Faces Emotion Classification with Bimodal Distribution Removal based on ResNet

Shun Wang
Research School of Computer Science, Australian National University ACT, Australia
u6261174@anu.edu.au

Abstract. Faces emotion classification task is difficult to achieve high classification accuracy, especially on small databases. ResNet is been proved to be a good neural network structure to address image classification tasks. It solves the degradation problem caused by the depth of neural network by implementing the unique residuals block so as to achieve higher accuracy. Outliers will impact the performance of neural networks, especially for small data sets. By removing the outliers in the training set, the Bimodal Distribution Removal (BDR) algorithm is a potential solution to improve the performance and speed up the training process. In this paper, I apply BDR algorithm and improved ResNet50, and make a comparison of their performance on this issue. The result shows that the improved ResNet50 with the BDR achieves 46% accuracy, the improved ResNet50 without the BDR achieves 42% accuracy. In the end, this paper concludes that the improved ResNe50 does an excellent job on image classification, while the BDR improves the accuracy by 4%

Keywords: Faces Emotion Classification, ResNet, Dropout, Bimodal Distribution Removal, Neural Network

1 Introduction

1.1 Background

In the previous research, three methods are used to accomplish face emotion classification: Simple neural network without the BDR, simple neural network with the BDR and deep neural network with BN[4] and dropout[5]. The data set is SFEW[1] which has extracted the features by LPQ[2] and PHOG[3]. The results showed that accuracy in the test set was 25%, 27% and 37% respectively.

In this paper, ResNet[11] is used to accomplish the task considering its great performance on image classification tasks. ResNet became well known in 2015 and influenced the development of Deep Learning in academia and industry since then.

We know that in computer vision, the "rank" of features increases with the depth of the network, and research shows that the depth of the network is an important factor in achieving good results. However, gradient diffusion/explosion is a big problem during the training process, resulting in no convergence. As a famous CNN structure, ResNet is been improved from VGG[12], its residual network holds a reference to the input of each layer, learning to form residual function, rather than just learning some functions without reference. This form of unique residual learning can solve the degradation problem caused by depth, making a deeper network become possible. The data set is the original SFEW dataset without extracting features.

1.2 Dataset description

SFEW2.0 (Facial Expressions in the wild)[1] is a dynamic facial expressions dataset, graphics in it are taken from movies. The dataset used in this paper is SFEW2.0 with face alignment. It includes 891 pictures in the training set, 431 pictures in the validation set and 372 pictures in the test set. Since the test set is not labelled, I decide to not use the test set here, instead, I divide the validation set into test set and validation set. See 2.1 for details.

The whole dataset contains 7 types of facial expressions which are “Angry”, “Disgust”, “Fear”, “Happy”, “Neutral”, “Sad” and “Surprise”. Fig. 1 shows how the image is processed by face alignment operation. Since the original data are cut from movies, the background of the picture, the scene of the film and the clothes and actions of the characters all have great impacts on the result, while face alignment can mitigate the effects of these factors.

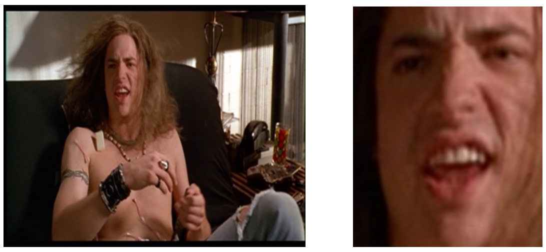
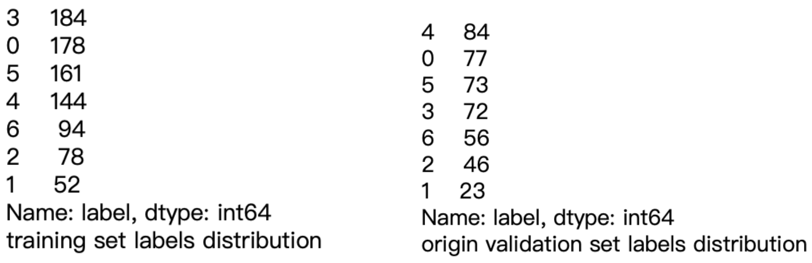


Fig. 1. The left shows the original image, the right shows image after alignment

Following figure 2 shows the image distribution after face alignment



```
3    184
0    178
5    161
4    144
6     94
2     78
1     52
Name: label, dtype: int64
training set labels distribution
```

```
4    84
0    77
5    73
3    72
6    56
2    46
1    23
Name: label, dtype: int64
origin validation set labels distribution
```

Fig. 2. Image distribution after face alignment

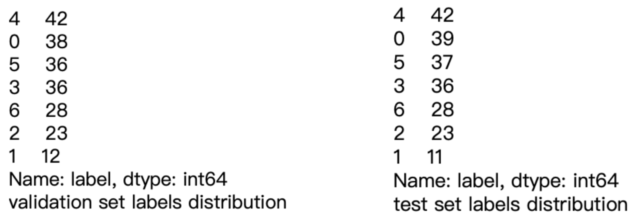
2 Method

2.1 Data pre-process

A high-quality data set is a vital element when it comes to training models. So I need to process the data before I start the model training process. In this paper, I adopt three methods to preprocess the data.

Firstly, in order to facilitate the later data loading process, I need to match the image with labels. I marked "Angry", "Disgust", "Fear", "Happy", "Neutral", "Sad" and "Surprise" as "0", "1", "2", "3", "4", "5" and "6" respectively. At this stage, it generates train_labels.csv and val_labels.csv.

