

An Effective Method for Gender Classification with Convolutional Neural Networks

Hao Zhang^(✉), Qing Zhu, and Xiaoqi Jia

School of Software Engineering, Beijing University of Technology,
Beijing 100124, China
{zhanghao.py, jiaxiaoqi}@emails.bjut.edu.cn, ccgszq@bjut.edu.cn

Abstract. A gender classification system uses a given image from human face to tell the gender of the given person. An effective gender classification approach is able to improve the performance of many other applications, including image or video retrieval, security monitoring, human-computer interaction and so on. In this paper, an effective method for gender classification task in frontal facial images based on convolutional neural networks (CNNs) is presented. Our experiments have been shown that the method of CNNs for gender classification task is effective and achieves higher classification accuracy than others on FERET and CAS-PEAL-R1 facial datasets. Finally, we built a gender classification demo, where input is the scene image per frame captured by the camera and the output is the original scene image with marked on detected facial areas.

Keywords: Gender classification · Gender recognition · Face detection · Convolutional neural networks · Deep learning

1 Introduction

It is exceedingly critical for us to make the information visualization in communication with others. Not only are we able to identify who his/her is, but also acquire the other information, for instances, gender, age, ethnicity, even mental state when we look at the person's face. It is not difficult task for individual beings, but a big challenge for machines. Gender classification plays a prominent role on many research areas, including images or videos retrieval, human-computer interaction, robotics vision, security monitoring, demographics studies and so on.

A computer system with the capability of gender classification has profound prospect in basic and applied research areas. Although the field of face recognition have been explored by a great deal of researchers, only a few studies on gender classification have been reported. The gender classification process can reduce half of the search space in face recognition, which is favorable for face identification with preprocessing step.

An effective and feasible method for gender classification in facial images has been presented in this paper, which can achieve a better precision for this task.

In addition, other methods for gender classification in face images, including feature extraction and classification, are also presented and compared. Our experiments show that the method in this paper enables to acquire the higher accuracy for gender classification task.

2 Related Works

Xia et al. [1] combined shape and texture features in their experiment. They used Random Forest on the FRGC-2.0 dataset with 10-fold cross-validation and achieved the accuracy of 93.27%. Timotius et al. [2] built a gender classifier with the edge orientation histogram, which is estimated on image pixels, and achieved the accuracy of 84.2% on the ViSiO Lab face database. Shan [3] adopted a compact Adaboost algorithm to learn the better description with local binary pattern (LBP) histogram feature and obtained the accuracy of $94.40 \pm 0.86\%$ on the LFW database [4]. Jabid et al. [5] proposed a novel texture descriptor, local direction pattern (LDP), to detect gender via facial image for gender recognition task, and achieved accuracy of 95.05% on FERET facial dataset. Habid et al. [6], who relied on the movement track of the video frame sequences, presented a gender classification algorithm based on LBP feature with boosting algorithm in the time-spatial domain and achieved high precision on CRIM, Vid-TIMIT and Cohn-Kanade video face databases. Principal component analysis (PCA) via non-linear support vector machine (SVM), representing those images as eigenvector in low-dimensional subspace, was applied in gender classification by Kumari et al. [7] in their experiment, which achieved the accuracy of 92% on the indian face database.

In China, the effective method proposed by Li et al. [8], based on AdaBoost algorithm to combine facial and hair information for gender recognition, achieved accuracy of $95.1 \pm 0.5\%$ on FERET and $95.0 \pm 0.8\%$ on BCMI datasets. Zhang [9] used VG-TSA algorithm with SVM to recognize gender on face image in the videos. Chu et al. [10] randomly cut the detected face image and distinguished gender with support subspace, which made it successful on face gender recognition task with accuracy of 91.13% on the FERET dataset. Chen et al. [11] proposed an iterative learning algorithm which combined active appearance models (AAM) with SVM, and obtained the great improvement on the IMM face database [12].

3 Convolutional Neural Networks

This section describes the feature extraction, the architecture of classification model, and the dropout regularization in our experiments.

