A Comprehensive Study of Face Recognition Using Deep Learning

Koichi Ito*, Hiroya Kawai*, and Takafumi Aoki*
* Graduate School of Information Sciences, Tohoku University
6-6-05, Aramaki Aza Aoba, Sendai, 9808579, Japan.
E-mail: ito@aoki.ecei.tohoku.ac.jp

***Abstract*—With the advent of deep learning, the performance of face recognition has been dramatically improved. On the other hand, there are few reports that discuss why the performance has been improved. In this paper, through comprehensive experiments, we analyze which regions are important in CNN-based face recognition. We employ the major four CNNs, AlexNet, ResNet, and EfficientNet, to be able to perform face recognition and three CNN visualization methods, Grad-CAM, Grad-CAM++, and Score-CAM, to visualize the regions in the face image that are emphasized by each face recognition method.**

I. Introduction

In biometric recognition, physical features such as face, iris, fingerprint, and palmprint, and behavioral features such as handwriting and gait are used for personal authentication [1]. Face recognition is more convenient and cost effective than other biometric recognition such as fingerprint recognition and iris recognition since face recognition does not require special equipment to acquire face images [2]. Face recognition has been already applied to person authentication in smartphones, immigration control, automatic ticket gates, security gates, etc. because of these advantages.

The most traditional approach of face recognition is extracting geometric or texture features from a face image . Keypoints on a face are detected from an image, and the geometric relation among the keypoints are used as features [3]. Face recognition methods using feature descriptors extracted from around keypoints such as Gabor filter and Local Binary Patterns (LBPs) have been proposed [2], [4]. With the recent advent of deep learning, the performance of face recognition has been dramatically improved [5]–[7]. DeepFace [5] is a pioneering work on face recognition using deep learning, which demonstrated that facial features extracted by convolutional neural network (CNN) trained on 4.4 million labeled face images consisting of 4,030 individuals can provide recognition accuracy comparable to that of humans. The difference between the CNN-based methods and the traditional methods is the input to the discriminators. The input for traditional methods is handcrafted feature descriptors, while that for CNN-based methods is face images. CNNs can achieve high recognition accuracy by automatically extracting features suitable for face recognition, however, it is not clear what kind of facial features CNNs is used for face recognition.

One of the approaches for interpreting CNNs is the visualization of feature maps using the class activation map [8]–[10], which allows us to analyze which regions CNNs pay attention to in object recognition. In this paper, through comprehensive experiments, we analyze which regions are important in CNN-based face recognition. We train CNNs, AlexNet [11], ResNet [12], and EfficientNet [13], to be able to perform face recognition. We employ three CNN visualization methods, Grad-CAM [8], Grad-CAM++ [9], and Score-CAM [10], to visualize the regions in the face image that are emphasized by each face recognition model.

II. Visualization of CNN Models

The inference process of CNNs is much more complex than that of standard machine learning, and it is difficult to interpret it. Interpreting the inference process of CNNs is important to explore the causes of false recognition by CNNs as well as to improve the performance of CNNs. One of the approaches to interpret the inference process of CNNs is to visualize which regions CNNs pay attention to in their inference. In the following, we review some typical methods for visualizing the inference evidence of CNNs.

(i) Mahendran et al. [14]

When using CNN to classify images into multiple classes, CNN calculates a classification score for each class and outputs the class with the highest score as the estimation result. It is possible to visualize the inference evidence of CNN by identifying input images that have a large classification score for a particular class and analyzing these characteristics. In this method, the weight parameters of CNN are fixed and the input is optimized to maximize the classification score of a particular class. The gradient descent method is used as well as the standard CNN training method as an optimizer. While it is possible to visualize the typical features of each class, it is not possible to visualize the inference evidence for a specific input image.

