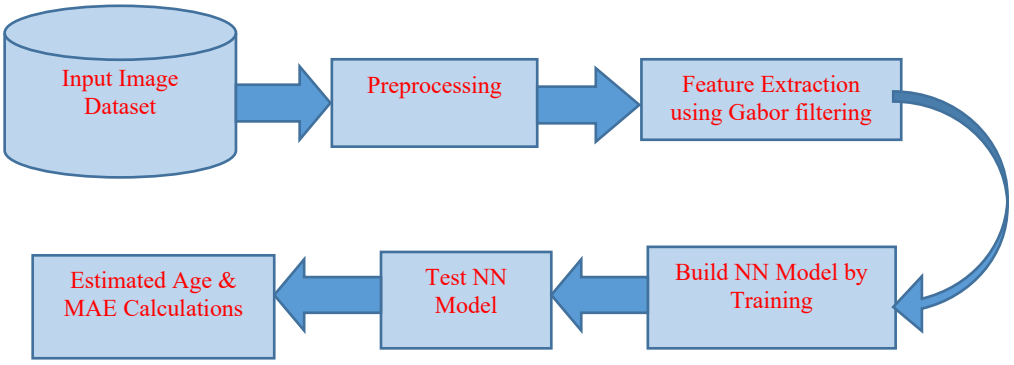

Fig. 1. Steps for age estimation using Neural Network

3.1 Pre-processingInput image dataset includes images from IMDB standard facial image database. To build efficient model all images undergone preprocessing using gaussian filtering to remove noise from images and producing smoothed image.

3.2 Feature Extraction using Gabor Filters

Image input is a multidimensional entity compared to 2 dimensional tabular input or textual input. While building the model to reduce complexity due to multidimensional nature of input images, it need to experience feature selection or feature extraction process. Feature selection process retrieves important subset of features from image which contributes more to form an image. Non important features are discarded. Feature extraction converts image from one form to another form by performing filtering operation and eliminates redundant features to reduce dimensions of image. To perform feature extraction linear and non linear filters are available. In this work we extracted features using Active Appearance model (AAM) and Gabor filter method. AAM is used to extract shape and appearance features of human face. These features are used as global features. Local features are the wrinkle features that can be extracted with Gabor features. As gabor filters are orientation sensitive the directions of the wrinkles on the forehead and the nose side can be extracted. Gabor is powerful linear filters used for texture analysis. It checks for availability of frequency content in specified direction and reject the others. Due to this feature these are also called as band pass filters. Advantage of gabor features extraction method is that we have choice to select scales and orientations, we designed 64 filters using 8 scales and 8 orientations. For example there is an elephant having certain pattern or stripes on its skin with different orientation. If we want to highlight or extract all those patterns we can use bank of 16 gabor filters at an orientation of 14.30 (i.e. if the first filter is at 00, then second will be at 14.30, the will be at 28.60, and so on). Fig. 2 shows filter bank of all 16 filters.

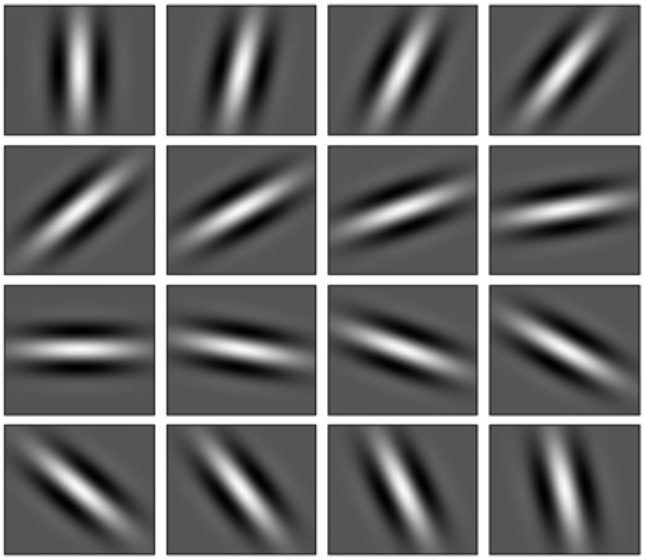

Fig. 2: A bank of 16 Gabor filter oriented at an angle of 14.30 (i.e. if the first filter is at 00, then the second will be at 14.30, the third will be at 28.60, and so on.)