3.1 Architecture

Convolutional neural networks (CNNs) are comprised of a series of convolutional and subsampling layers sequentially, and then followed by one or more fully

connected layers in multi-layers perceptron network. CNNs represent the multi-stage Hubel-Wiesel architecture, which extract local features with high resolution and turn them into a lower resolution combined more complex features.

The lower layers consist of alternative layers: convolutional layers and pooling layers. Each convolutional layer presents a two-dimensional discrete convolutional operation on its source image with a given kernel (filter) and applies a nonlinear transfer function. The objective of pooling layers is that reducing the size of the input map thought averaging or summarizing neurons from a small spatial neighborhood. The upper layers, however, are fully connected and correspond to a traditional multi-layers perceptron networks, as illustrated in Fig. 1.

In CNNs, such as LeNet-5 proposed by LeCun et al. [13], shift-invariance is implemented thought subsampling layers. A small non-overlapping receptive fields in these layers receive input neurons of the previous layer. Each neuron calculates the sum of its inputs by a training coefficient, and adds a bias. At last, the intermediate results can be trained through a nonlinear transfer function.

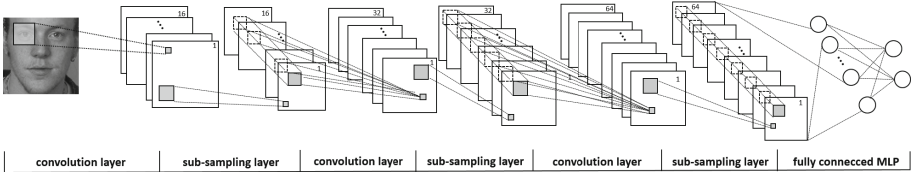


Fig. 1. Convolutional neural networks architecture.

3.2 Convolutional Layers

The purpose of the convolutional layer is that extracting underlying patterns covered within local regions of the input images throughout the dataset, which convolves by a filter over the input image pixels and computes the inner-product with the kernel and pixels at every position of the image.

The input to a convolutional layer is \mathbf{x}_i , where $\mathbf{x}_i \in \mathbb{R}^{w \times h}$ is the $w \times h$ matrix, which is corresponding to the pixels of input map i (image with width w and height h).

The convolutional layer will have k kernels with $m \times n$ size, where m and n is smaller than the width and height of the image respectively. The size of the filters, which are convolved with the image to engender k feature maps with $(w - m + 1) \times (h - n + 1)$ size, leads to the local connected structure.

In convolutional layer, the feature maps in the previous layer need to be convolved with learnable kernels and pass through the activation function to figure out the output feature map. Each output map can be combined with multiple input maps through convolutions. Apparently, we have that

$$\mathbf{a}_j^{(l)} = f \left(\sum_i^k \mathbf{k}_{ij}^{(l)} * \mathbf{a}_i^{(l-1)} + b_j^{(l)} \right) \quad (1)$$

where $*$ is convolutional operator, l represents the layer in the networks, $\mathbf{a}_i^{(l-1)} = \mathbf{x}_i$ when $l = 2$ (the 1st layer in the network is data input layer, and the 2nd layer is convolutional layer), $\mathbf{k}_{ij}^{(l)} \in \mathbb{R}^{m \times n}$ is the kernel of the l -th layer in the networks between the i -th input map and the j -th output map, $f(\cdot)$ is non-linear transfer function to activations, such as hyperbolic tangent function.

An additive bias b_j is given at each output map, but for a specific output map, the input maps will be convolved with different kernels distinctly. In other words, if both the j_1 -th output map and the j_2 -th output map sum over the i -th input map, then the kernels are applied to the i -th input map, which is different from the j_1 -th and j_2 -th output maps.

3.3 Pooling Layers

The objective of pooling [14] layers is to acquire spatial invariance through scaling down the resolution of the feature maps. And then, each map needs to be subsampled typically with mean or max pooling over $p \times q$ adjoining regions, where p and q is the size (width and height respectively) of patches in the pooling layer. In general, we have that:

$$\mathbf{a}_j^{(l)} = g\left(\mathbf{a}_i^{(l-1)}\right), \forall i \in R_j \quad (2)$$

where R_j is the j -th pooling region with $p \times q$ size in the i -th input map.