(ii) Local Interpretable Model-agnostic Explanations (LIME) [15]

Simple machine learning models such as linear models are inferior in classification performance compared to complex models such as CNNs, while having high interpretability. LIME visualizes the inference evidence of CNNs by learning a new local approximation model around specific images and interpreting the approximation model. First, the input image is divided into superpixels, and several images are created with some of the superpixels masked. These images are input to

CNN to be visualized, and classification scores are calculated. Next, a linear model is trained to output the classification score of CNN using the masking pattern of the superpixels as input. The weight parameters of the trained linear model are used as a measure of the importance of the corresponding superpixels. By masking the superpixels with low weights, the regions that are important for CNN inference can be visualized. While visualization is possible regardless of the network architecture of CNNs, it is difficult to achieve precise visualization since the visualization results depend on the superpixels.

(iii) Guided-backpropagation (GBP) [16]

In training CNN, a loss is calculated between the output of CNN and the correct label, and the gradient is calculated for each weight parameter of CNN for the loss. Since the gradient represents the decrease in loss with increasing values of the weight parameters, the weight parameters can be updated to minimize the loss by multiplying the gradient by the learning rate and subtracting it from the weight parameters. If the same procedure is used to find the pixel-wise gradient of the input image for the loss, the gradient represents the impact of each pixel of the input image on the loss. In GBP, the per-pixel gradient of the input image to the loss is obtained by focusing on positive gradients. By visualizing the magnitude of the gradient as a heat map, we can visualize the information that has positively influenced the inference results. Similar to Mahendran et al. [14], which uses the gradient of the input, GBP can visualize the inference evidence for a specific input image. However, in recent years, it has been pointed out that the visualization results from GBP do not depend on the weight parameters of the CNN, and edge information is easily emphasized [17].

(iv) Class Activation Mapping (CAM) [18]

The first step in classifying using CNNs is to extract features using convolutional and pooling layers. The extracted features are converted to 1D vectors and input to all the convolutional layers to calculate the classification score. Some of the CNNs proposed so far for image classification use Global Average Pooling (GAP) to convert the extracted features to a 1D vector [12], [13], [19]. GAP is a method that aggregates features into a single value per channel by calculating the average value for each channel of the features. In most cases, GAP is used to reduce the features to a 1D vector, and then a single fully-connected layer is used to calculate the classification score. Therefore, the fully-connected layer can be regarded as a linear model that calculates the classification score for each class using the 1D features as input. The weight parameters of the fully-connected layer corresponding to each class are the measures of the importance of each channel of features in the calculation of the classification score. In CAM, the weighted sum of the features output by the final convolutional layer in the channel direction is calculated based on the weight parameters of the fully-connected layers, and output as a heat map. Although it can provide relatively detailed visualization, it is not very useful for general purpose since the target CNN needs to satisfy the aforementioned network architecture.

Several methods have been proposed to compute the channel-wise weights of features independent of the CNN network architecture [8]–[10]. Fig. 1 shows an overview of CAM and its advanced methods. Grad-CAM [8] and its extended method, Grad-CAM++ [9], use gradients for weighting, as well as GBP [16]. These methods generate a heat map by calculating the pixel-wise gradients of the features against the classification scores, weighting the features, and adding them in the channel direction. In Grad-CAM, the gradients calculated for each pixel are averaged for each channel and multiplied by each channel of the feature value. On the other hand, Grad-CAM++ multiplies the weighted sum of the gradients calculated for each pixel to each pixel of the feature.

