Assessing Accuracy of Ensemble Learning for Facial Expression Recognition with CNNs

Alessandro Renda[1,2](✉), Marco Barsacchi[1,2], Alessio Bechini[1], and Francesco Marcelloni[1]

[1] Department of Information Engineering, University of Pisa, Via G. Caruso, 56122 Pisa, Italy
{alessio.bechini,francesco.marcelloni}@unipi.it
[2] University of Florence, Florence, Italy
{alessandro.renda,marco.barsacchi}@unifi.it

Abstract. Automatic facial expression recognition has recently attracted the interest of researchers in the field of computer vision and deep learning. Convolutional Neural Networks (CNNs) have proved to be an effective solution for feature extraction and classification of emotions from facial images. Further, ensembles of CNNs are typically adopted to boost classification performance.

In this paper, we investigate two straightforward strategies adopted to generate error-independent base classifiers in an ensemble: the first strategy varies the seed of the pseudo-random number generator for determining the random components of the networks; the second one combines the seed variation with different transformations of the input images. The comparison between the strategies is performed under two different scenarios, namely, training from scratch an ad-hoc architecture and fine-tuning a state-of-the-art model. As expected, the second strategy, which adopts a higher level of variability, yields to a more effective ensemble for both the scenarios. Furthermore, training from scratch an ad-hoc architecture allows achieving on average a higher classification accuracy than fine-tuning a very deep pretrained model. Finally, we observe that, in our experimental setup, the increase of the ensemble size does not guarantee an accuracy gain.

Keywords: Facial expression recognition · Convolutional Neural Network · Ensemble learning

1 Introduction

One of the most powerful communication tools is represented by human expressions: out of all the information exchanged in an oral communication, facial expressions account for 55%, whereas the plain language only for 7% [26]. Moreover, in 1971, Ekman et al. [4] showed that members of both preliterate and literate cultures use the same facial expression to convey any specific emotion. Human facial expressions of emotion are universal, related to biological and evolutionary factors rather than cultural or environmental ones. A set of distinctive patterns of the facial muscles characterizes each one of the so-called basic emotions: happiness, sadness, anger, fear, surprise, disgust. Thus, the Facial Expression Recognition (FER) problem has attracted the attention of the Computer Vision and Machine Learning communities: the ability to automatically perform FER over human facial images opens up the possibility to develop several applications in different fields, from Human Computer Interaction to Data Analytics, emotional health and sentiment analysis [17].

The core of an automatic FER system is represented by the *feature extraction* functionality, aimed at extracting a representative and discriminating set of features from the original facial images. Real-world applications ask for robust feature extractors, able to cope with image variations typical of an “in-the-wild” setting [3], such as occlusions, different head poses and illumination conditions. Hand-crafted feature extractors turned out to be inadequate for this challenging scenario, lacking the ability to generalize on incoming images: thus, the need has arisen for new, more flexible methods. As Deep Learning methods obtained excellent results in a wide variety of similar problems [15,16], their application in the context of FER has been explored as well. Convolutional Neural Networks (CNNs) can be regarded as one of the most popular models used for this purpose; they autonomously learn a hierarchical representation of the features of the original images [11]. The success of recent classification systems relies on the use of large collections of labeled data for training: 2012 ImageNet [2], for example, is a dataset of 1.4 million images with 1000 classes. On the other hand, annotating a large dataset of facial expression images is a difficult and time consuming task: FER2013 is one of the largest datasets of this kind built so far, and contains 35,887 images of different subjects.

A general, effective solution for boosting classification performance is represented by *ensemble techniques*, which combine multiple, *diverse* base learners (networks in our case). Several strategies have been proposed for the production of error-independent networks and for merging their classification outputs [5,12,24] but, to the best of our knowledge, in the FER context, their relative effectiveness has not been adequately investigated.

