Auto Data Augmentation for Testing Set

Wanshun Gao[1] and Xi Zhao[2,3](✉)

[1] School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China
g-wanshun@stu.xjtu.edu.cn

[2] School of Management, Xi'an Jiaotong University, Xi'an 710049, China
zhaoxi@ieee.org

[3] The Key Lab of the Ministry of Education for Process Control & Efficiency Engineering, Xi'an 710049, China

Abstract. Testing phase augmentation is a fast way to further improve the performance of image classification when CNN (Convolutional Neural Network) is already trained for hours. Limited attempts have been made to find the best augmentation strategy for testing set. We propose a reinforcement learning based augmentation strategy searching method for testing phase augmentation. With the augmentation strategy, we augment each testing image and integrate features of its augmented images into one feature. The reinforcement learning method searches the best parameters in the augmentation strategy which is formed as a matrix in this paper. Using the proposed method, we achieve competitive accuracies on image classification and face verification.

Keywords: Image augmentation · Deep reinforcement learning · Face verification

1 Introduction

Data augmentation is a method which introduces unobserved data or features to help CNNs (Convolutional Neural Networks) achieve higher accuracy on image classification [2,3]. Data augmentation contains training phase and testing phase augmentation. Training phase augmentation transforms each training image to another image and feeds the transformed image to a CNN for training. Compared to training phase augmentation, testing phase augmentation is composed of image transformation [8,12,17,20] and feature fusion [4,8,11,12,17]. Image transformation transforms a testing image to multiple images for feature extraction. The features of these images are then merged to one feature with feature

Wanshun Gao is a student. This work is supported by the National Natural Science Foundation of China (Grant No. 91746111, Grant No.71702143), Ministry of Education & China Mobile Joint Research Fund Program (No. MCM20160302), Shaanxi provincial development and Reform Commission (No. SFG2016789), Xi'an Science and Technology Bureau (No. 2017111SF/RK005-(7)), the Fundamental Research Funds for the Central Universities, Tang Zhongying Foundation for Zhongying Young Scholars.

fusion. If the CNN is trained already, testing phase augmentation may further increase the classification accuracy.

In testing phase augmentation, recent researches apply limited image transformations [8,12,17,20] on testing images. Jia *et al.* and Parkhi *et al.* [8,17] crop each testing image into four corners, the center, and their mirrored versions from image size 256×256 to 224×224. In total, 10 patches are cropped from the testing image. Such cropping augmentation achieves higher accuracy than flipping augmentation [2]. DeepFace [20] transforms every testing image to its 3D aligned image, the gray-level image plus image gradient magnitude and orientation, and the 2D-aligned image. Masi *et al.* [12] transform the testing facial image to multi-pose facial images to produce identity preserving transformations. Above image transformations usually come from the experience of researchers. Other image transformations, such as auto contrast and rotation which are proven to increase the classification accuracy in training phase augmentation, are unexplored in existing testing phase augmentation works. Typically, an augmentation method comprises several image transformations and their magnitudes. There are many possibilities when finding a best augmentation method. Especially, for different testing sets, the best augmentation strategy may be different. For example, flipping augmentation strategy achieves more performance improvement on face identification [4] than it achieves on image classification [2]. Finding the best augmentation method from many image transformations, for the specific testing set, is still an unsolved problem.

In this paper, we propose a reinforcement learning method composed of an augmentation environment and a controller model to find the best testing phase augmentation strategy. The augmentation strategy comprises image transformations, their magnitudes and the type of feature fusion. The proposed method samples an augmentation strategy from the controller model as an action of the augmentation environment which invokes a state-of-the-art CNN model and outputs an evaluation score. The controller model takes in the evaluation score as a reward of the augmentation strategy, and updates itself with a batch of rewards. To directly use the state-of-the-art CNN whose deep learning framework may be incompatible with the controller model, we implement the reinforcement learning method to run on two deep learning frameworks which are coordinated to each other. The comparison between traditional methods and the proposed method is depicted as Fig. 1.

Our testing phase augmentation method achieves state-of-the-art performance on image classification datasets (CIFAR-10, CIFAR-100) and face verification datasets (LFW, CFP-FP and Age-DB30). Our contributions are:

- To the best of our knowledge, it is the first auto testing phase augmentation method. We design the augmentation strategy of testing phase, and search the best augmentation strategy using a reinforcement learning method.
- We implement the components of the proposed method to run on different learning frameworks.
- Competitive accuracies are achieved on image classification and face verification.