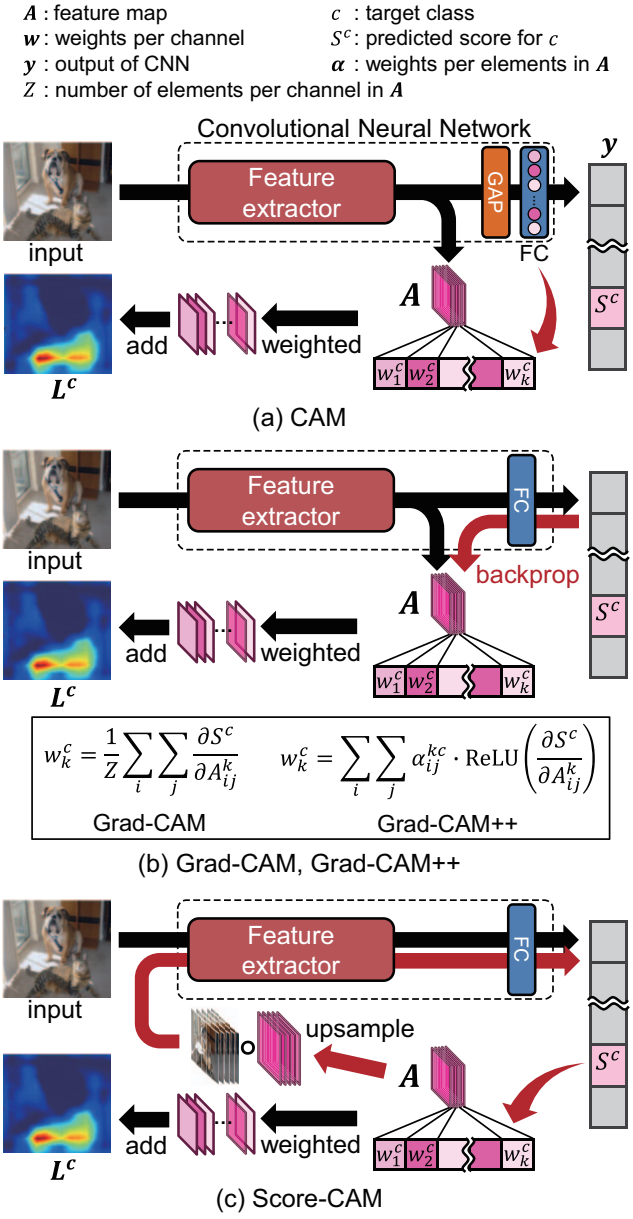

Fig. 1. Overview of CAM and its advanced methods.

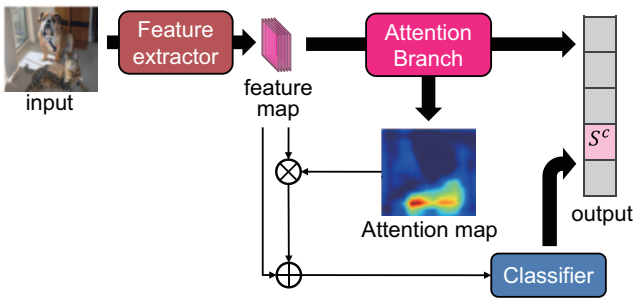

Fig. 2. Overview of ABN.

Another method that calculates the weights for each channel of the features using a method other than gradient is Score-CAM [10]. In Score-CAM, the features are normalized to the range $[0, 1]$ per channel and upsampled to the same size as the input image. The upsampled features are multiplied by the input image for each channel to generate mask images, and each mask image is input to CNN to obtain a classification score. The weight of the feature for each channel is determined based on the classification score.

(v) Attention Branch Network (ABN) [20]

As mentioned above, there are limitations in the network architecture of CNNs to which CAM [18] can be applied. ABN is the concatenation of another network (Attention Branch) that applies CAM to the target CNN. Figure 2 shows an overview of ABN. The target CNN is divided into a feature extractor and a classifier, and the output of the feature extractor is input to the attention branch. The attention branch generates a heat map (attention map) that represents important regions for inference as well as classifying classes. The output of the feature extractor is multiplied by the attention map, and the classifier performs class classification separately from the attention branch. By using the classification results for training, the performance of CNN can be improved while visualizing the inference evidence. On the other hand, when applying the visualization to trained CNN, it is necessary to train the concatenated attention branch separately, which requires more time and effort to visualize the inference evidence.

III. Face Recognition Using CNN

In this paper, we use AlexNet [11], ResNet [12] and EfficientNet [13] as backbone networks for face recognition using CNNs. We extract 512-D feature vectors from face images using each CNN and classify them using a classifier consisting of one fully connected layer. The feature vectors are extracted from a pair of face images using CNN, and the verification performance is evaluated based on the cosine similarity between the feature vectors. Note that, in this paper, the architecture after the final pooling layer of each network is changed to GAP layer, the batch normalization layer, the dropout layer (drop ratio 0.5), and the fully-connected layer (512 units). We describe the overview of each CNN in the following.