Two kinds of conventional operators for $g(\cdot)$ can be chosen: average and max operation. The former takes the arithmetic mean of the elements in each pooling region:

$$\mathbf{a}_j^{(l)} = \frac{1}{|R_j|} \sum_{i \in R_j} \mathbf{a}_i^{(l-1)} \quad (3)$$

but the largest element is picked up via the max operator in each pooling region:

$$\mathbf{a}_j^{(l)} = \max_{i \in R_j} \mathbf{a}_i^{(l-1)} \quad (4)$$

Both two kinds of pooling operators have own disadvantages when training deep convolutional networks. When it comes to the average pooling, all elements in a pooling region need to be took fully into account, even though many of them have low magnitude. While max pooling does not suffer from these weaknesses, we are able to find it easily to overfit the training set in practice, making it difficult to generalize well to the test instances. In feedforward propagation, the pooling regions with $p \times q$ size are scaled down as a single value. And then, this single value attains an error calculated by backwards propagation from the previous layer. This error is then just feedforwarded to the place where it comes from. Since it only comes from the one place within the region of $p \times q$ size, the back-propagated errors from max-pooling layers are considerable sparse.

3.4 Regularization

For regularization, we apply dropout [15] regularization on the hidden layer with a constraint on l_2 -norms of the weight vectors. Dropout enables to prevent overfitting thought dropping units out randomly with a probability p in the network. For instance, setting to zero with a probability p of the hidden units during feed-forward and back-propagation in the network. That is, given the hidden layer, also called perceptron layer (after pooling layer), $\mathbf{a}^{(l-1)} = [a_1, \dots, a_h]$, instead of using

$$\mathbf{a}^{(l)} = f(\mathbf{w} \cdot \mathbf{a}^{(l-1)} + b) \quad (5)$$

but for output unit $\mathbf{a}^{(l)}$ in feedforward propagation, dropout represents

$$\mathbf{a}^{(l)} = f(\mathbf{w} \cdot (\mathbf{a}^{(l-1)} \circ r) + b) \quad (6)$$

where \circ is the element-wise multiplication operator, and $r \in \mathbb{R}^h$ is a masking vector of Bernoulli random variables with probability p of being 1. At training time, the gradients are able to be updated only through the unmasked units via back-propagation. At test phase, the weight vectors we have learned enable to be scaled by p such that $\hat{\mathbf{w}} = p\mathbf{w}$, where $\hat{\mathbf{w}}$ is employed without dropout in the test procedure. In addition, we constrain l_2 -norms of the weight vectors by rescaling \mathbf{w} such that $\|\mathbf{w}\|_2 = s$ whenever $\|\mathbf{w}\|_2 > s$ after a gradient descent step.

4 Experiments

This section describes the datasets, data preprocessing steps, detailed hyperparameters, as well as the training procedure in our experiments.

4.1 Datasets and Computers

Our experiments were carried out on the FERET [16] dataset and CAS-PEAL-R1 [17] formal subsets, labeled by ourselves, as described in Table 1. FERET face database dataset consists of 1351 8-bit grayscale images of human from 193 people (108 males and 85 females for each 7 images), heads with views ranging from frontal to left and right profiles. CAS-PEAL-R1 formal subsets made up 1040 Chinese face images from 208 people (119 males and 89 females for each 5 images).

Table 1. Summary statistics for the facial datasets.

Dataset	#images	#people	#males	#females	Each
FERET	1351	193	108	85	7
CAS-PEAL-R1	1040	208	119	89	5

We used up to a workstation machine with Ubuntu 14.04 for our experiments. It has 2 Intel Core CPUs, each for 8 cores (Intel(R) Core(TM) i7-5960X CPU at 3.00 GHz), 32 GB memory and a GPU with GeForce GTX TITAN Z. (6 GB memory). Our experiments below are carried out on GPU with Theano 0.7.0.

4.2 Data Preprocessing

The facial images in the datasets are preprocessed before model training. There are some preprocessed steps as follows.