The present work is aimed at shedding light on the effectiveness of two simple techniques to generate diversity among the base classifiers of an ensemble: *Seed Strategy*, i.e. varying only the seed of the random number generator in the learning procedure of each network, and *Preprocessing Strategy*, combining the seed variation with different transformations of the input images. It is important to underline that different scenarios can be considered, and we perform this analysis in two of them: (i) training from scratch an ad-hoc architecture, *CNN10-S* (S stands for scratch), and (ii) fine-tuning a pre-trained state of the art model *VGG16-FT* (FT stands for fine-tuning). Both architecture were chosen for their recognized importance in the literature and availability to the research community [18,19]. It is worth pinpointing that the paper focus is on experimentally comparing different ensemble strategies, instead of achieving the best absolute accuracy on the FER-2013 dataset.

The remainder of the paper is organized as follows: in Sect. 2 we describe the typical approaches for training CNNs. In Sect. 3 we provide a detailed description of our experimental framework, from the used datasets to the proposed ensemble strategies, along with the scenarios for the comparison. In Sect. 4 we discuss the results of the experimentation, and finally Sect. 5 concludes the paper.

2 Brief Introduction to CNN Training Approaches

CNN [16] is a class of feed-forward neural networks: it is a convenient choice for input data with known topology, such as 2D or 3D pixel matrices that represent grayscale or RGB images, respectively.

As a Machine Learning model, the supervised learning procedure for CNNs aims to minimize the training error by experimenting a labeled dataset. However, the real objective is to perform well on new, unseen examples. To evaluate this generalization capability, a validation set is used during the training: several techniques are typically adopted to reduce the discrepancy between training and validation errors, such as dropout [21], data augmentation [23], and weight regularization [7]. Besides these, gathering and annotating more data is one of the best practices to reduce the risk of overfitting, but this is often difficult and time-consuming for many applications.

In the present work, we refer to a well-known, medium-size dataset (FER2013, described in Sect. 3.1). Against this background, two scenarios are taken into account: training an ad-hoc model from scratch, and using a pre-trained model. We tackled the FER problem following both the approaches.

2.1 Training a Model from Scratch

All the weights in the model are randomly initialized: they characterize the behaviour of every action unit. Along the training, an optimization algorithm, typically based on stochastic gradient descent (SGD) [7], iteratively updates the weights in order to minimize a cost function. In this scenario, the capacity of the model is carefully tuned, considering the limited size of the dataset.

2.2 Using a Pre-trained Architecture

Training from scratch a novel architecture on datasets of limited size has recently become unpopular [1]. Instead, a highly effective approach can be based on exploiting the pre-training of a large network, with higher capacity, over a big dataset, and then re-purposing such a network for the application of interest. Indeed, modern CNNs for Computer Vision show a common behaviour [25]: the features extracted in the first layers are quite standard and do not depend on the specific image dataset, while the high level features are strongly related to the considered task. Weights in the first layer typically learn filters that resemble fixed patterns, such as edge detectors, color blobs detectors, Gabor filters, etc. In the last few years, this approach has gained popularity mainly for two reasons: the availability of big labeled datasets for classification tasks, e.g. ImageNet with 1.4M images, and the availability of pretrained state of the art models such as VGG [20], Inception [22], and ResNet [10]. In our work, we use an already pretrained VGG16 model.

3 Experimental Setup

In this section, we describe the dataset used in the present work and the experimental approach. We recall that we want to compare two strategies for generating variability among base classifiers in an ensemble. In order to evaluate the general validity of the results, we perform the comparison in two typical scenarios: training from scratch an ad-hoc architecture, and fine-tuning a state of the art model. Experiments have been carried out over a server equipped with Nvidia GTX 1080 Ti with 11 GB Memory.

3.1 FER-2013 Facial Expression Dataset

The Facial Expression Recognition 2013 (FER-2013) dataset [8] has been chosen for our experiments because it is the most commonly adopted for this task, as reviewed in [19]: it is one of the largest collections of *in-the-wild* facial images consisting of 35.887 images from 7 classes: Neutral (6197), Anger (4945), Disgust (547), Fear (5121), Happiness (8988), Sadness (6076), and Surprise (4001). The official split of FER-2013 has been used after the removal of 11 black images, and it consists of a training set with 28699 images, a validation set with 3588 images, and 3589 images as test set.

The classification accuracy on FER2013 represents the performance measure of the models used in the present work. To the best of our knowledge, the best model achieves a 75.2% accuracy on FER2013 [19], while the average human accuracy on FER2013 is 65%.