(i) AlexNet[1]: AlexNet was the top-performing network architecture in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012[2], consisting of five convolutional layers and two fully-connected layers. The total number of weight parameters in AlexNet is 2,601,792.

(ii) ResNet[1]: It has been reported that the performance of CNNs before ResNet does not improve even if the layer depth is increased beyond a certain level due to problems such as vanishing gradient. ResNet achieves deeper CNNs by introducing a skip structure called the shortcut connection, and achieves higher classification accuracy than conventional CNNs in image classification. In this experiment, we use ResNet-34, which contains 16 convolution blocks consisting of two convolution layers. The total number of weight parameters in ResNet is 21,548,352.

(iii) EfficientNet[3]: EfficientNet is automatically designed considering the balance of network depth (number of convolutional layers), breadth (number of output channels of convolutional layers), and resolution (input size of convolutional layers). EfficientNet achieves higher accuracy with fewer parameters than conventional CNNs. EfficientNet has a structure consisting of a large number of serialized convolutional blocks as proposed in MobileNets [19]. In this experiment, we use EfficientNet-B1, which consists of 23 convolutional blocks. The total number of weight parameters in the network is 6,266,016.

IV. Experiments and Discussion

In this section, we describe the training of face recognition methods and the analysis of their inference evidence. We generate several different face recognition methods consisting of different combinations of types of CNNs, with and without face detectors, and with and without pretraining using the ImageNet dataset [21]. Guided-backpropagation (GBP) [16], Grad-CAM [8], Grad-CAM ++ [9], and Score-CAM [10] are used to visualize the inference evidence of face recognition methods.

A. Datasets

For training and performance evaluation of the face recognition methods, we use the public datasets: VGGFace2 dataset [22] and Labeled Faces in the Wild (LFW) dataset [23], respectively. Each image is resized to 160×160 pixels, normalized to the pixels having the range $[-1, 1]$, and input to the CNN. For VGGFace2, 10% of the images are separated by ID and used as validation data. The validation data is used to control the learning rate and the number of epochs as a measure of the generalization performance of CNNs, and also to visualize the inference evidence.

(i) VGGFace2[4]: VGGFace2 consists of 3,310,000 face images taken from 9,131 people. The number of images per person is $80 \sim 843$ (average 362.6 images). The size of the images

[1]https://pytorch.org/docs/stable/torchvision/models.html
[2]http://image-net.org/challenges/LSVRC/2012/
[3]https://github.com/lukemelas/EfficientNet-PyTorch
[4]https://github.com/ox-vgg/vgg_face2

is not standardized, and is approximately 300 pixels per side. It is recommended that the data be separated into training data consisting of 3,140,604 images and test data consisting of 169,396 images. In this experiment, we use the test data (500 people) for training CNNs to reduce the training time.

(ii) LFW[5]: The LFW consists of 13,233 face images taken from 5,749 people. The number of images per person is $1 \sim 530$ (average 2.3 images). The image size is standardized to 250×250 pixels. In this experiment, we follow the official experimental protocol and use 6,000 pairs of 7,701 images (3,000 genuine pairs and 3,000 impostor pairs) for performance evaluation.

B. Evaluation Metrics

As mentioned above, in this experiment, we extract feature vectors from paired images using CNN and calculate the cosine similarity between the feature vectors. The closer the cosine similarity between the feature vectors is to 1, the smaller the angle between the two feature vectors, i.e., the more similar the two feature vectors are. A threshold value is set for the cosine similarity, and if the similarity is higher than the threshold value, the two images are considered as a genuine pair. If the similarity is lower than the threshold, the two images are considered as an impostor pair. These results are compared with the correct labels in the dataset, and the accuracy and error rate are calculated.

