An optimization of facial feature point detection program by using several types of convolutional neural network

Shyota Shindo[1], Takaaki Goto[2], Tadaaki Kirishima[3], Kensei Tsuchida[4]
[1,3,4]Toyo University, 2100 Kujirai, Kawagoe, Saitama, Japan
[2]Ryutsu Keizai University, 3-2-1 Shin-Matsudo, Matsudo, Chiba, Japan

Article Info

Article history:

Received Jan 17, 2019
Revised Apr 7, 2019
Accepted May 10, 2019

Keywords:

Facial feature point detection
Neural network
Convolutional neural network

ABSTRACT

Detection of facial feature points is an important technique used for biometric authentication and facial expression estimation. A facial feature point is a local point indicating both ends of the eye, holes of the nose, and end points of the mouth in the face image. Many researches on face feature point detection have been done so far, but the accuracy of facial organ point detection is improving by the approach using Convolutional Neural Network (CNN). However, CNN not only takes time to learn but also the neural network becomes a complicated model, so it is necessary to improve learning time and detection accuracy. In this research, the improvement of the detection accuracy of the learning speed is improved by increasing the convolution layer.

Corresponding Author:

Takaaki Goto,
Ryutsu Keizai University,
3-2-1 Shin-Matsudo, Matsudo, Chiba, Japan.
Email: tg@gotolab.net

1. INTRODUCTION

A facial feature point is a local point indicating a place such as an eye end or a mouth end of a facial image. The detection of facial feature points is applied to important technologies such as facial expression estimation and biometric authentication using facial images. Many detection methods have been proposed so far, but with the advent of Convolutional Neural Network (CNN) in recent years, many researches on detection methods using CNN have been conducted, and detection with higher accuracy is getting expected [1].

However, CNN learning takes time. If the layer of CNN becomes deep and the number of training data is large, the learning time becomes huge. As a method for speeding up learning, there are methods using GPU with good performance, and methods for devising hardware such as adding main memory. In addition, the methods for devising software are [2, 3, 4]. Among them, there are a few methods [5] to devise pre-processing for input data.

In this paper, we aim at improving preprocessing of input data and speed up learning of facial feature point detection program using CNN. CNN was implemented in Python with reference to the program of Yamashita et al. [6]. We propose a method to reduce the number of layers of CNN by applying Laplacian filter to preprocessing and reducing image features.

2. RELATED WORKS

Facial feature point detection can be obtained by various methods such as CNN and image processing. As a conventional method, Cootes et al's Active Appearance Model (AAM) is available [7]. In this method, the average Shape is obtained by using the coordinate points of the facial images of all the learning data, and the average face is obtained by using the pixel values. Principal component analysis is performed using the coordinate points of this Shape and the pixel values in the Shape, and the change amount is obtained. Appearance can show the features of the front face and Shape can express the orientation and shape of the face. By combining two, it becomes possible to create a face image that can respond to changes in face orientation and shape. That is, in order to obtain the facial feature point of the input face image, Shape and Appearance are updated by the gradient descent method. However, this method can not deal with unknown image data, resulting in low accuracy. As a method of image processing, Vukadinovic et al. independently detect each feature point using a Gabor filter (a filter that extracts the direction of the line in the image) [8].

Recently used method is CNN. Since winning in the object recognition category of ILSVRC in 2012, CNN has received great attention. As a facial feature point detection method using CNN, there is a method by Kimura et al [9]. In this method, facial organ points are detected by learning input values as 100×100 grayscale images and learning teaching data as coordinate values of facial organ points of the images. This makes it possible to cope with unknown image data. There is also a method of creating an optimum mini batch in CNN mini batch learning. There is also a method of creating an optimum mini batch in CNN mini batch learning [10]. Minagawa et al. do not use CNN but detect it using DNN [11]. In this proposed method, points are marked as a learning sample in an image among existing correct feature point and a certain range of the feature point, and a transfer vector representing the relative position from the feature point to each learning sample is used. However, with this approach, it is necessary to use a separate DNN for each organ and the accuracy is reduced.