3.2 Ensemble Design Strategies

There are two possible approaches for the design of an ensemble of neural networks [5,24]: the *implicit* (or *direct*) method aims to generate an ensemble of error-independent base classifiers by introducing one or more sources of variability. The *explicit* (or *overproduce and choose*) method involves a further optimization step: a subset of networks is selected from an initial large set by optimizing an error diversity measure out of selected base classifiers. In order to keep our model as simple as possible, we consider two direct ensemble design strategies: Seed Strategy and Preprocessing Strategy. We combine the outputs computed by the base classifiers by using the most common aggregation schemes: average and majority voting. For each strategy, a fixed-size ensemble of nine networks is used.

Seed Strategy (SE). The training procedure makes an extensive use of random choices, namely in the following operations: (i) initial distribution of weights; (ii) shuffle of the dataset; (iii) data augmentation; (iv) dropout.

This strategy thus exploits the simplest way to piece together an ensemble of diverse single CNNs: it sets a different seed value for the random number generator used in building up each base classifier, thus ensuring diversity across the members of the ensemble.

Preprocessing Strategy (PS). As proposed in [13], it makes use of another source of variability across the networks in the ensemble: a *preprocessing* layer is added before the CNN input stage. Nine networks are obtained by combining seed variability and preprocessing variability. Three different seeds are used in order to generate three networks for each of the following groups (Fig. 1):

– networks fed with the original, unchanged images (*default*);
– networks fed with images that underwent histogram equalization (histEq), which show an enhanced contrast with respect to the original ones [6];
– networks fed with images that underwent illumination normalization (iNor): it results in a smoothed version of the illumination-induced variations of the original images [9].

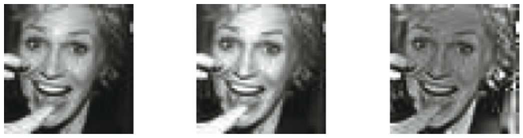

Fig. 1. The three versions of a sample image from FER2013 dataset adopted in the Preprocessing Strategy. *Left:* default, original image. *Center:* image modified by histogram equalization. *Right:* image modified by illumination normalization.

3.3 Two Scenarios of Interest: Adopted Models and Parametrization

The proposed strategies are evaluated on two typical scenarios: training from scratch an ad-hoc architecture, and fine-tuning a pre-trained model. Hereafter we describe the relative model, the preprocessing stage, the learning procedure, and the specific data augmentation step used to obtain a wider training set.

CNN10-S: Training from Scratch an Ad-hoc Architecture

Model. We trained from scratch a classical feed-forward CNN (Fig. 2): it is a 10-layers network resulting in 1,769,447 trainable parameters. It mimics the VGG-B architecture [20] by the Visual Geometry Group of the University of Oxford, modified with batch normalization layers and dropout layers according to the specification proposed by [19].

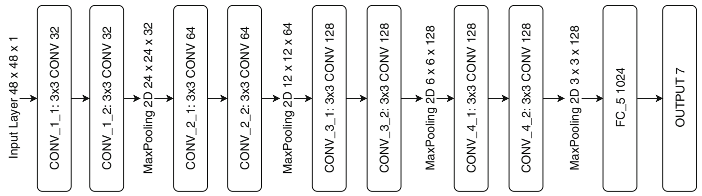

Fig. 2. Scheme of the architecture used in CNN10-S (10 is the depth, S stands for "scratch"). The network is entirely trained from scratch.

Preprocessing Stage. After applying one of the transformations described in Sect. 3.2, a global mean value μ, and a global standard deviation value σ were evaluated over the training set. The normalization step was performed by subtracting μ and dividing by σ. The transformation was then applied to every training, validation, and test image.

Learning Procedure. Following the approach proposed in [19], we used a stochastic gradient descent procedure (momentum = 0.9) to minimize the loss function; it is composed by a cross-entropy term and a L2 regularization term ($\lambda = 0.0001$). The batch size is 200 and the minimum number of epochs is 300. Since then, validation accuracy is monitored by stopping the training procedure after awaiting 20 epochs since the last improvement. The learning rate is a piecewise constant function of the training step (boundaries: [12000, 18000, 24000, 30000, 36000], values: [0.1, 0.05, 0.025, 0.0125, 0.00625, 0.003125]).