In this experiment, we use the accuracy, Equal Error Rate (EER) and Area Under the Curve (AUC) as evaluation metrics for face recognition. The accuracy is the maximum value of the correct response rate for all test data. The threshold is set so that the correct response rate is maximized. EER is the error rate such that False Acceptance Rate (FAR) and False Rejection Rate (FRR) are equal. FAR is the ratio of pairs that are incorrectly considered the genuine pairs among the impostor pairs, and becomes larger the lower the threshold is set. FRR is the ratio of pairs that are incorrectly considered the impostor pairs among the genuine pairs, and becomes larger the larger the threshold is set. A large FAR causes security problems, while a large FRR reduces the convenience of the biometric recognition system. In personal authentication, EER is used as the major evaluation metric since it is necessary to design an authentication system considering the balance between FAR and FRR. The curve with the True Acceptance Rate (TAR) on the vertical axis and the FAR on the horizontal axis is called the Receiver Operating Characteristic (ROC) curve. AUC is the area of the lower part of the ROC curve and takes values in the range $[0, 1]$. The closer AUC is to 1, the lower FAR and the higher TAR are, i.e., the more ideal the authentication system is.

C. Experimental Condition

If face detection is performed in preprocessing, Multi-task Cascaded CNN (MTCNN)[6] [24] is used as a face detector. We set a margin of 20 pixels when cropping face regions and select the detection regions that are close to the center of the image and have a large area preferentially. For pretraining, we download and use pre-trained models on the ImageNet dataset from the distributors of each network architecture. For optimization, Nesterov Accelerated Gradient (NAG) [25] is used. The initial learning rate is set to 0.01, and if the loss of validation data does not improve for five consecutive epochs, it is reduced by 10% of the original value. If the loss of the validation data does not improve for 10 consecutive epochs or after 100 epochs, the training is finished.

D. Recognition Accuracy for LFW

Table I shows the recognition accuracy of each face recognition model on the LFW dataset. AlexNet, ResNet-34, and EfficientNet-B1 achieve the highest recognition accuracy in this order. For all CNNs, the highest accuracy is achieved when both face detection and pretraining are used. The highest accuracy is achieved when only face detection is used, when only pretraining is used, and when both face detection and pretraining are not used, in that order. Most of CNNs achieve EER of less than 10%, which confirms that the training was done correctly.

E. Visualization of the Inference Evidence for Face Recognition CNNs

First, we compare the visualization methods. For ResNet-34 (no face detection and no pre-training), we input 10 face images randomly sampled from the validation data, and obtain the classification scores of the correct response classes. Based on the classification scores, visualization is performed using four visualization methods, and the visualization results are compared. Fig. 3 shows a comparison of the visualization methods. For GBP, we can observe that the edges such as the contours of the face and nose are emphasized. The area around the eyes is also emphasized for both face images. In Grad-CAM and Grad-CAM++, there is no significant difference in the output results, and the areas around the mouth, nose, cheeks, and eyes are highlighted for all the face images. In Score-CAM, there is a difference in the highlighted area for each face image. The highlighting area is more limited than in Grad-CAM. In addition to the areas around the mouth, nose, cheeks, and eyes that are similar to those in Grad-CAM, the forehead is strongly highlighted in some images.

Next, we compare the visualizations among the face recognition methods. In the same way, we input 10 face images to each face recognition model and visualize the inference evidence for each model using Score-CAM. Fig. 4 shows the comparison of the visualization results among the face recognition methods. Comparing the visualization results of AlexNet (without face detection and without pre-training) and EfficientNet-B1 (without face detection and without pretraining), it can be observed that common regions are highlighted for the same face image. On the other hand, the visualization results of AlexNet (with face detection) and EfficientNet-B1 (with face detection) are partially different. Some of the visualization results also differ depending on whether or not

[5]http://vis-www.cs.umass.edu/lfw/

[6]https://github.com/timesler/facenet-pytorch

TABLE I
SUMMARY OF RECOGNITION ACCURACY OF EACH CNN FOR LFW.