Conventional methods were those that independently detect each feature by image processing, or one using CNN. Although there is a method using DNN, complicated processing is involved. In this research, we aim to propose a new method which is more accurate and faster than the conventional method. In addition, we also compare execution time and detection accuracy with detection by effective CNN in facial feature point detection.

In [4], authors search for an object from the image, clip it out, and make it an input value to CNN. And CNN judges whether it is a face or not (R - CNN). This paper is a research on accelerating R-CNN. Our method is seeking facial organ points for image data which has already cut out the face. [1] proposes a face authentication system that Facebook made. It is different from our research that the face orientation is detected and affine transformation is performed. In [6], the proposed method reduces the coordinate values of facial organ points. [12] shows the result that it becomes high performance when learning facial organ point regression and classification of features such as wearing glasses simultaneously (TCDCN).

As in the model of [13] , there are researches that adapted Laplacian filter to pretreatment. However, this is a technique for reducing the variability of the input pattern by the face detection program and does not mention the learning speed.

[14] describes facial feature point detection using Gabor filter. The method in this research differs from this research. In paper [15], face recognition is used in the alarm system. A method of extracting a feature quantity by a histogram is used for a face recognition method. In paper [16], the Enhanced Local binary pattern (EnLBP) is performed to compress the image and stored in the database. Authors of this paper have proposed a method to recognize faces by comparing saved images with images EnLBPed of input images.

3. RESEARCH METHOD

Python is used for the programming language of machine learning conducted in this research. Many machinery learning support libraries are provided in Python, but we do not use these libraries because this study does not make the comparison of execution times ambiguous.

Table 1 shows the PC environment in which machine learning was performed. In addition, Python has a library for CUDA that allows GPU to calculate, but in this research we have not done calculations with GPU at all.

Table 1. Environment

Item	Spec
CPU	Intel(R)Celeron(R)2957U, 1.40GHz, Multi-Core
Memory	4.00GB

First of all, CNN of the structure of Figure 1 which is the standard in this research was implemented

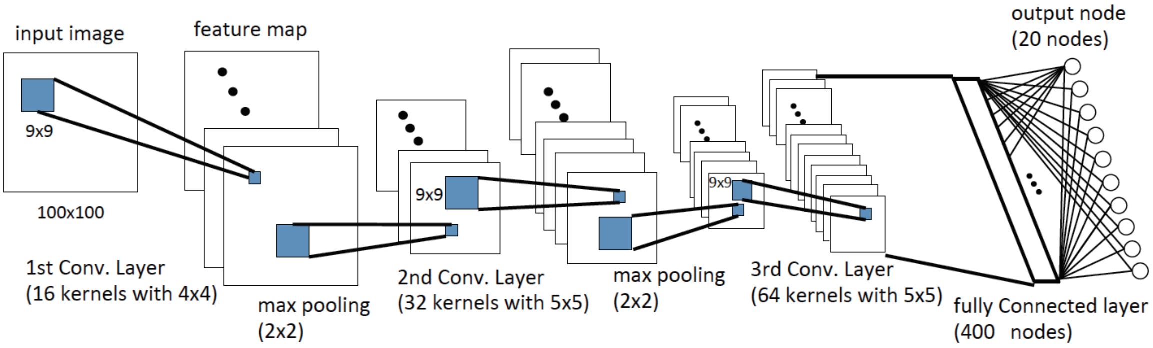

Figure 1. The structure of CNN [10]