Gray-Scale. Firstly, the color images, if they have 3 color channels, are converted into grayscale images as follows.

$$\hat{x}_{ij} = 0.299x_{ij}^{(R)} + 0.587x_{ij}^{(G)} + 0.114x_{ij}^{(B)} \quad (7)$$

where $x_{ij}^{(R)}$, $x_{ij}^{(G)}$ and $x_{ij}^{(B)}$ are the pixels of R, G, B channels.

Face Detection. Secondly, histogram equalization is used for adjusting facial image intensities to enhance contrast. And then, we need to detect face area in the image using Harr-likes features with Adaboost method [18] and cut the detected areas. What's more, we are able to resize the images, outputs in the previous step, to 80×80 on FERET dataset and 100×100 on CAS-PEAL-R1 dataset.

Normalization. Ultimately, data normalization, a function map transformed $0 \sim 255$ into $0 \sim 1$, can be used in these datasets. The formula is shown in the following:

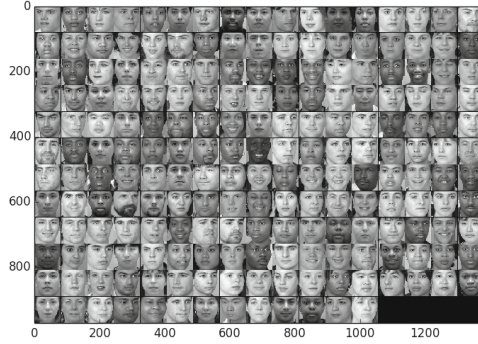
$$\hat{x}_{ij} = \frac{x_{ij} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})} \quad (8)$$

where \hat{x}_{ij} is normalized value, proceeding from original pixel x_{ij} of \mathbf{x} .

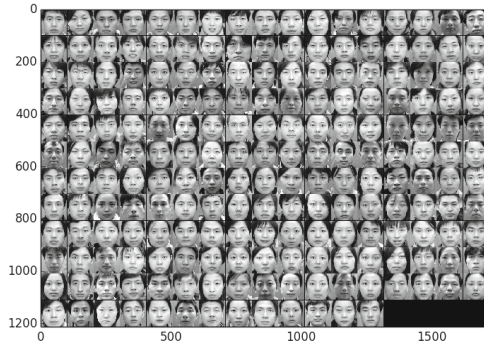
The visualised samples of facial image on FERET and CAS-PEAL-R1 datasets after preprocessing has been shown in Fig. 2.

4.3 Hyperparameters and Training

The CNN architecture with dropout regularization [15] and two kinds of learning algorithms have been implemented by ourselves with Theano. For all datasets, we randomly split all data via 20%–80% as training set and test set separately. Learning algorithms at training procedure has been carried out through Stochastic Gradient Descent (SGD) [19] over mini-batched with learning rate decay update rule and ADADELTA [20] method.



(a) FERET dataset



(b) CAS-PEAL-R1 dataset

Fig. 2. Face image samples visualization after preprocessing on FERET and CAS-PEAL-R1 datasets.

Networks Structure. The exact network architecture on FERET and CAS-PEAL-R1 datasets in our experiment have been listed in Tables 2 and 3. Taking FERET dataset for example, the size of input layer is 80×80 , and followed by convolution with max-pooling layers many times, increasing the number of kernels and reducing the size of patch. Ultimately, flattening the output in the pervious layer, concatenating by fully connected layers with dropout regularization and making classification via softmax layer. The fourth column in the Tables 2 and 3, namely #params, means the number of parameters we need to learn, for instance, 0.4K is equal to $16 \times 5 \times 5$.

For convolution layers, the number of kernels are increased with network going deeper. But for pooling layers, we need to acquire spatial invariance by scaling down the size of the feature maps with max-pooling over given contiguous

Table 2. CNN architecture parameters on FERET.