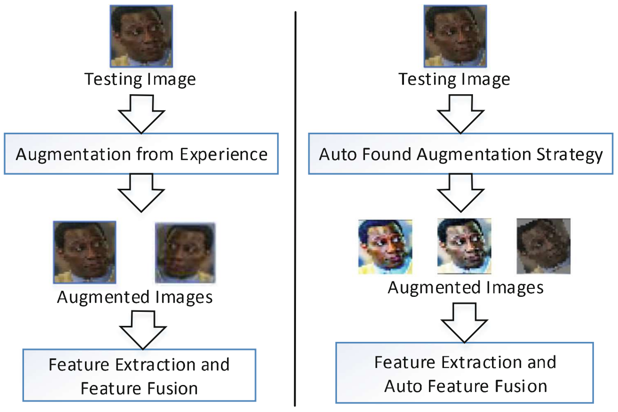
Testing Image
Augmentation from Experience
Augmented Images
Feature Extraction and Feature Fusion
Testing Image
Auto Found Augmentation Strategy
Augmented Images
Feature Extraction and Auto Feature Fusion

Fig. 1. Comparison of traditional methods (Left) with the proposed method (Right). The traditional methods augment testing images by experience. Using augmentation strategy which is found automatically, we augment testing images to more types, and combine the features of augmented images with the feature fusion found automatically.

Rest of this paper is organized as follows. Section 2 reviews recent works on image augmentation and parameter optimization. The differences between the proposed method and other works are demonstrated. Section 3 shows the framework and three parts of the proposed method. The details of the augmentation environment and the controller model are shown in each subsections of Sect. 3. Experimental results are list in Sect. 4.

2 Related Work

Image augmentation is widely used for image classification [8], emotion classification [24], semantic segmentation [10], face recognition [12], person re-identification [14] etc. In training phase augmentation, the early augment method is random corp, rotation, flip and so on. MixUp [6,21,23] augment examples and labels on convex combination of them. It reduces the effect of corrupt labels and stabilizes the training network. The augmentation method tries to increase the classification difficulty of permanent identities. It is interesting that, although they used strong in-plane augmentation at training time, improved recognition at test-time if face imagery is aligned with a 2D similarity transformation using detected face landmarks [13]. Autoaugment [3] designs its augmentation strategy as selecting possible image processing functions in order, and transforming an image with selected functions and their magnitudes. Such augmentation strategy lacks feature fusion, which is critical to testing phase augmentation. In testing phase augmentation, both feature pooling [4,8,12,17] and feature concatenation [11] are utilized to fuse features of augmented testing images. Inspired by Autoaugment, we design a matrix-like augmentation strategy for testing phase augmentation. For a testing set, we describe the augmentation strategy as a matrix. Each element in the matrix represents an operation, which is comprised of an image transformation function and the magnitude of the function. Elements in a row form an augmentation where features of transformed images are concatenated. Further more, we pool these concatenated features of all rows to one augmented feature, corresponding to a testing image.

In machine learning, hyper-parameters are predefined parameters before model training. Hyper-parameter optimization will result in the best performance when training a model. There are large amount of parameter optimization methods, such as Random Search, Bayesian Parameter Optimization, DRL (Deep Reinforcement Learning) etc. Bergstra [1] proposed a meta-modeling approach to optimize hyper-parameter automatically. Bayesian optimization is employed to optimize Expected Improvement (EI). It is first proven useful for optimizing hundreds of hyper-parameters. These methods [1,15] are able to optimize non fixed length hyper-parameters. Compared to them, Zoph and Le [25] proposed a more general and flexible method, which used a Recurrent Neural Network (RNN) interact with a child neural network and searched the hyper-parameters of the child neural network. The method is further extended to search more powerful neural architectures [18,26]. We inherit the idea of Zoph and Le [25] to automatically searching a testing phase augmentation strategy.

3 Methods

We decompose this section to three parts: augmentation environment, controller model θ_c and compatible implementation. Augmentation environment takes in an augmentation strategy and returns a reward to controller model. Controller model outputs augmentation strategy samples, and updates its weights with rewards. Compatible implementation helps augmentation environment running with independent deep learning framework, which may be different with controller model.

3.1 Augmentation Environment

Augmentation environment is to evaluate the performance of augmentation strategies that sampled from the controller model, by calling a trained CNN model θ_t. The evaluation score will return to controller model for weights updating. Based on the task of augmentation environment, the evaluation score can be classification accuracy or verification accuracy.