3.1. Implementation of Facial Feature Point Detection by CNN

As shown in Fig. 1, the CNN hierarchically follows the convolution layer and the pooling layer after the input image, and then passes through the fully connected layer. For the structure and activation function of CNN in this study, we refer to [10], and the convolution layer and pooling layer shall be as shown in Figure 1. Because CNN is a supervised learning, CNN uses the squared error between the output value and the teacher data as a loss function. Furthermore, it transmits error to the input layer by error back propagation method and updates by gradient descent method. We use the Labeled Faces in the Wild (LFW) data set. The LFW data set cuts out an image with the detection range from the forehead of the face image to the submaxillary as the detection range and annotates the image after normalizing the cut out image to 100 × 100 in that range. The coordinate value and the clipping range at that time are publicly available. Annotation contains a total of 10 points, 4 eyes at both eyes and at the inner corner, 2 at the bottom of the nose, 4 points at both ends of the lip and above and below. Since the learning amount is not enough with only the LFW data set, 1500 sheets of image data are subjected to data aggregation to increase the number of images for learning to 20000 sheets. Data augmentation performed is noise addition, translational movement of ±5 pixels up and down, left and right, averaging by mean filter, sharpening by sharpening filter.

As learning methods, batch learning, stochastic gradient descent method, mini batch can be used, but in this research mini batch is adopted. Creating a mini-batch is carried out by randomly selected from the image obtained by increasing perform data augmentation. In addition, we will use the same epoch every mini batch in learning, not shuffle in order. For mini batch, we make 20 sheets per batch and divide the error accumulated during the gradient descent method by the number of mini batches. The learning update rate is $le-4(0.0001)$, the number of epochs is 20, and updating is done 20,000 times.

Next, the structure of CNN will be explained. First, as an input value, an image is clipped to the range shown in the LFW data set, and an image normalized to a size of 100 × 100 and made into grayscale is used. Next, the convolution layer and the pooling layer are alternately arranged and each has three layers. The filter size of each convolution layer is 9 × 9, and the number of movements in the convolution operation is 1 pixel. The activation function uses Maxout with two adjacent feature maps. The number of filters is 16, 8 × 32, and 16 × 64. Each pooling layer performs max-pooling by filter size 2 × 2. Finally, the total coupling layer and the output layer are composed of one layer, and the input value becomes one dimension of the feature map through the pooling layer. In this paper, the number of input values is 1152, and the number of output values is 20. The number of outputs is the binary value of the coordinate values x and y on which the annotation is performed, and it is twice the number of the feature points. The output value is between 0 and 1, and the activation function uses linear combination. Since the output is between 0 and 1, the teacher data is divided by 1000, and when it is detected, it is obtained by multiplying the output value by 1000. Also, to prevent over learning, Dropout is provided in all the bonding layers. The probability of Dropout is 50%, and it is independent for each image in the subset. The structure of CNN is shown in Table 2.

Table 2. Details of CNN

	Size of images	Filter, number of weight	Activation function
Input Layer	100 × 100		
Convolutional layer 1	92 × 92	16	Maxout
Pooling Layer 1	46 × 46		
Zero padding	48 × 48		
Convolutional layer 2	40 × 40	8 × 32	Maxout
Pooling Layer 2	20 × 20		
Convolutional layer 3	12 × 12	16 × 64	Maxout
Pooling Layer 3	6 × 6		
Full connected Layer		1152	
Output Layer			Linear Combination

As a result, the execution time was about 61 hours. However, with only 20 epochs, the error converged only to about 0.012. Detection accuracy using test data was 96%. Since the learning was carried out at 20 epochs this time, the accuracy deteriorated considerably but the convergence of the error was quite slow but it gradually became smaller, so if update has been done 300 thousands, it is expected that the error can converge to almost 0. In the neural network devised in this research, the first goal is to make the convergence of errors faster and better than this result. The second goal is to implement things that leave 20 more epochs and continue to converge even less error.