Network	Face detection	Pre-training	Accuracy [%]	EER [%]	AUC
AlexNet			88.55	11.50	0.953
		✓	89.65	10.51	0.961
	✓		93.22	7.033	0.981
	✓	✓	94.07	6.100	0.985
ResNet-34			88.92	11.33	0.959
		✓	93.23	6.833	0.981
	✓		93.37	6.700	0.982
	✓	✓	95.82	4.325	0.992
EfficientNet-B1			92.07	7.980	0.973
		✓	93.88	6.233	0.986
	✓		94.52	5.533	0.988
	✓	✓	**96.33**	**3.733**	**0.994**

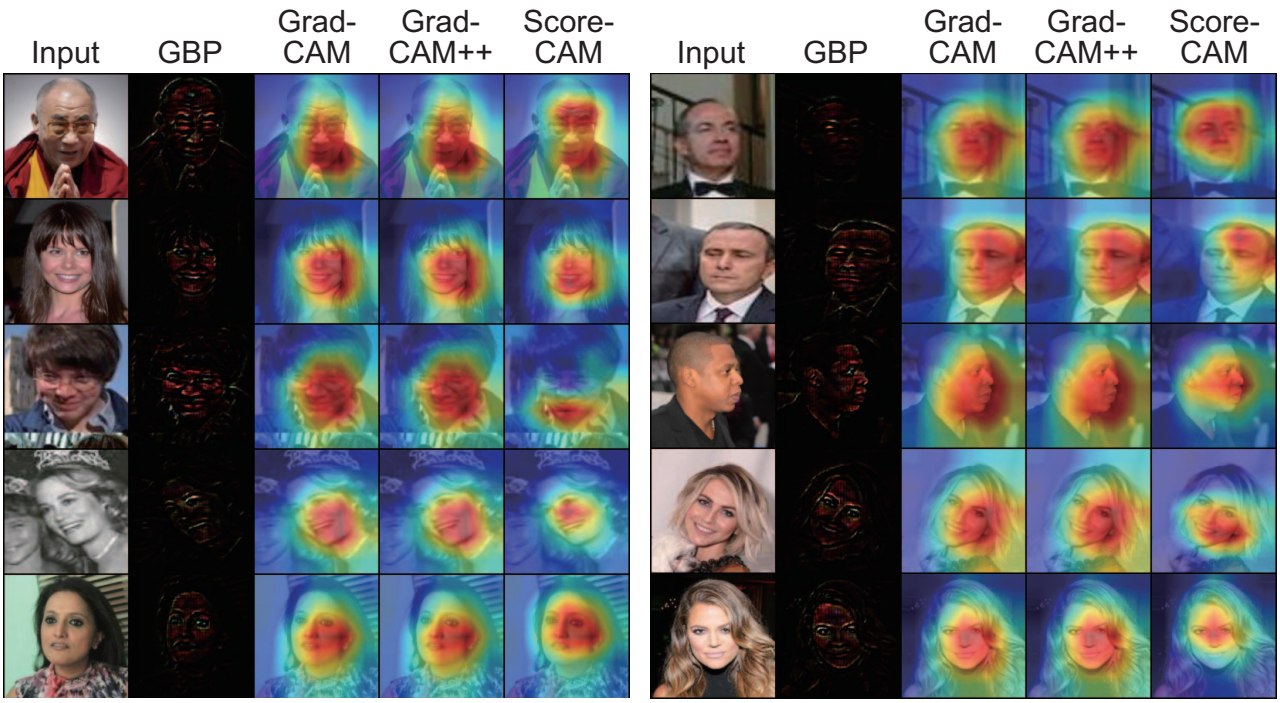

Fig. 3. Comparison of visualization results for ResNet-34 (no face detection and no pre-training).

pretraining is performed. These visualization results suggest that the part of the face that the face recognition method focuses on during recognition is not fixed and varies depending on the input image, CSSs, and the training method.

V. CONCLUSION

In this paper, we analyzed the visualization of the inference evidence of CNN-based face recognition methods through comprehensive experiments. We demonstrated that the part of the face that the CNNs focus on during recognition depends on the input image, the network architecture, and the training method. Based on these results, there is a possibility to improve the performance of face recognition methods by guiding the region of interest of CNNs through learning, like ABNs.