Existing image transformation functions in testing phase augmentation only contains crop, flip and grey-level. We extend it to all 19 image transformation functions in Python Imaging Library, which are “shearX/Y”, “translateX/Y”, “rotate”, “color”, “posterize”, “solarize”, “contrast”, “sharpness”, “brightness”, “autocontrast”, “equalize”, “invert”, “flipLR/UD”, “blur” and “smooth”. Additionally, we add “cutout” [5], and “identity”. The identity function means none of any image processing is done. It is important to keep original image in augmented images, and also convenient when doing non-augmentation evaluation.

To make image transformations functions accompany with two types of feature fusion, we formulize a testing phase augmentation strategy as an augmentation matrix T (Fig. 2). Each element $T_{i,j}$ represents an image that is augmented by an image transformation function and the magnitude of the function. We discrete the range of the magnitude to 10 values uniformly. $T_{i,:}$ or T_i (elements in row) represents concatenated augmentation, which means features of these augmented images are concatenated. $[T_1; T_2; ...; T_n]$ represent pooling augmentation which means all concatenated features are pooling to one feature, which is used for classification or verification. That is, in an augmentation matrix, one element has 210 options. If an augmentation matrix has 6 elements, it will be $210^6 \approx 8.58 \times 10^{13}$ options, which is very large.

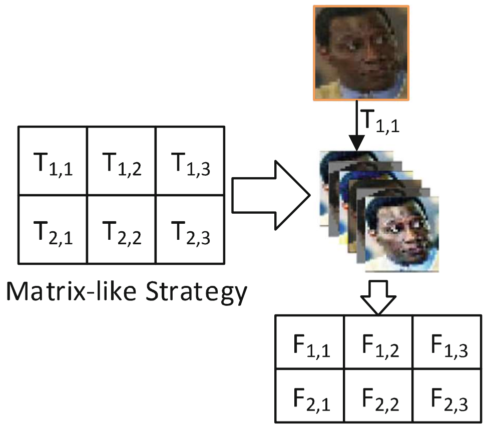

Fig. 2. Matrix-like augmentation strategy.

3.2 Controller Model

In the augmentation matrix T, concatenated augmentation T_i and pooling augmentation $[T_1; T_2; ...; T_n]$ are sequential. RNN (Recurrent Neural Network) is designed for the sequential data, which is appropriate to our augmentation method. In this paper, we adopt LSTM (Long Short Term Memory) as an implementation of RNN. To predict m elements in concatenated augmentation T_i, we set the size of initial input to m. Every time step in m time steps, a new softmax layer takes in the hidden state of LSTM, and predicts an elements in T_i. After m time steps, controller model predicts one concatenated augmentation T_i. With n times repetition above, controller model generates pooling augmentation $[T_1; T_2; ...; T_n]$. If $m = 1, n > 1$, controller model just generates pooling augmentation. The same with concatenated augmentation when $n = 1, m > 1$. The details are depicted as Fig. 3.

Generally, the reward is the accuracy of dataset. We try to maximize the reward to get higher accuracy. The purpose mainly focus on the maximization of the augmentation performance. So we exclude the bias of CNN performance, by minus the accuracy of raw images. The difference of augmented images and raw images is the reward to update controller. PPO (Proximal Policy Optimization) algorithm is used to update the weights of controller model.

It's the best to search the augmentation strategy on whole testing set. But it costs too much time to calculate a accuracy of whole testing set, even when the size of testing set is large. So we sample a mini-dataset from testing set for searching the test testing phase augmentation. The controller model aims to maximize the expected reward $\mathbb{E}_{a \sim \pi(a|C;\theta)}[r(a|C)]$. Here, the reward $r(a|C)$ is the accuracy on training images with a CNN model C. Several augmentation strategy a are sampled from the strategy $\pi(a|C;\theta)$. We apply the Adam optimizer to update the controller parameters θ.

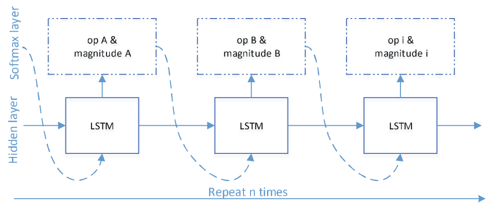

Fig. 3. The controller model. For each LSTM cell, we can sample an action from its softmax layer at the first step. The action represents an image operation and the magnitude of the operation. The next step, the last hidden layer and the action sample are inputs of the same LSTM cell. The output is the updated hidden layer and softmax layer. We sample a new action from the updated softmax layer. After m steps, we can obtains m sequential actions, which are applied for augment an image to another image. By repeat n times, we can obtains n augmented images.