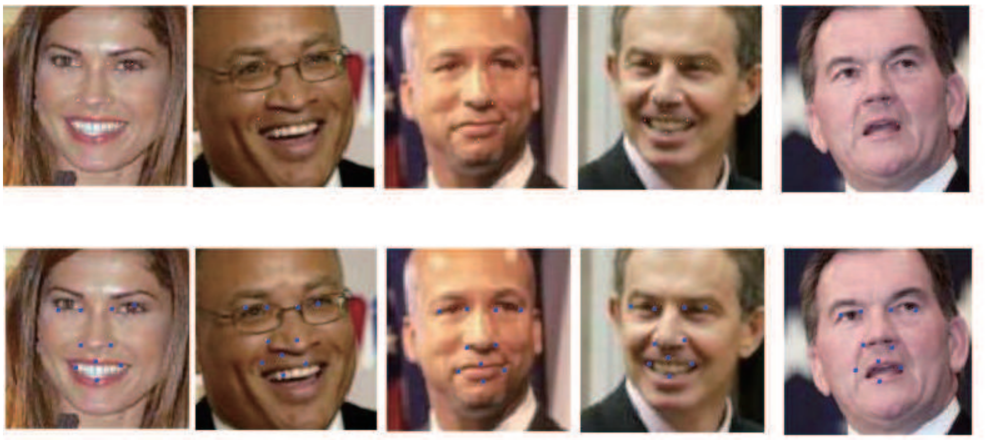

Figure 2. A result of facial feature detection by CNN (Photographic images are obtained from [17])

3.2. Outline of proposed neural network

We propose a neural network which learns images with DNN or CNN after applying the Laplacian filter. The Laplacian filter is a kind of filter processing used for image processing, and it is possible to extract only the portion where the difference in luminance value is drastic from the image. By utilizing this property, information amount is reduced from the input image, and the structure of the neural network is simplified.

There are two reasons why we decided to use the Laplacian filter: (1)It is not necessary to consider the difference in skin color due to race by using Laplacian filter, and (2) This is because it is possible to discriminate facial feature points such as the edge of the eyes and the edge of the mouth by only the outline from human eyes therefore we assumed that machine learning can also discriminate same as human.

In order to use the image processed by Laplacian filter for learning of neural network, all input images were processed with Laplacian filter. The coefficient of the filter is as follows:

$$Filter = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Instead of processing the input image as resized to 100 × 100 as it is, the input image is changed to a size of 102 102 with zero padding, and filtering is performed. Figure 3 shows input images after applying Laplacian filter.

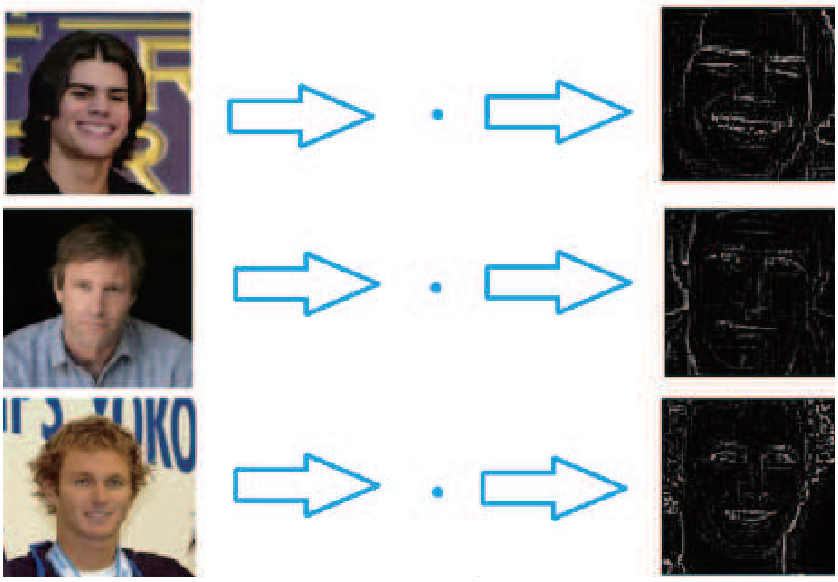

Figure 3. Input images after applying Laplacian filter (Photographic images are obtained from [17])

Learning was done with neural networks using these images. As input values, a one-dimensional input image reduced by two pooling layers is used as preprocessing. The structure of the neural network is shown in Figure 4.