Data Augmentation. To artificially increase the training set size, every input image undergoes the following transformations: zero-padding from 48×48 to 54×54, and selection of a random crop of size 48×48; random horizontal flip with probability 0.5.

VGG16-FT: Fine-Tuning a Pretrained Model

Model. The reference pretrained model used in our framework is described in [18] and has been released as Caffe model by the Visual Geometry Group:

- the architecture is the VGG16 [20]: it is a 16-layers network resulting in 134,289,223 parameters; a dropout layer is added after FC_7 layer to reduce overfitting.
- the available weights of the model have been obtained by pretraining on a dataset for face recognition: the authors in [18] proposed a method for collecting and annotating 2.6M images from 2.6K different identities.

Since transfer learning is more successful when the source task and the target task are more similar, this pretrained model perfectly fits on our case-study, i.e. the classification of emotion from facial images.

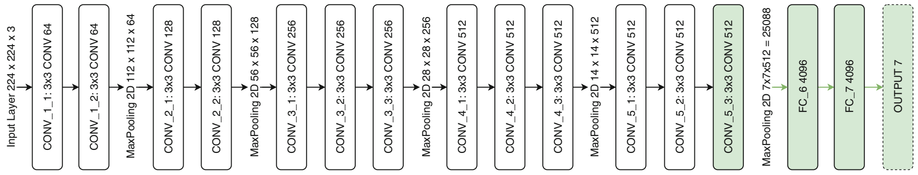

Fig. 3. The VGG16-FT architecture. During the first training step, only the output layer (dotted box on the right) is updated. During the second training step, the layers represented with filled boxes are fine-tuned.

Preprocessing Stage. The pretrained architecture, with a $224 \times 224 \times 3$ input layer, requires that grayscale images of FER2013 are upscaled to 224×224 and replicated on three channels. Images are then zero-centered by subtracting the global mean value μ.

Learning Procedure. The following steps are performed:

- the original output layer is removed (it was designed for another classification task). We add our custom output layer consisting of 7 units with softmax activation. Dropout is added before the output layer to reduce overfitting.
- *Step 1: Training the output layer.* The whole network, except the newly added output layer, is kept frozen, i.e. weights are not updated during training. Since the output weights are randomly initialized, the loss function is high in the first steps: including the convolutional layers in the learning procedure would damage the representations previously learned by such layers, because of a large error signal back-propagating through the network. The classifier is trained for 5 epochs using the Adam optimizer [14] with a learning rate of 0.00005 and a categorical cross-entropy loss function. In this step, the number of trainable parameters is 28,679. The batch size is 64.
- *Step 2: Fine-Tuning.* As shown in Fig. 3 all the hidden layers after *conv5_3* are unfrozen. Learning rate is halved and a new training procedure jointly fine-tunes these layers and the output added layer. In this step, the number of trainable parameters raises to 121,934,343.

Data Augmentation. To enable a fair comparison among the two scenarios, we perform the same data augmentation adopted for CNN10-S: images are zero-padded to size 256×256. A random crop of size 224×224 is extracted from either the padded image or its horizontally flipped version.

4 Experimental Results

In this section we show the experimental results: the performance of the proposed models is evaluated in terms of accuracy on the FER2013 test set. For both the ensemble strategies and the model architectures, three groups of nine networks were trained and evaluated in order to assess the stability of the measures.

Being able to rely on batches of 27 networks, we thus further investigated the accuracy obtained increasing the ensemble size. Results are reported in Table 1 and summarized in Fig. 4.