Although many researches focusing on recognition performance have been reported, the recent challenge is to improve the security for biometric recognition systems to prevent attacks from malicious third parties. Template protection and data encryption can protect the data inside the system, however, countermeasures against spoofed input data are indispensable, which is called anti-spoofing [26]. With the increasing use of cloud services such as social networking services, it has become common for users to post their face images in user profiles, making it easy to collect personal information including face images from the Internet. Therefore, a malicious third party can attack a face recognition system by using face images obtained on the Internet, and thus face anti-spoofing has been actively investigated as a countermeasure. In general, face anti-spoofing prevents face recognition systems from spoofing attacks by detecting whether the input image

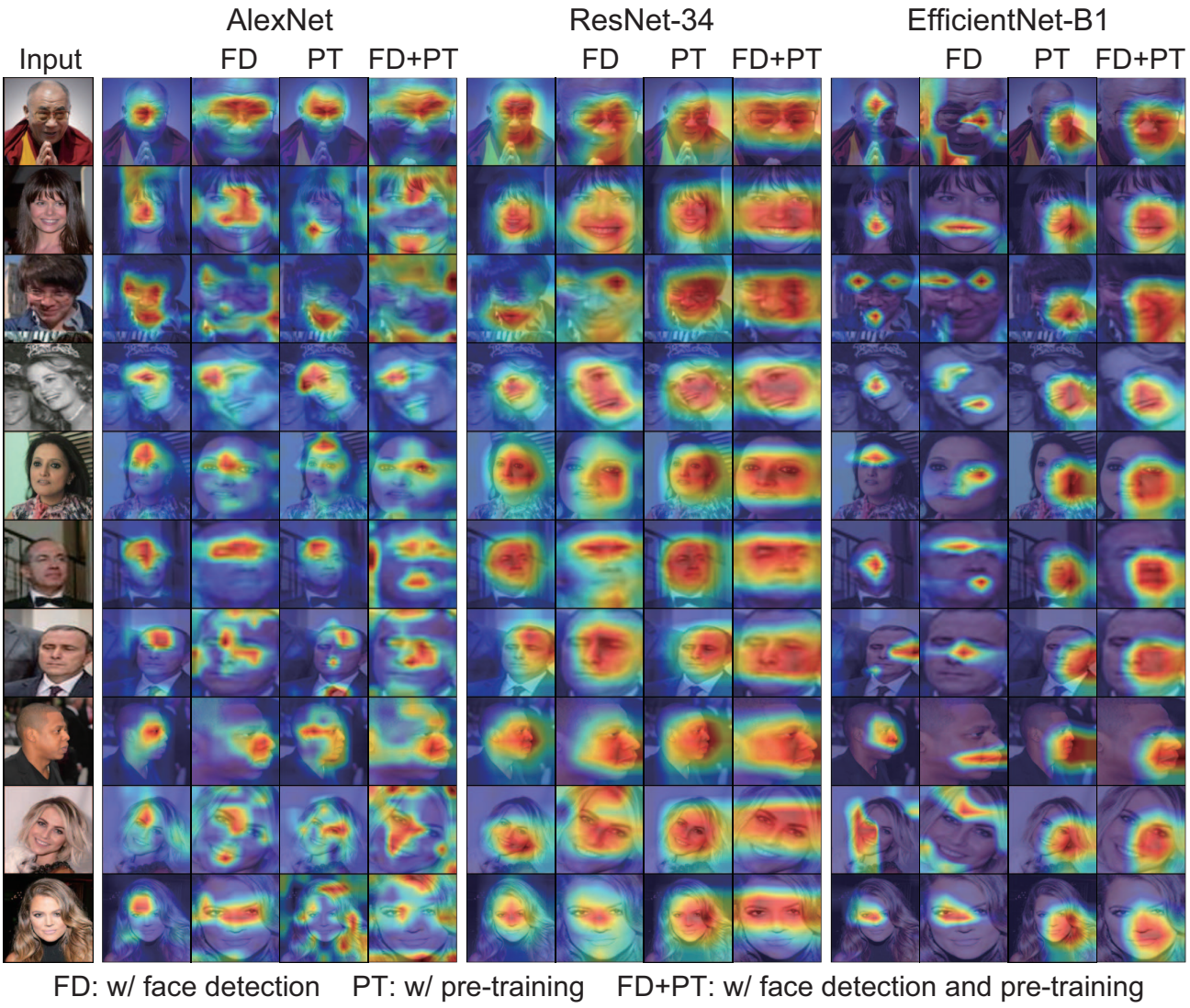

Fig. 4. Comparison of visualization results using Score-CAM.

is real or fake in advance. Face recognition systems must be improved in terms of security, while face images can still be obtained from the Internet. The results provided in this paper indicate which part of the face is recognized by CNNs, and therefore, the results could significantly contribute to improving the safety of face recognition methods using CNNs.

REFERENCES

[1] A. Jain, P. Flynn, and A. Ross, *Handbook of Biometrics*. Springer, 2008.

[2] S. Li and A. Jain, *Handbook of Face Recognition*. Springer, 2011.

[3] R. Jafri and H. R. Arabnia, "A survey of face recognition techniques," *J. Information Processing Systems*, vol. 5, no. 2, pp. 41–68, Jun. 2009.

[4] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.

[5] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1701–1708, Jun. 2014.

[6] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 815–823, Jun. 2015.

[7] J. Deng, J. Guo, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 4685–4694, Jun. 2019.

[8] R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parkih, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," *Proc. Int'l Conf. Computer Vision*, pp. 618–626, Oct. 2017.

[9] A. Chattopadhay, A. Sarkar, P. Howlader, and V. Balasubramanian, "Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks," *Proc. IEEE Conf. Applications Computer Vision*, pp. 839–847, Mar. 2018.