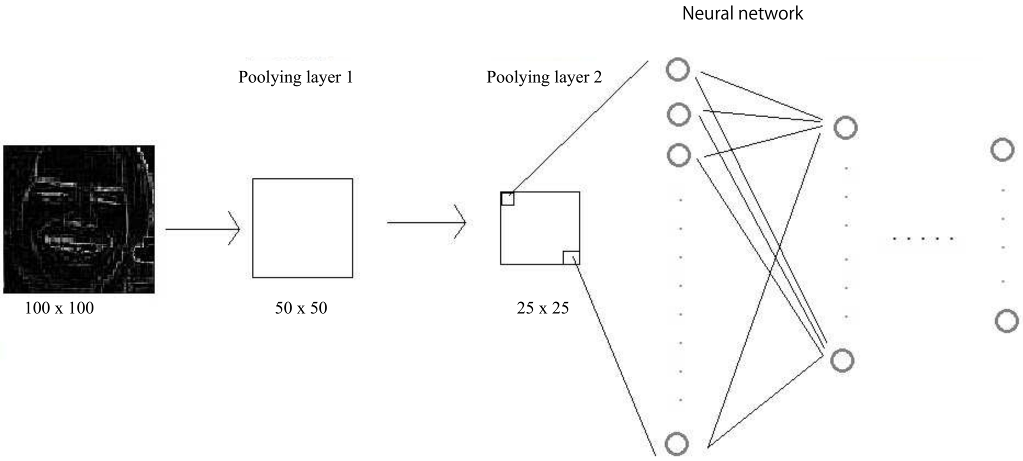

Figure 4. Framework of beta type neural network

4. RESULTS AND DISCUSSION

We implemented and evaluated the type neural network with various configurations.

4.1. Beta type 5-layer DNN

We implemented and verified 3 - layer DNN, 4 - layer DNN and 5 - layer DNN. Table 3 shows the beta type 5-layer DNN that finally converged most. Learning was conducted with three layers of intermediate layers in order to further improve discrimination power. Also, although the activation function used Maxout and Sigmoid up to the last time, Maxout has a better result of convergence of error even if layers are added, so fix it to Maxout. The composition of each layer is summarized in Table 3. The initial value is 1e-3 to -1e-3, and the learning rate is 1e-2.

Table 3. Framework of beta type 5-layer DNN

	Size of images, number of units	Activation function	Probability of Dropout
Pooling Layer 1	100×100		
Pooling Layer 2	50×50		
Input Layer	625		
Intermediate layer 1	600	Maxout	50%
Intermediate layer 2	500	Maxout	25%
Intermediate layer 3	400	Maxout	10%
Output Layer	20	Linear Combination	

The execution time was about 2 hours when the number of units was the minimum and about 4 hours at the maximum. The error did not decrease from nearly 0.01. Sometimes the error was as large as 0.02. As a result of increasing the number of interlayers, the error convergence has improved considerably. However, since the error increases with the input image, we found that the discrimination power is still weak. Therefore, we added a convolution layer and thought about learning with CNN.

Then we had another experiment to detect with CNN. As the input value, use the input image resized to 50×50. Then, the convolution layer and the pooling layer are repeated several times to pass to the entire binding layer. The structure of CNN is shown in Figure 5.