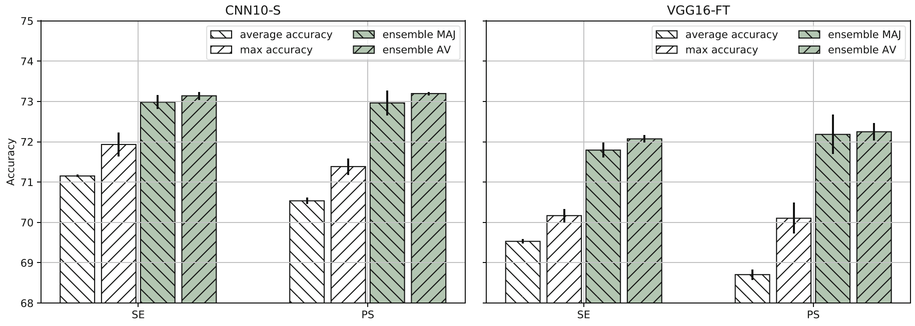

Fig. 4. Base classifiers (white bars) and ensemble (filled bars) accuracy for both strategies. In each group, the former white bar represents average of base classifiers accuracy, the latter represents the best base learners accuracy; the former filled bar represents ensemble accuracy using average voting, the latter represents ensemble accuracy using majority voting. *Left:* trained from scratch CNN10-S. *Right:* fine-tuned VGG16-FT.

Comparison Between CNN10-S and VGG16-FT. A first result is that the architecture CNN10-S (Table 1A), trained from scratch, achieves better performance than the architecture VGG16-FT (Table 1B), used as a pretrained model with fine-tuning (Fig. 4): the discrepancy between the accuracy values (between 1% and 2%) is confirmed both in terms of base and ensemble classifiers. Nevertheless, the two scenarios share common trends: they are analyzed in the following paragraphs.

Accuracy of Base Classifiers. For each strategy we can rely on three groups of nine networks. The low standard deviation of the average accuracy inter-groups suggests that results are fairly stable, independently of the strategy and the architecture. Both using CNN10-S and VGG16-FT, we observed that networks of SE strategy achieve better performance than networks of PS strategy (Fig. 4, white bars). Furthermore, the intra-group analysis suggests that networks of the PS strategy have a higher standard deviation, especially for VGG16-FT. Indeed, we noticed that the introduction of histogram equalization and illumination normalization leads to a slight performance drop compared to the use of default images.

Ensemble Accuracy. We define the ensemble gain as the difference between ensemble accuracy and average base classifier accuracy: combining preprocessing

Table 1. Accuracy of base classifiers and ensembles (average and majority voting), and the respective ensemble gains, are reported for each repeated measure. The mean and the standard deviation values are reported for each strategy.

A. Results obtained by training CNN10-S from scratch					
Network architecture CNN10-S					
	Base classifiers	Ensemble AV.	Gain AV.	Ensemble MAJ.	Gain MAJ.
SE	71.165 ± 0.520	73.084	1.919	72.862	1.697
	71.180 ± 0.495	73.279	2.099	72.862	1.681
	71.109 ± 0.275	73.057	1.947	73.224	2.114
Mean ± Std	71.152 ± 0.031	73.140 ± 0.099	1.989 ± 0.079	72.982 ± 0.171	1.831 ± 0.201
PS	70.465 ± 0.585	73.140	2.675	72.973	2.508
	70.496 ± 0.472	73.224	2.727	72.583	2.087
	70.645 ± 0.649	73.224	2.579	73.335	2.690
Mean ± Std	70.535 ± 0.078	73.196 ± 0.039	2.660 ± 0.062	72.964 ± 0.307	2.428 ± 0.253
B. Results obtained by fine-tuning VGG16-FT					
Network architecture VGG16-FT					
	Base classifiers	Ensemble AV.	Gain AV.	Ensemble MAJ.	Gain MAJ.
SE	69.546 ± 0.320	71.942	2.396	71.580	2.034
	69.462 ± 0.333	72.137	2.675	71.775	2.313
	69.583 ± 0.487	72.137	2.554	72.026	2.443
Mean ± Std	69.530 ± 0.050	72.072 ± 0.092	2.542 ± 0.114	71.793 ± 0.182	2.263 ± 0.170
PS	68.636 ± 1.187	72.388	3.752	72.527	3.892
	68.592 ± 0.941	71.942	3.350	71.496	2.904
	68.886 ± 0.403	72.416	3.529	72.527	3.641
Mean ± Std	68.705 ± 0.130	72.249 ± 0.217	3.544 ± 0.165	72.184 ± 0.486	3.479 ± 0.419

and seed variability ensures a higher gain value than just varying the seed. Nevertheless, in both the scenarios, PS strategy and SE strategy lead to very close ensemble performances (Fig. 4, filled bars). Despite being based on a deeper model, ensemble learning in VGG16-FT proves to be more effective than in CNN10-S, since it shows higher ensemble gain.