When any image is convolved through each filter in filter bank, edge and texture oriented at an angle at which gabor filter is oriented will be detected. When filter responds to a particular feature then the filter has a distinguishing value at the spatial location of that feature. The filter possesses a real and an imaginary component representing orthogonal directions. In the gabor filter, 5 Parameters controls shape and size of the Gabor filter in deciding features

Lambda (λ):

The wavelength governs the width of the strips of Gabor function. Increasing the wavelength produces thicker stripes and decreasing the wavelength produces thinner stripes.

Theta (Θ):

The theta controls the orientation of the Gabor function. The zero degree theta corresponds to the vertical position of the Gabor function.

Gamma (ɣ):

The aspect ratio or gamma controls the height of the Gabor function.

Sigma (σ):

The bandwidth or sigma controls the overall size of the Gabor envelope.

Psi (Ψ) :

*S*inusoidal function phase offset controls size of gabor filters.

Fig.3 shows sample images from IMDB facial dataset. These images will undergo gabor feature extraction, resultant sample feature extracted sample images are shown in Fig.4.

Fig. 3. Sample images from IMDB facial dataset

Fig. 4. Sample images after performing feature extraction

3.3 Building Neural Network Model

The most well-known neural networks is Multilayer perceptron (MLP). MLP network maps nonlinear relationship from input space x to output space y. Non Linear network can solve non-linearly separable problems.

The inputs x_i, j=1,2,..,J to the neurons will be multiplied by weights w_{ij} ,where w_{ij} is weight of i^{th} input in j^{th} layer and summed up with constant term bias b. Result of this is fed as input to activation function f.

The output of node y_i becomes in Eq. (1)

$$y_i = f(\sum_{j=1}^{J} w_{ji} x_j + b) \qquad (1)$$

Summarizing the procedure of training algorithms for MLP networks:

a. Initially the structure of the network that is total layers and number of neurons in each layer is identified. Then activation functions are decided and the network parameters, weights and biases, are initialized.

b. Secondly, parameters associated with the training algorithm like error goal, maximum number of epochs (iterations), etc., are defined.

c. Neural network model is built by calling train algorithm.

d. Finally, result is first tested by simulating the output of the neural network with the measured input data. This is compared with the expected outputs to find error.

3.3.1 Architecture of Neural Network

Architecture of MLP network adapted in this work is represented in Fig.5, it comprises of an input layer, one hidden layers having 10 neurons followed by an output layer. Sigmoid activation function in hidden layer and linear activation function in output layer is chosen. As sigmoid is nonlinear activation function network will learn linear and nonlinear relationship between input and output vector. Sigmoid transfer function squashes output of network in between 0 and 1. If linear output neurons are used in the last layer the network outputs can take on any value whereas for sigmoid neurons network are limited to a small range.