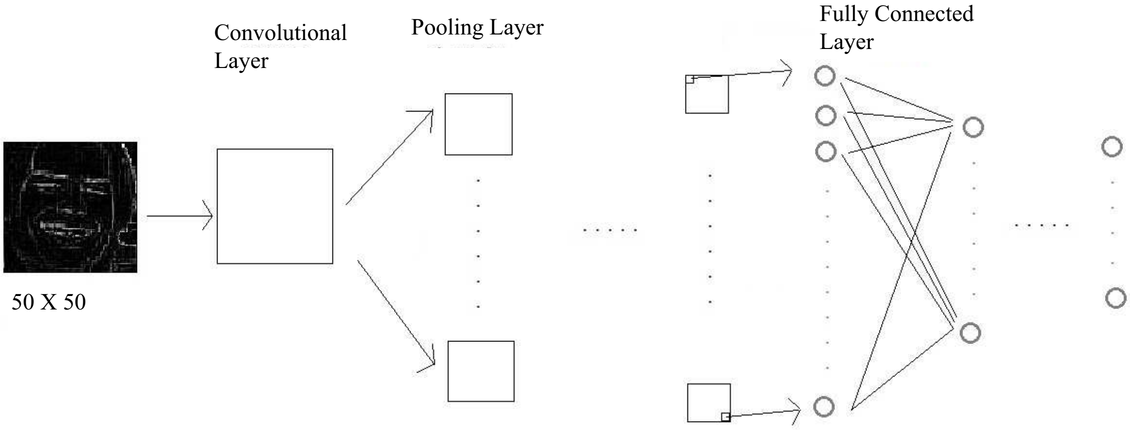

Figure 5. The structure of beta type CNN

4.2. Beta type CNN with 2-layer convolution layer

In this time, the input image is passed directly to the convolution layer without being reduced by the pooling layer. The output of the convolution layer is gradually increasing. The total coupling layer uses the previous DNN. The size of the filter of the convolution layer is 11×11, the initial value is $1e-3$ to $-1e-3$, and the learning rate is $5e-3$. The composition of each layer is summarized in Table 4.

Table 4. Framework of beta type CNN with 2-layer convolution layer

	Number of channel × hight × width	Activation function	Probability of Dropout
Input Layer	50×50		
Convolution Layer 1	$50 \times 40 \times 40$	Maxout	
Pooling Layer 1	$25 \times 20 \times 20$		
Convolution Layer 2	$40 \times 10 \times 10$	Maxout	
Pooling Layer 2	$20 \times 5 \times 5$		
Input of All connected layer	500		50%
Intermediate layer 1	300	Maxout	25%
Intermediate layer 2	200	Maxout	10%
Output Layer	20	Linear Combination	

The execution time was about 6 hours when the output number of the convolution layer and the number of units of the middle layer was the minimum, the maximum case was about 16 hours, and the error converged to 0.02.

The more convergence layers are added, the better the error convergence. From this result, it was found that the convergence of error was improved considerably when CNN added a layer rather than DNN. However, it also turned out that the execution time also significantly increased. From this, it is expected that β-type CNN can be considerably increased in speed.

If the convolution layer has two layers, we increase the output number of the convolution layer more, it will be a faster and more accurate classifier than the CNN implemented. However, in order to raise the accuracy without increasing the execution time any more, the learning was performed by further reducing the feature amount of the input image. Reduction is to set the small pixel value of the input image to 0.

Even if small pixel values are reduced, since edges of the eyes can be recognized by human eyes, learning is carried out with three patterns of 10 pixels or less, 20 pixels or less, 50 pixels or less of the input image set to 0. The execution result converges to about 0.02 if less than 10 pixels is made 0, and when the 20 pixels or less is made 0, the error converged to about 0.013. And if we set 0 pixel or less to 0, the error converged to about 0.03. From this result, very good convergence was obtained by setting 0 to 20 pixels or less.

In the method, the error has already converged to about 0.013 at the time of reaching 10 epochs, and the error did not become smaller thereafter.

Table 5. Result of learning time and convergence of error for each NN

	The maximum learning time	The convergence of error
Beta type NN	10m	Failed
Beta type 3-layer DNN	1h30m	Failed
Beta type 4-layer DNN	3h	Failed
Beta type 5-layer DNN	4h	Almost success
Beta type CNN	22h	Success

As shown in Table 5, in Beta type NN, learning failed with all patterns. In Beta type DNN, things of 3 layers and 4 layers failed to learn. Five layers of learning results were quite good, but it was not enough. And Beta type CNN succeeded in learning.

For the CNN not pretreated with the Laplacian filter, the learning time of the β type CNN pretreated was about 2.8 times faster and the detection accuracy was 1% lower as shown in Table 6.

Table 6. The accuracy comparison

	Base line CNN	β type CNN
Learning time	61 hours	22 hours
Accuracy	96%	95%

5. CONCLUSION

In this research, facial feature points were detected by CNN using the programming language Python. Then, we attempted to create a neural network with a fast learning time and a higher precision than this CNN. In this research, input images are processed by Laplacian filter, and then learning is done with neural network. The point that the image processing processing is performed first is devised, and the learning time has drastically decreased compared with the neural network (the image processing by the Laplacian filter is not done).