Type	Kernel size	Output size	#params
input	-	80 × 80	-
convolution	16@5 × 5	16@76 × 76	0.4 K
max-pooling	16@2 × 2	16@38 × 38	0.4 K
convolution	32@5 × 5	32@34 × 34	0.8 K
max-pooling	32@2 × 2	32@17 × 17	0.8 K
convolution	64@3 × 3	64@15 × 15	0.57 K
max-pooling	64@3 × 3	64@5 × 5	0.57 K
flatten	-	1600	-
dropout(0.2)	-	1600	-
fully connected	-	256	409 K
dropout(0.5)	-	256	-
softmax	-	2	0.5 K

regions. Apparently, the loss function of our network is negative log-likelihood in output layer as follows.

$$\mathcal{L}(\theta = \{W^1, b^1, \dots, W^l, b^l\}) = -\frac{1}{|\mathcal{D}|} \sum_{i=0}^{|\mathcal{D}|} \log(P(Y = y^{(i)} | x^{(i)}, \theta)) \quad (9)$$

where θ is the parameters we need to learn in this network. W^i and b^i is represented by the weight matrix and bias in the i -th hidden layer respectively, as well as \mathcal{D} stands for the samples of training dataset, so that $|\mathcal{D}|$ means the number of training data.

Learning Algorithm. In our experiments, two kinds of learning algorithms, SGD [19] and ADADELTA [20], have been used to optimize the loss function and obtained the optimal parameters in the training step.

A stochastic gradient trainer with momentum μ and learning rate α updates parameter θ at step t by blending the current velocity v with the current gradient $\frac{\partial \mathcal{L}}{\partial \theta}$. Besides, we make the learning rate decreased at every step gradually, so the learning decay rate γ is no more than zero and the gradient update rule is followed by Eqs. (10)–(12). (For more training details, please refer to the article [19].)

$$v_{t+1} = \mu v_t - \alpha_t \frac{\partial \mathcal{L}}{\partial \theta} \quad (10)$$

$$\theta_{t+1} = \theta_t + v_{t+1} \quad (11)$$

$$\alpha_{t+1} = \alpha_t \gamma \quad (12)$$

The ADADELTA method uses the same general strategy as all first-order stochastic gradient methods, in the sense that these methods make small parameter

Table 3. CNN architecture parameters on CAS-PEAL-R1.

Type	Kernel size	Output size	#params
input	-	100 × 100	-
convolution	16@5 × 5	16@96 × 96	0.4 K
max-pooling	16@2 × 2	16@48 × 48	0.4 K
convolution	32@5 × 5	32@44 × 44	0.8 K
max-pooling	32@2 × 2	32@22 × 22	0.8 K
convolution	64@5 × 5	64@18 × 18	1.6 K
max-pooling	64@3 × 3	64@6 × 6	1.6 K
flatten	-	2304	-
dropout(0.2)	-	2304	-
fully connected	-	256	589 K
dropout(0.5)	-	256	-
softmax	-	2	0.5 K

adjustments iteratively using local derivative information. The difference with ADADELTA is that as gradients are computed during each parameter update, an exponentially-weighted moving average (EWMA) gradient value, as well as an EWMA of recent parameter steps, are maintained as well. And the update rule is followed by Eqs. (13)–(16). (For more training details, please refer to the article [20].)

$$g_{t+1} = \rho g_t + (1 - \rho) \left(\frac{\partial \mathcal{L}}{\partial \theta} \right)^2 \quad (13)$$

$$v_{t+1} = \frac{\sqrt{x_t + \epsilon}}{\sqrt{g_{t+1} + \epsilon}} \frac{\partial \mathcal{L}}{\partial \theta} \quad (14)$$

$$x_{t+1} = \rho x_t + (1 - \rho) v_{t+1}^2 \quad (15)$$

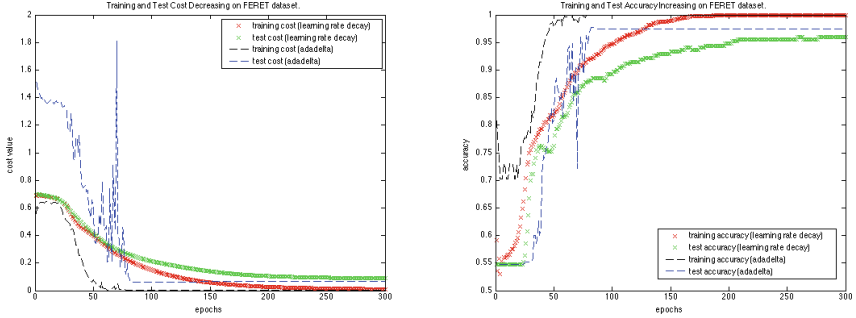
$$\theta_{t+1} = \theta_t - v_{t+1} \quad (16)$$