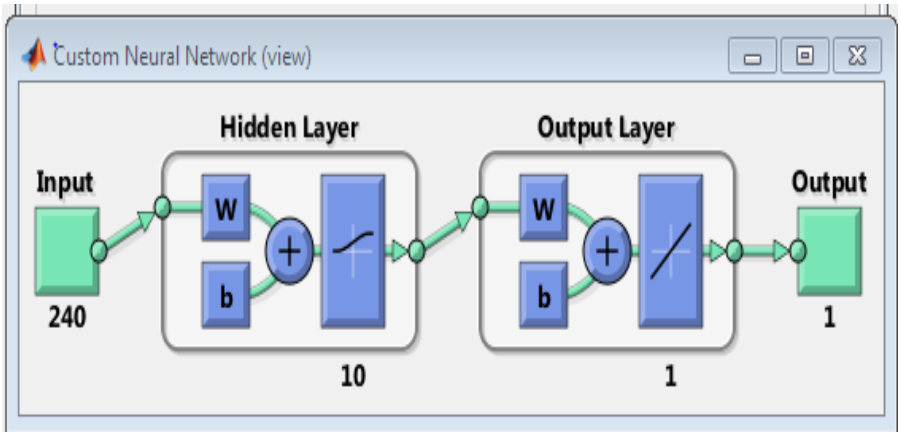

Fig. 5 Architecture of Neural Network

3.3.2 Creating and Training a network (newff)

After deciding neural network architecture MLP network is created using newff method in Matlab. Newff function takes 4 arguments. First input is minimum and maximum values for each element of input vector. Array of size of each layer is specified in second argument. In our case 10, 1 indicates it is 2 layered network having 10 neurons in first hidden layer and one neuron in output layer. Third input is array of transfer functions used in each layer. In this work, sigmoid (logsig) is activation function for hidden layer and linear activation (purelin) function for output layer is used. Name of training function here trainlm is mentioned as last argument. Trainlm algorithm typically takes more memory but less time. The training parameters for trainlm are epochs, show, goal, time, min_grad, max_fail, mu, mu_dec, mu_inc, mu_max, mem_reduc. Few parameters will take default values and hidden from user. Mu value is multiplied by mu_dec whenever the performance function is reduced by a step. It is multiplied by mu_inc whenever a step would increase the performance function. If mu becomes larger than mu_max, the algorithm is stopped. The parameter mem_reduc is used to control the amount of memory used by the algorithm. Epochs, goal, learning rate parameters are set as below. Larger the learning rate bigger is step but if it is too large then algorithm becomes unstable, if it is too small algorithm will take time to converge. Show will decide when to display training status to user. Input dataset is divided in training and testing datasets in 70:30. Newff function will create network object, stores in net variable and initializes weights and biases of the network. Now the network is ready for training. Train function accepts first argument as MLP network object generated by newff. Second argument is input vector (x') and third is output vector (y').Train function perform training on training dataset using back propagation method. During the training network weights are updated along the negative of the gradient. To improve network generalization regularization and early stopping is used. Training can be performed using batch method or incremental method. In batch training weight and bias will be updated after all inputs are fed to network, whereas in incremental method weight and bias will be updated each time an input is fed to network. Trained network is evaluated on testing dataset. Result of same is compared with desired output to calculate error and MAE (mean absolute error)

```
net=newff (minmax(x'),[10 1], {'logsig','purelin'},'trainlm');
net.trainparam.epochs=5;
net.trainparam.goal=1e-25;
net.trainparam.lr=0.01;
net=train(net,x',y');
```

3.3.3 Training algorithm

The network will be trained with Batch Gradient Descent with Momentum (Traingdm), Scaled conjugate gradient back propagation (Trainscg) and Levenberg-Marquardt backpropagation algorithm (Trainlm). All of these algorithms use the gradient of the performance function to determine how to adjust the weights to minimize performance. The gradient is determined using a technique called back propagation, which involves performing computations backwards through the network.

a) Batch Gradient Descent with Momentum (Traingdm)

This is advance version of train gradient descent (traingd) where momentum is included. It provides faster convergence by updating weights and biases in the direction of negative gradient of the performance function. Momentum allows a network to respond to the local gradient and recent trends in the error surface as well. Momentum acts like low-pass filter and allows the network to ignore small features in the error surface. Contribution of adding momentum in a network that it will not allow network to get stuck in a shallow local minimum.

b) Scaled Conjugate Gradient (Trainscg)

Normal conjugate gradient algorithms requires a line search at each iteration where the network response to all training inputs for each search leading to increase in complexity and more time. The scaled conjugate gradient algorithm (SCG), developed by Moller [Moll93], and was designed to avoid the time-consuming line search. Fundamental idea is to combine the model-trust region approach with the conjugate gradient approach.

c) Levenberg-Marquardt back propagation algorithm (Trainlm)

Levenberg-Marquardt is efficient algorithm but requires more memory space. Like the quasi-Newton methods, the Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix.

4. Evaluation Metric

To evaluate performance Mean Absolute Error (MAE) measure is used. MAE identifies the deviation of the output of classifier from actual output to be obtained, and it is expressed as in Eq. (2)

$$MAE = \frac{1}{N}\sqrt{\sum_{i=1}^{N}\left(Y_i - \overline{Y}_i\right)^2} \qquad (2)$$

Where, Y_i and $\overline{Y}_i$ indicates the actual outcome of the classifier, and desired output respectively.