[10] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu, "Score-CAM: Score-weighted visual explanations for convolutional neural networks," *Proc. IEEE Conf. Computer Vision and Pattern*

Recognition Workshops, pp. 111–119, Jun. 2020.
[11] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Proc. Annual Conf. Neural Information Processing Systems*, pp. 1–9, 2012.
[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 770–778, Jun. 2016.
[13] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *Proc. Int'l Conf. Machine Learning*, vol. 97, pp. 6105–6114, Oct. 2019.
[14] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 5188–5196, Jun. 2015.
[15] M. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You?: Explaining the predictions of any classifier," *Proc. Int'l Conf. Knowledge Discovery and Data Mining*, pp. 1135–1144, Aug. 2016.
[16] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *CoRR*, vol. abs/1412.6806, Apr. 2015.
[17] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," *Proc. Annual Conf. Neural Information Processing Systems*, pp. 9525–9536, Dec. 2018.
[18] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 2921–2929, Jun. 2016.
[19] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, pp. 1–9, Apr. 2017.
[20] H. Fukui, T. Hirakawa, T. Yamashita, and H. Fujiyoshi, "Attention branch network: Learning of attention mechanism for visual explanation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 10 697–10 706, Jun. 2019.
[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 248–255, Jun. 2009.
[22] Q. Cao, L. Shen, W. Xie, O. Parkhi, and A. Zisserman, "VGGFace2: A dataset for recognising faces across pose and age," *Proc. IEEE Conf. Automatic Face and Gesture Recognition*, pp. 67–74, May 2018.
[23] G. Huang, M. Matter, H. Lee, and E. Miller, "Learning to align from scratch," *Proc. Annual Conf. Neural Information Processing Systems*, pp. 773–781, Dec. 2012.
[24] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, pp. 1499–1503, Oct. 2016.
[25] Y. Nesterov, "A method of solving a convex programming problem with convergence rate O(1/k2)," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.
[26] S. Marcel, M. Nixon, and S. Li, *Handbook of Biometric Anti-Spoofing*. Springer, 2014.