Parameters Settings. In our experiment, $\tanh(\cdot)$ activation function has been used for convolutional layers and fully connected layers, and setting dropout rates (p) of 0.2 and 0.5 for flatten layers and fully connected layers, initial bias b set 0 and initial weight matrix $W \sim U \left[-\sqrt{\frac{6}{I+O}}, \sqrt{\frac{6}{I+O}} \right)$ for each layer, where I and O is represented by the feature maps of input and output in the hidden layer respectively.

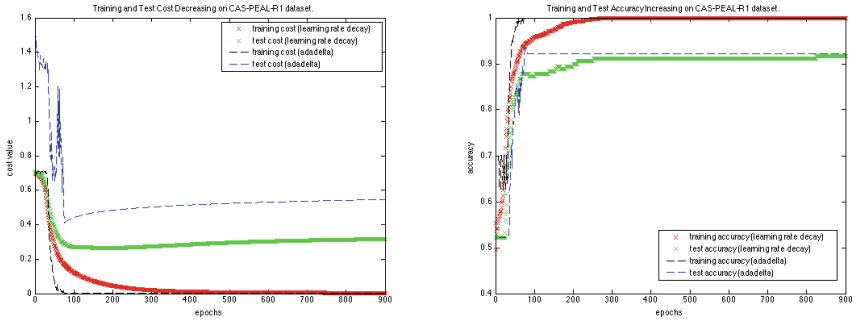
For SGD trainer, initial learning rate α_0 of 0.01, learning decay rate γ of 0.998 and mini-batch size of 30 for stochastic gradient descent learning algorithm. But for ADADELTA trainer, decay constant ρ of 0.95 and constant ϵ of 10^{-6} should be set. These hyperparameters have been selected through a grid search on the FERET and CAS-PEAL-R1 datasets.

4.4 Results and Comparison

As can be seen in Fig. 3, the results have been demonstrated as follows. On the one hand, the cost value with learning rate decay method has smoothly declined and gradually stabilized on all datasets when the epoch increased. In contrast, the cost value has a large number of shaking in the first few epochs and gradually kept stable on all datasets with Adadelta method.



(a) FERET dataset



(b) CAS-PEAL-R1 dataset

Fig. 3. Cost decreasing and accuracy increasing of training set and test set on FERET and CAS-PEAL-R1 datasets.

On the other hand, the accuracy rates increases smoothly with epoch increased. The misclassification rates of test data, as described by Eq. (17), achieves 4.07 % at the 259th epoch on FERET and 8.33 % at the 825th epoch on CAS-PEAL-R1 with SGD learning algorithm. And the error rates obtains 2.23 % at the 82th epoch on FERET and 6.67 % at the 75th epoch on CAS-PEAL-R1 with ADADELTA learning algorithm.

$$\epsilon = \frac{1}{|\mathcal{D}|} \sum_i^{|\mathcal{D}|} I\{y_i \neq \hat{y}_i\} \tag{17}$$

where $I(\cdot)$ is the indicator function, which is that $I\{a \text{ false statement}\} = 0$ and $I\{a \text{ true statement}\} = 1$. And then, y_i is the label of image \mathbf{x}_i , correspondingly, \hat{y}_i is the label of predication, $|\mathcal{D}|$ is the number of test data sets.

In training such networks, the cost value stops decreasing after a few epochs and remains at a level lower than that of learning rate decay method, which achieves a great performance until convergence.

Accordingly, we compared with mainly different methods for gender classification on FERET and CAS-PEAL-R1 datasets, as described in Table 4. Image pixels with logistic regression method have been chosen as our baseline. Compared with these methods which included in Logistic Regression, RBF-SVM, C.H.O, Adaboost and Stacked-Autoencoders, the misclassification rates of our works are 2.23% on FERET and 6.67% on CAS-PEAL-R1, which is lower than previous works. Our works have achieved the great improvements with higher classification accuracy.