5. Experimental Results

Experimentation is performed in MATLAB using a benchmark face dataset IMDB. Epochs are changed from 2 to 10 to observe changing MAE. Other parameters such as learning rate, mu, minimum gradient and goal are set to 0.4, 0.001, 1e-07 and 1e-25 respectively.

Fig.6 and Table 1 shows analysis of Mean absolute error (MAE) obtained for Trainlm, Trainscg, Traingdm algorithms. MAE 4.53 at epoch 4 is obtained. This shows that Trainlm outperforms than other algorithm.

Table 1. MAE at different epochs for 3 training algorithms

Training Algorithm	Epochs	MAE
Trainlm	2	6.04
	4	4.53
	6	5.12
	8	5.09
	10	5.54
Trainscg	2	9.22
	4	5.46
	6	5.47
	8	5.81
	10	20.46
Traingdm	2	14.46
	4	24.86
	6	5.24
	8	5.53
	10	5.58

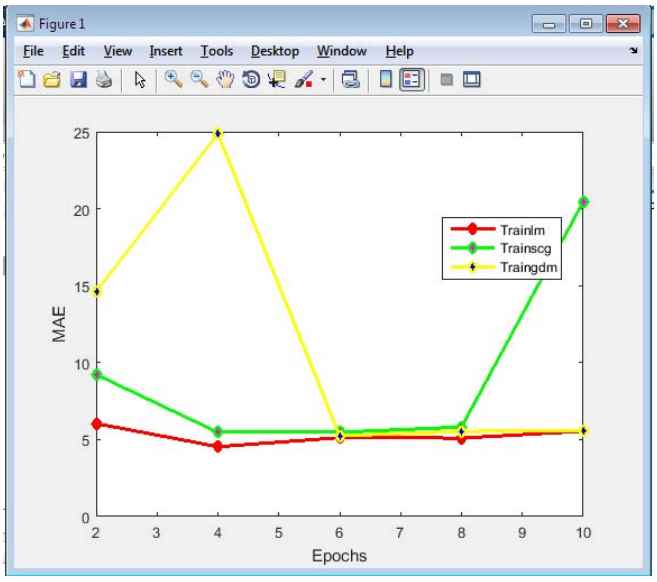

Fig. 6 Comparative analysis using the IMDB face database based on MAE for training algorithm.

MAE at epoch 6 for all three algorithms is shown in Table 2 and Fig.7

Table 2 MAE at epoch 6 for all three algorithms

Training Algorithm	MAE
Trainlm	5.12
Trainscg	5.47
Traingdm	5.24

Fig. 7 MAE at epoch 6 for all three algorithms

As Trainlm is effective algorithm to build neural network model with minimum error for new input data, experimentation is extended to observe and analyze MAE of Trainlm at different learning rates from 0.1 to 0.5 for varying epochs from 2 to 10. Results for this are shown in Fig.8 and Table 3. This shows that minimum MAE 4.46 is obtained at learning rate 0.4 at 8th epoch.

Table 3 MAE obtained for Trainlm at various learning rates

Learning Rate	Epochs	MAE
0.1	2	15.38
	4	4.55
	6	4.76
	8	5.75
	10	5.17
0.2	2	12.95
	4	4.56
	6	4.75
	8	5.36
	10	4.68
0.3	2	5.29
	4	4.70
	6	5.10
	8	5.71
	10	5.39
0.4	2	12.84
	4	4.60
	6	4.75
	8	4.46
	10	5.99
0.5	2	4.90
	4	5.03
	6	5.14
	8	4.54
	10	4.73