Even if the adopted aggregation schemes (average and majority voting) lead to comparable results, average voting shows slightly higher performance: in our framework, with low intra-group accuracy variability, average voting represents the proper choice. Indeed, majority voting is typical less sensitive to the output of a single base classifier since it considers only the predicted labels.

It is worth noting that each ensemble achieves better performance than the best base classifier composing it.

Increasing the Number of Base Classifiers. Let A, B, C be the three groups of networks produced for each strategy. We could rely on 3 ensembles of 9 networks (A, B, C), 3 ensembles of 18 networks (AB, AC, BC), and 1 ensemble of 27 networks (ABC). Figure 5 shows the results.

PS strategy shows a slight boost in performance with CNN10-S (+0.110%), but a drop using VGG16-FT (−0.028%). On the other hand, SE strategy shows a promising trend with VGG16-FT (+0.316%) while decreases with CNN10-S (−0.139%). Values in brackets are obtained by subtracting the accuracy value of the 27-nets ensemble and the 9-nets ensemble, considering average voting.

From the above considerations and by analyzing the trend shown in the plot, it is not possible to state that the increase of the number of base classifiers considerably improves, in general, the performance of our ensembles. Further, using the proposed model and the adopted parametrization, the training procedure is extremely time-consuming.

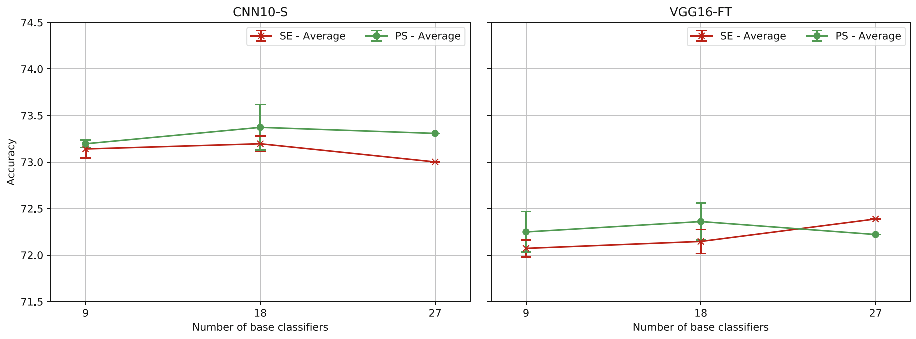

Fig. 5. Ensemble accuracy values versus number of base classifiers in the ensemble. Mean and standard deviation of three values are available for the ensembles with 9 and 18 networks, while a single value is available for the ensemble with 27 networks. For each strategy, we considered only average voting. *Left:* trained from scratch CNN10-S. *Right:* fine-tuned VGG16-FT.

5 Conclusion

In this paper we evaluated the performance of two design strategies for generating ensembles of CNNs used to tackle the FER problem, namely the Seed Strategy and the Preprocessing Strategy. The former generates diversity among base classifiers by simply varying the seed; the latter combines different values of the pseudorandom number generator with the introduction of different transformations of the input images.

Using a well known medium-sized dataset (FER2013), we carried out our comparison following two approaches: training an ad-hoc model from scratch (CNN10-S) and fine-tuning a pretrained model (VGG16-FT).

Results have shown that the ad-hoc architecture is an appropriate choice for the considered task, since it performs better than the fine-tuned model, both considering base classifiers and ensemble accuracy. Nevertheless, using a pretrained model requires less effort.

In the presented experimental setup, Seed Strategy and Preprocessing Strategy achieve comparable results using both the approaches (CNN10-S and

VGG16-FT). However, the variability induced by the Preprocessing strategy allows obtaining significantly higher ensemble gain than using the solely seed variation.