In addition, it is exceedingly decreased for the training time-consuming problem with GPU mode compared to CPU mode, as demonstrated on Tables 5 and 6. No matter what the datasets are, training time with GPU mode is 150× faster than other (with CPU mode) approximately. It is a very competitive method, which perform significantly better than their contemporaries.

Table 4. Test error rates on classification with different approaches.

	Methods	FERET	CAS-PEAL-R1
Baseline	Logistic regression	14.00 %	11.50 %
Ours	CNN (SGD)	4.07 %	8.33 %
	CNN (ADADELTA)	2.23 %	6.67 %
Others	LDP + SVM [5]	4.95%	-
	(C.H.O)[8]	4.90 ± 0.5%	-
	SubSVM [10]	8.87%	-
	Pixels + RBF-SVM [21]	-	14.04 %
	LBP + RBF-SVM [21]	-	20.55 %
	Adaboost [21]	-	15.90 %
	Autoencoder + RBF-SVM [22]	14.66 %	16.37 %
	Stacked-autoencoders [22]	9.98 %	11.71 %

5 Application to Gender Recognition from Camera

Finally, we carried out the CNNs model, which were trained and tuned on hyper-parameters by ourselves, to build a application for gender classification with facial images.

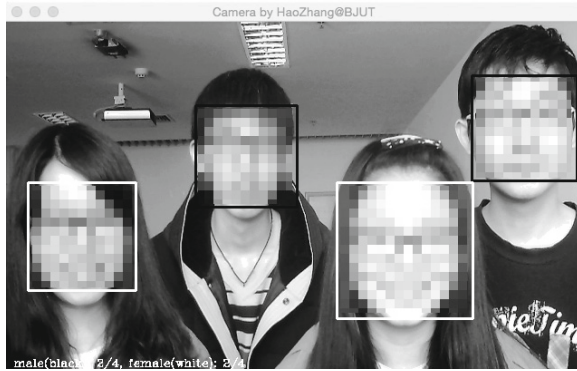
Image acquisition on natural scene is captured by camera. And then, the whole image is preprocessed to acquire the area about human face, as described

Table 5. Average execution time per epoch on FERET dataset.

Methods	Data	Theano CPU	Theano GPU
CNN (SGD)	Training	65.690 s	0.351 s
	Test	3.979 s	0.023 s
CNN (ADADELTA)	Training	66.174 s	0.358 s
	Test	2.923 s	0.022 s

Table 6. Average execution time per epoch on CAS-PEAL-R1 dataset.

Methods	Data	Theano CPU	Theano GPU
CNN (SDG)	Training	59.067 s	0.408 s
	Test	3.195 s	0.022 s
CNN (ADADELTA)	Training	59.602 s	0.401 s
	Test	3.106 s	0.021 s

**Fig. 4.** Application of gender classification based on CNNs model.

in Sect. 4.2. Next step, the facial area of image as 2d signal inputs is put into the CNNs model, which has been trained parameters previously, included the weight matrix and bias for each layer, the kernels for each convolutional layer, and the patch size for pooling layer, the dropout rates for perceptron layers and so on. At last, we get the outputs from the model, which enables to distinguish male from female on facial images, and label on the original image. The rectangle with black lines on face represents male, and the white one stands for female, as illustrated in Fig. 4.

6 Conclusion

An effective and feasible method for gender classification in facial images has been presented in this paper. The underlying gender descriptors on face would

need to be general, invariant to pose, illumination, expression, and image quality. Our works demonstrate that CNNs model can learn from many samples to distinguish gender effectively on facial images. Adadelta method, to tune the learning rate automatically, is able to make convergence rapidly and acquire the much lower misclassification rates in our works.

The ability, to present a remarkable improvement in gender classification, attests to the potential of such coupling to become significant in other vision domains as well. Finally, this method has been applied in the gender recognition system in order to build a application to predict the gender with facial images captured by camera.

In the next step, we still need to improve the accuracy for classification performance and focus on high-precision recognition under the influence of different positions and orientations.