For future work, it is conceivable to add improvements such as using a Laplacian filter or increasing the convolution layer of CNN.

REFERENCES

[1] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 1701–1708.

[2] D. Triantafyllidou and A. Tefas, “A fast deep convolutional neural network for face detection in big visual data,” in *Advances in Big Data*, P. Angelov, Y. Manolopoulos, L. Iliadis, A. Roy, and M. Vellasco, Eds. Springer International Publishing, 2017, pp. 61–70.

[3] D. Triantafyllidou, P. Nousi, and A. Tefas, “Fast deep convolutional face detection in the wild exploiting hard sample mining,” *Big Data Research*, vol. 11, pp. 65 – 76, 2018, selected papers from the 2nd INNS Conference on Big Data: Big Data and Neural Networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214579617300096

[4] S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[5] Q. Gao, P. Forster, K. R. Mobus, and G. S. Moschytz, “Fingerprint recognition using cnns: fingerprint preprocessing,” in *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196)*, vol. 3, May 2001, pp. 433–436 vol. 2.

[6] T. Yamashita, T. Watasue, Y. Yamauchi, and H. Fujiyoshi, “Facial point detection using convolutional neural network transferred from a heterogeneous task,” in *2015 IEEE International Conference on Image Processing (ICIP)*, Sep. 2015, pp. 2725–2729.

[7] T. F. Cootes, G. J. Edwards, and C. J. Taylor, “Active appearance models,” in *Computer Vision — ECCV’98*, H. Burkhardt and B. Neumann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 484–498.

[8] D. Vukadinovic and M. Pantic, “Fully automatic facial feature point detection using gabor feature based boosted classifiers,” in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, Oct 2005, pp. 1692–1698 Vol. 2.

[9] M. Kimura, H. Fukui, T. Yamashita, Y. Yamauchi, and H. Fujiyoshi, “Facial point detection based on deep convolutional neural network with optimal minibatch,” in *TECHNICAL REPORT OF IEICE. CNR, Technical Committee on Cloud Network Robotics (CNR)*, vol. 114, no. 455. The Institute of Electronics, Information and Communication Engineers, Feb. 2015, pp. 87–88, (in Japanese). [Online]. Available: https://ci.nii.ac.jp/naid/110010014760/

[10] T. Yamashita, M. Kimura, H. Fukui, Y. Yamauchi, and H. Fujiyoshi, “Optimal mini-batch procedure for facial piont detection based on a deep convolutional neural network,” in *The 21st Symposium on Sensing via Image Information*, 2015, (in Japanese).

[11] Y. Minagawa, M. Abe, and Q. Zhao, “Automatic face feature extraction based on neural networks,” in *SICE Tohoku 284*, vol. 284, no. 2, 11 2013, pp. 1–4, (in Japanese).

[12] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, “Learning deep representation for face alignment with auxiliary attributes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 5, pp. 918–930, May 2016.

[13] C. Garcia and M. Delakis, “Convolutional face finder: a neural architecture for fast and robust face detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1408–1423, Nov 2004.

[14] K. Sudhakar, P. Nithyanandam, “An accurate facial component detection using gabor filter,” *Bulletin of Electrical Engineering and Informatics*, vol. 6, no. 3, pp. 287–294, September 2017.

[15] Ri Cerd Ng, Kian Ming Lim, Chin Poo Lee, Siti Fatimah Abdul Razak, “Surveillance system with motion and face detection using histograms of oriented gradients,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 14, no. 2, pp. 869–876, May 2019.

[16] Srinivasa Perumal Ramalingam, Nadesh R. K., SenthilKumar N. C., “Robust face recognition using enhanced local binary pattern,” *Bulletin of Electrical Engineering and Informatics*, vol. 7, no. 1, pp. 96–101, March 2018.

[17] LFW, “Labeled Faces in the Wild (LFW),” http://vis-www.cs.umass.edu/lfw/.