To the best of our knowledge, this is the first work that analyze the effectiveness of simple ensemble strategies using Deep Learning approaches for the FER task. Since we did not make specific assumptions based on the facial images, it could represent a starting point for further investigation also in other Computer Vision classification tasks.

In future work, we will investigate if other models, which use the same or other pretraining datasets, allow achieving comparable or better performance. Further, we will analyze the performance of other state of the art models and will evaluate the effect of introducing other factors of variation in the design of ensemble strategies, considering also their computational load.

References

1. Chollet, F.: Deep Learning with Python. Manning Publications Co., Shelter Island (2017)
2. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2009, pp. 248–255. IEEE (2009)
3. Dhall, A., Goecke, R., Joshi, J., Sikka, K., Gedeon, T.: Emotion recognition in the wild challenge 2014: baseline, data and protocol. In: Proceedings of the 16th International Conference on Multimodal Interaction, pp. 461–466. ICMI 2014. ACM (2014). https://doi.org/10.1145/2663204.2666275
4. Ekman, P., Friesen, W.V.: Constants across cultures in the face and emotion. J. Pers. Soc. Psychol. **17**(2), 124 (1971)
5. Giacinto, G., Roli, F.: Design of effective neural network ensembles for image classification purposes. Image Vis. Comput. **19**(9), 699–707 (2001)
6. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 3rd edn. Prentice-Hall Inc., Upper Saddle River (2006)
7. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016). http://www.deeplearningbook.org
8. Goodfellow, I.J., et al.: Challenges in representation learning: a report on three machine learning contests. In: Lee, M., Hirose, A., Hou, Z.-G., Kil, R.M. (eds.) ICONIP 2013. LNCS, vol. 8228, pp. 117–124. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42051-1_16
9. Gross, R., Brajovic, V.: An image preprocessing algorithm for illumination invariant face recognition. In: Kittler, J., Nixon, M.S. (eds.) AVBPA 2003. LNCS, vol. 2688, pp. 10–18. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44887-X_2
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
11. Hertel, L., Barth, E., Käster, T., Martinetz, T.: Deep convolutional neural networks as generic feature extractors. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–4, July 2015. https://doi.org/10.1109/IJCNN.2015.7280683

12. Ju, C., Bibaut, A., van der Laan, M.J.: The relative performance of ensemble methods with deep convolutional neural networks for image classification. arXiv preprint arXiv:1704.01664 (2017)
13. Kim, B.K., Dong, S.Y., Roh, J., Kim, G., Lee, S.Y.: Fusing aligned and non-aligned face information for automatic affect recognition in the wild: a deep learning approach. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 48–57 (2016)
14. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015). https://arxiv.org/abs/1412.6980
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
16. LeCun, Y., Kavukcuoglu, K., Farabet, C.: Convolutional networks and applications in vision. In: Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 253–256. IEEE (2010)
17. Martinez, B., Valstar, M.F.: Advances, challenges, and opportunities in automatic facial expression recognition. In: Kawulok, M., Celebi, M.E., Smolka, B. (eds.) Advances in Face Detection and Facial Image Analysis, pp. 63–100. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-25958-1_4
18. Parkhi, O.M., Vedaldi, A., Zisserman, A., et al.: Deep face recognition. In: BMVC, vol. 1, p. 6 (2015)
19. Pramerdorfer, C., Kampel, M.: Facial expression recognition using convolutional neural networks: state of the art. arXiv preprint arXiv:1612.02903 (2016)
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015). https://arxiv.org/abs/1409.1556
21. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
22. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
23. Wang, J., Perez, L.: The effectiveness of data augmentation in image classification using deep learning. Technical report (2017)
24. Wen, G., Hou, Z., Li, H., Li, D., Jiang, L., Xun, E.: Ensemble of deep neural networks with probability-based fusion for facial expression recognition. Cogn. Comput. **9**(5), 597–610 (2017). https://doi.org/10.1007/s12559-017-9472-6
25. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems, pp. 3320–3328 (2014)
26. Zhang, T.: Facial expression recognition based on deep learning: a survey. In: Xhafa, F., Patnaik, S., Zomaya, A.Y. (eds.) IISA 2017. AISC, vol. 686, pp. 345–352. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-69096-4_48