The gradient is:

$$\nabla J(\theta) = \frac{1}{N} \sum_{i=1}^{N} [\nabla_\theta \log \pi(a_i|C;\theta)(r(a_i|C) - b_i)] \tag{1}$$

where, N is the number of samples in every controller step. Baseline b_i is a moving average of historical rewards. Augmentation strategy a_i is that augment a image to several images.

After training, we choose the augmentation strategies with high rewards during training. Then we test them on testing set with trained CNN models.

3.3 Compatible Implementation

Generally, DRL (Deep Reinforcement Learning) runs on single DLF (deep learning framework). In our case, both the controller model and the augmentation environment are deep neural networks. The augmentation environment invokes a state-of-the-art CNN model to evaluate augmentation strategies, while the CNN model is implemented with its own DLF which may not be the same with the DLF of the controller model. If migrating the DLF of the CNN model to the same DLF of the controller model, it will spend much time on CNN migration and CNN training. Also, it often exists differences when compared with the original CNN model, because of parameters tuning.

To use the state-of-the-art CNN directly, we implement the proposed DRL method with compatible mode. The controller model is implemented on TensorFlow which is widely used in academic and industrial circle. The concept of TensorFlow is building graph and running the graph to get results. Once the graph is built, it cannot be modified any more. Therefore, when the state-of-the-art CNN model in augmentation environment is ran on TensorFlow, we build the graphs of the controller model and the augmentation environment first. Then we run the graph of the controller model to sample augmentation strategy, and the graph of the augmentation environment to get evaluation result of the input augmentation strategy. When the state-of-the-art CNN model is ran on Mxnet, we only build the graph of the controller model.

To update the controller model with input rewards, we define the reward of the controller model as a Placeholder instead of a Tensor. Besides, we define another Placeholder for the corresponding augmentation strategy of the reward. We feed both the reward and its corresponding augmentation strategy to the controller model.

TensorFlow tends to allocate full GPU memory if not configured. We configure the GPU allocation of TensorFlow based on the framework of augmentation environment. When the augmentation environment runs on TensorFlow, we allocate the full GPU memory. When the augmentation environment runs on Mxnet, we allocate the GPU memory to minimal percent. In the experiment section, we allocate forty percent of whole GPU memory for TensorFlow.

4 Experiments

We evaluate the testing phase augmentation method by searching the best augmentation strategy on image classification dataset CIFAR-10 and three face verification datasets. Then, we evaluate the transferability of augmentation strategy by applying the best augmentation strategy of CIFAR-10 to CIFAR-100. Final, we evaluate the effective of compatible mode by demonstrating the GPU resource allocation under compatible mode and single mode. All experiments are ran on a Titan Xp GPU with 12GB GPU memory. When training the controller model, the learning rate is 0.35. The controller model samples 10 augmentation strategies in every training step, and samples 2000 augmentation strategies in total. The best augmentation strategy is selected from historical strategies which has the highest train accuracy.

4.1 Datasets

CIFAR-10 [9] has 10 classes with totally 50000 training images and 10000 testing images. To search the best augmentation strategy on testing images, we randomly divide 10000 testing images to 2048 images and 7936 images. 2048 images are used to training controller model, and 7936 images are used to test the best augmentation strategy that controller model has found.

CIFAR-100 [9] has 100 classes. Each class contains 500 training images and 100 testing images. Total 10000 testing images are used on the experiment of transferability of augmentation strategy.

Three face verification datasets are LFW [7], CFP-FP [19] and Age-DB30 [16]. The same as we did on CIFAR-10, we randomly separate 2048 pairs from original testing pairs of images for training controller model, and the rest of original testing pairs for testing.

4.2 Searching Augmentation Strategy on CIFAR-10

To evaluate the effective of the proposed method, we search the best augmentation strategy in six searching spaces with the possibilities from 210 to 8.58×10^{13}. In the augmentation matrix, the possibilities of searching space is $210^{m \times n}$, where m, n is the length of columns and rows. Theoretically, the proposed method supports unlimited size of the augmentation matrix. But, with the size increasing, the searching space is exponentially growing. It also needs more time to process testing images and find the best strategy. We limit the row length of the augmentation matrix no more than 6. The CNN in augmentation environment is trained with training set augmented by Autoaugment [3]. The architecture of CNN is Wide-ResNet-28-10 [22]. We apply pooling augmentation on the softmax layer of Wide-ResNet-28-10. The column length of the augmentation matrix can only be one. The results are demonstrated in Table 1.