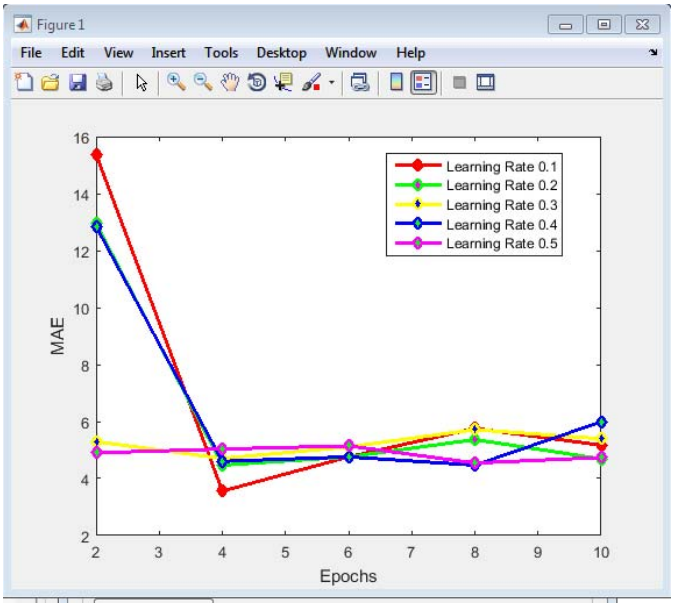

Fig. 8. MAE obtained at different learning rates

Conclusion

In this paper we analyzed performance measure MAE for human age estimation by building neural network model (NN) trained using Trainlm, Trainscg and Traingdm training algorithms. NN outperforms when trained with Trainlm algorithm compared to other algorithm. This shows that Trainlm is more efficient algorithm.

References

[1] X. Geng, Y. Fu, and K. S. Miles, "Automatic Facial Age Estimation," 11th Pacific Rim Int. Conf. Artif. Intell., pp. 1–130, 2010.
[2] A. Lanitis, C. Draganova, and C. Christodoulou, "Comparing Different Classifiers for Automatic Age Estimation," IEEE Trans. Syst. Man, Cybern. Part B Cybern., vol. 34, no. 1, pp. 621–628, 2004.
[3] Z. Yang and H. Ai, "Demographic Classification with Local Binary Patterns," Adv. Biometrics, pp. 464–473.
[4] C.-W. L. and H. Y. M. L. Chung-Chun Wang, Yi-Chueh Su, Chiou-Ting Hsu, "Bayesian age estimation on face images," in 2009 IEEE International Conference on Multimedia and Expo, New York, NY, 2009, pp. 282–285.
[5] X. Geng, Z. Zhou, S. Member, and K. Smith-miles, "Automatic Age Estimation Based on Facial Aging Patterns Automatic Age Estimation Based on Facial Aging Patterns," Pami, vol. 29, no. 200343, pp. 2234–2240, 2007.
[6] Y. F. and T. S. Huang, "Human Age Estimation With Regression on Discriminative Aging Manifold," in IEEE Transactions on Multimedia, vol. 10, no. 4, pp. 578–584.
[7] B. Ni, Z. Song, and S. Yan, "Web image mining towards universal age estimator," Proc. seventeen ACM Int. Conf. Multimed. - MM '09, p. 85, 2009.
[8] G. Guo, G. Mu, Y. Fu, and T. S. Huang, "Human age estimation using bio-inspired features," 2009 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work. CVPR Work. 2009, pp. 112–119, 2009.
[9] G. Guo, Y. Fu, C. R. Dyer, and T. S. Huang, "Image-based human age estimation by manifold learning and locally adjusted robust regression," IEEE Trans. Image Process., vol. 17, no. 7, pp. 1178–1188, 2008.
[10] Y. Z. Y. Zhang and D.-Y. Y. D.-Y. Yeung, "Multi-task warped Gaussian process for personalized age estimation," Comput. Vis. Pattern Recognit. (CVPR), 2010 IEEE Conf., pp. 2622–2629, 2010.
[11] Kuang-Yu Chang and Chu-Song Chen, "A Learning Framework for Age Rank Estimation Based on Face Images With Scattering Transform," IEEE Trans. Image Process., vol. 24, no. 3, pp. 785–798, 2015.
[12] D. T. and J. Y. S. Wang, "Relative Attribute SVM+ Learning for Age Estimation," IEEE Trans. Cybern., vol. 46, n, pp. 827–839.
[13] Y. Sun, X. Wang, and X. Tang, "Deep Learning Face Representation From Predicting 10 000 Classes," Cvpr, pp. 1891–1898, 2014.
[14] K. S. K. and B. E. R. A. A. Shejul, "Comprehensive review on facial based human age estimation," in International conference on Energy, Data Analytics & Soft Computing (ICECDS), 2017, pp. 3211–3216.