Acknowledgments. This research is partially supported by Beijing Natural Science Foundation (Grant 4152008). Hao is supported by the Foundation of Science and Technology of Beijing University of Technology (Grant ykj-2013-9341). We sincerely thank the anonymous reviewers for their thorough reviewing and valuable suggestions. In addition, the authors would also give warm thanks to Yutong Yu for the data labelling works, and Lei Wang and Zhiqiang Wang for their comments and discussions.

References

1. Xia, B., Ben Amor, B., Huang, D., Daoudi, M., Wang, Y., Drira, H.: Enhancing gender classification by combining 3D and 2D face modalities. In: 2013 Proceedings of the 21st European Signal Processing Conference (EUSIPCO), pp. 1–5, September 2013
2. Timotius, I.K., Setyawan, I.: Using edge orientation histograms in face-based gender classification. In: 2014 International Conference on Information Technology Systems and Innovation (ICITSI), pp. 93–98, November 2014
3. Shan, C.: Learning local binary patterns for gender classification on real-world face images. *Pattern Recogn. Lett.* **33**(4), 431–437 (2012). *Intelligent Multimedia Interactivity*
4. Huang, G.B., Mattar, M., Lee, H., Learned-Miller, E.: Learning to align from scratch. In: NIPS (2012)
5. Jabid, T., Kabir, M.H., Chae, O.: Gender classification using local directional pattern (LDP). In: 2010 20th International Conference on Pattern Recognition (ICPR), pp. 2162–2165, August 2010
6. Hadid, A., Pietikainen, M.: Combining appearance and motion for face and gender recognition from videos. *Pattern Recogn.* **42**(11), 2818–2827 (2009)
7. Kumari, S., Sa, P.K., Majhi, B.: Gender classification by principal component analysis and support vector machine. In: Proceedings of the 2011 International Conference on Communication, Computing and Security, ICCCS 2011, pp. 339–342. ACM, New York (2011)
8. Li, B., Lian, X.-C., Lu, B.-L.: Gender classification by combining clothing, hair and facial component classifiers. *Neurocomputing* **76**(1), 18–27 (2012). Seventh International Symposium on Neural Networks (ISNN 2010) *Advances in Web Intelligence*

9. Zhang, M.: Design and implementation of video-based face gender recognition system using manifold learning. Master's thesis, University of Electronic Science and Technology (2013)
10. Chu, W.-S., Huang, C.-R., Chen, C.-S.: Gender classification from unaligned facial images using support subspaces. *Inf. Sci.* **221**, 98–109 (2013)
11. Chen, H., Wei, W.: Support vectort aam based iterative learning algorithm for gender classification. *J. Zhejiang Univ. (Eng. Sci.)* **39**(12), 1989–2011 (2005)
12. Nordstrøm, M.M., Larsen, M., Sierakowski, J., Stegmann, M.B.: The IMM face database - an annotated dataset of 240 face images. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, May 2004
13. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2324 (1998)
14. Zeiler, M.D., Fergus, R.: Stochastic pooling for regularization of deep convolutional neural networks. *CoRR*, abs/1301.3557 (2013)
15. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014)
16. The defense advanced research team: the facial recognition technology (feret) database. Technical report, United States Department of Defense (1996)
17. Gao, W., Cao, B., Shan, S., Chen, X., Zhou, D., Zhang, X., Zhao, D.: The CAS-PEAL large-scale chinese face database and baseline evaluations. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **38**, 149–161 (2008)
18. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: *Proceedings of the 2002 International Conference on Image Processing*, vol. 1, pp. I-900–I-903 (2002)
19. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. In: *Neurocomputing: Foundations of Research*, pp. 696–699. MIT Press, Cambridge (1988)
20. Zeiler, M.D.: ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701 (2012)
21. Lu, L.: A study of gender classification and age estimation based on face image. Ph.D. thesis, Shanghai Jiao Tong University (2010)
22. Zhang, H., Zhu, Q.: Gender classification in face images based on stacked-autoencoders method. In: *2014 7th International Congress on Image and Signal Processing (CISP)*, pp. 486–491, October 2014