As Table 1 shown, pooling augmentation is always effective, when the number of rows increase from 2 to 6. If we only transform original testing images without pooling augmentation (1 row), the accuracy of transformed images is no better than the accuracy on the original testing images. As the rows grows, the improved accuracy between training and testing becomes closer. Although the searching space is exponentially growing, we can easily find effective augmentation strategy in relatively small searching steps.

4.3 Searching Augmentation Strategy on Face Verification Datasets

We demonstrate the accuracies of the best augmentation strategy on three face verification datasets in Table 2. The CNN model (L50E-IR) in augmentation environment is trained by Arcface [4]. Compared with usual flipping augmentation, the proposed method can find better augmentation strategy on all three datasets. On CFP-FP, we increase the accuracy higher than other two datasets. It may because current alignment method is not suitable for profile-front face verification (Fig. 4).

Table 1. Evaluation of pooling augmentation with the number of rows growing.

	Train Mean (STD)	Test Mean (STD)
None	97.22	97.32
1 row	97.33 ± 0.07	97.22 ± 0.12
2 rows	97.58 ± 0.11	97.50 ± 0.18
3 rows	97.48 ± 0.11	97.34 ± 0.25
4 rows	97.55 ± 0.08	97.50 ± 0.10
5 rows	97.62 ± 0.08	97.61 ± 0.13
6 rows	97.59 ± 0.07	97.60 ± 0.09

Table 2. The improvement on three datasets with L50E-IR.

Dataset	Augmentation strategy	Acc.
LFW	None	99.70
LFW	[4]	99.80
LFW	Ours	99.82
CFP-FP	None	92.01
CFP-FP	[4]	92.76
CFP-FP	Ours	93.06
Age-DB30	None	97.60
Age-DB30	[4]	97.70
Age-DB30	Ours	97.80

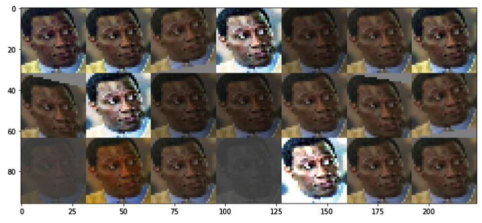

Fig. 4. The augment result on a facial images.

4.4 The Transferability of Testing Phase Augmentation Strategy

To evaluate the transferability of testing phase augmentation strategy found on CIFAR-10 with the CNN Wide-ResNet-28-10, we calculate the accuracy of the best augmentation strategy on testing set by replacing CIFAR-10 with CIFAR-100 or replacing Wide-ResNet-28-10 with Shake-Shake. Further, we calculate the accuracy by replacing both CIFAR-10 and Wide-ResNet-28-10 with CIFAR-100 and Shake-Shake. These results are shown in Tables 3, 4 and 5.

From Table 3, we can tell that the best augmentation strategy found on CIFAR-10 is transferable to CIFAR-100 which has the similar distribution with CIFAR-10. The accuracy increases more than it increases on CIFAR-10.

Table 3. Evaluate the transferability of testing phase augmentation strategy on CIFAR-100.

	Mean Accuracy (STD)
None	83.27
1 row	83.20 ± 0.31
2 rows	84.01 ± 0.37
3 rows	83.67 ± 0.42
4 rows	84.00 ± 0.26
5 rows	84.20 ± 0.23
6 rows	84.30 ± 0.13

Table 4 demonstrates that, we increase the accuracy on all Shake-Shake models. Although Autoaugment [3] already augment training set for training Shake-Shake models, we can also get a higher accuracy by using the best augmentation strategy found on CIFAR-10 with Wide-ResNet-28-10.

Table 4. Evaluate the transferability of testing phase augmentation strategy with Shake-Shake.

Test Aug	Shake-Shake 32	Shake-Shake 96
None	97.54	98.01
6 rows	97.74	98.39

When we replace both CIFAR-10 and Wide-ResNet-28-10 that used during controller model training, the best augmentation strategy also shows its effective. As Table 4 shows, the accuracy is higher than it on testing set without augmentation.

Table 5. Evaluate the transferability of testing phase augmentation strategy on CIFAR-100 with Shake-Shake.

Test Aug	Shake-Shake 32	Shake-Shake 96
None	82.88	84.89
6 rows	83.55	85.92

4.5 Compatibility of Controller Model and Augmentation Environment

To evaluate the compatibility of controller model and augmentation environment, we demonstrate the GPU resource allocation under compatible mode and single mode. In the experiment section, we employed two DL (deep learning) frameworks, TensorFlow and Mxnet. For CIFAR-10 and CIFAR-100, the DL frameworks of controller model and augmentation environment both are TensorFlow. For LFW, CFP_FF and AgeDB30, the DL framework of controller model is TensorFlow and augmentation environment is Mxnet (Table 6).

Table 6. Compatible of controller model and augmentation environment.

Environment	GPU resource
None	4919 M
TensorFlow	10775 M
Mxnet	10667 M

5 Conclusion

This paper proposed an auto testing phase augmentation method. It is effective to find the best augmentation strategy on the testing set, even the CNN model is trained using augmented training set. The best augmentation strategy also can be used on other datasets which has the similar distribution with the dataset where the best augmentation strategy was found on.

In further work, we will analysis the difference between training phase augmentation and testing phase augmentation, and design the parallelization of image transformations to speed the image processing time.

References

1. Bergstra, J., Yamins, D., Cox, D.D.: Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. JMLR (2013)
2. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: delving deep into convolutional nets. In: British Machine Vision Conference (2014)
3. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: learning augmentation policies from data. arXiv preprint arXiv:1805.09501 (2018)
4. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: additive angular margin loss for deep face recognition. arXiv preprint arXiv:1801.07698 (2018)
5. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017)

6. Guo, H., Mao, Y., Zhang, R.: Mixup as locally linear out-of-manifold regularization. arXiv preprint arXiv:1809.02499 (2018)
7. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: a database for studying face recognition in unconstrained environments. Technical report 07–49, University of Massachusetts, Amherst, October 2007
8. Jia, Y., et al.: Caffe: convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
9. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report. Citeseer (2009)
10. Liu, S., Zhang, J., Chen, Y., Liu, Y., Qin, Z., Wan, T.: Pixel level data augmentation for semantic image segmentation using generative adversarial networks. arXiv preprint arXiv:1811.00174 (2018)
11. Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., Song, L.: Sphereface: deep hypersphere embedding for face recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
12. Masi, I., Hassner, T., Tran, A.T., Medioni, G.: Rapid synthesis of massive face sets for improved face recognition. In: 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), pp. 604–611. IEEE (2017)
13. Masi, I., Wu, Y., Natarajan, T.H.P.: Deep face recognition: a survey. In: Conference on Graphics, Patterns and Images (SIBGRAPI), October 2018
14. McLaughlin, N., Del Rincon, J.M., Miller, P.: Data-augmentation for reducing dataset bias in person re-identification. In: 2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 1–6. IEEE (2015)
15. Mendoza, H., Klein, A., Feurer, M., Springenberg, J.T., Hutter, F.: Towards automatically-tuned neural networks. In: Workshop on Automatic Machine Learning, pp. 58–65 (2016)
16. Moschoglou, S., Papaioannou, A., Sagonas, C., Deng, J., Kotsia, I., Zafeiriou, S.: AgeDB: the first manually collected, in-the-wild age database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 51–59 (2017)
17. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: British Machine Vision Conference (2015)
18. Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient neural architecture search via parameter sharing. In: ICML (2018)
19. Sengupta, S., Chen, J.C., Castillo, C., Patel, V.M., Chellappa, R., Jacobs, D.W.: Frontal to profile face verification in the wild. In: IEEE Conference on Applications of Computer Vision, February 2016
20. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: closing the gap to human-level performance in face verification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1701–1708 (2014)
21. Verma, V., et al.: Manifold mixup: learning better representations by interpolating hidden states. Stat **1050**, 4 (2018)
22. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
23. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: Mixup: beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)

24. Zhu, X., Liu, Y., Li, J., Wan, T., Qin, Z.: Emotion classification with data augmentation using generative adversarial networks. In: Phung, D., Tseng, V.S., Webb, G.I., Ho, B., Ganji, M., Rashidi, L. (eds.) PAKDD 2018. LNCS (LNAI), vol. 10939, pp. 349–360. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93040-4_28
25. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)
26. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8697–8710 (2018)