Secondly, as I mentioned before, I decide not to use the original test set as it's not labelled. Therefore, I use random sampling to divide the validation set into validation set and test set with sampling rates of 0.5 and 0.5 respectively. That means the new validation set takes about 50% of the data, the test set takes about 50%. As fig 3 shows.



```
4    42
0    38
5    36
3    36
6    28
2    23
1    12
Name: label, dtype: int64
validation set labels distribution
```

```
4    42
0    39
5    37
3    36
6    28
2    23
1    11
Name: label, dtype: int64
test set labels distribution
```

Fig. 3. Distribution of new validation set and test set

Thirdly, since the data set is relatively small, Data Augmentation[6] is conducted during the training of the network, including Resize, HorizontalFlip, RandomBrightness, ShiftScaleRotate, Compression, HueSaturationValue, Normalize. these Data Augmentation operations are based on a package called albumentations[15]. In addition, since I use Resnet50 in this paper, I need to resize the input into 224*224.

2.2 Improved ResNet

It is a 7 classes image classification problem, ResNet seems a good choice. In this paper, we use the pre-trained net50 implemented by pytorch. The reason I choose the pre-trained model is to reduce the training time[16]. Although this pre-trained process is based on the ImageNet dataset, not very related to the dataset we use in this paper. But still, using the pre-trained parameters here to initialize my neural network can reduce the training time and achieve better results.

In the process of implementation, some adjustments are made to the structure. I change "the last full connect layer of ResNet50" into "the full connect layer with dropout", which is mainly to handle the overfitting problem. More specifically, I add dropout(0.6) before the full connect layer, followed by the first full connect layer which has 2048 input features and 512 output features. then the activation function layer SELU, followed by dropout(0.8). Then there is the second full connect layer which has 512 input features and 7 output features. The structure is shown as Fig.4.

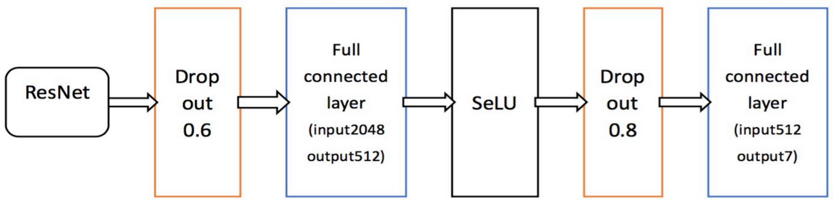

ResNet
Drop out 0.6
Full connected layer (input2048 output512)
SeLU
Drop out 0.8
Full connected layer (input512 output7)

Fig. 4. Structure of Improved ResNet50

Since it's a multi-classification task, the decision boundary of this problem is probably not linearly separable. So I decide to add a nonlinear activation function between the input layer and the hidden layer. The SeLU[13] function seems to be a proper nonlinear activation function because the distribution of the output of SeLU can be normalized to 0 mean and unit variance, so as to ensure that the gradient will not explode or disappear during the training process. Compared with ReLU, the similarity is that they are all gentle in the negative half-axis, so when the variance of activation is too large, it can be reduced to prevent the gradient explosion. The difference is that the positive half-axis of SeLU is greater than 1, which means the variance can be increased when it's too small, so as to prevent the gradient disappearance. So the activation function has a fixed point, and even if the network gets deeper, the output of each layer still remains 0 as means and 1 as the variance. The formula of SeLU function is shown below.

$$\text{selu(x)} = \lambda \begin{cases} x & if\ x > 0 \\ \alpha e^{x} - \alpha & if\ x \leq 0 \end{cases} \tag{1}$$

As for the loss function, I use Cross Entropy Loss because it's very suitable for multi-classification tasks[7]. The Cross Entropy Loss function in pytorch is shown as below.

$$\text{loss(x, class)} = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) \tag{2}$$

Where "x" represents the predicted value which is the 7-dimensions output here, "class" represents the target. It needs to be noted that the "class" here can only be expressed with scalar rather than one-hot form. "j" represents the index of the output vector.

The optimizer I use here is Adam[14] with learning rate at 0.0001. The reason why I choose Adam is that it has a faster convergence speed and better effect compared with other optimizers.

Since the training set is so small, serious overfitting will often occur if we don't do anything about it. Therefore, I also use dropout to prevent overfitting. In addition, I use batch learning in the training process, so that the gradient can be more accurately oriented towards the direction of the extreme value and cause less training shock. This effect can be seen in 3.1

2.3 Bimodal distribution removal (BDR)

There often will be some outliers in most datasets, which will slow down the convergence process, or worse, make the estimator converge to the incorrect solution. BDR[8] is designed to solve such problems. The BDR algorithm removed outliers in the training set by calculating the loss of each pattern. So here is the idea, after the training process has lasted for dozens of epochs, if the loss generated by each pattern is analyzed, it will show a bimodal distribution. The non-noisy patterns will concentrate in the smaller loss area, and the outliers will concentrate in the larger loss area. Once the outliers are isolated, they can be removed. In this section, the loss is calculated in the same way as in 2.2, which is Cross Entropy loss. Due to the small data set, only one step BDR was performed to prevent overfitting. The general BDR progress can be broken down into the following steps:

- Implement the training process with all training sets

- When the loss is reduced to a certain extent, calculate the corresponding loss of each pattern.
- Calculate the mean loss as M1
- Put those potential outliers whose loss is greater than M1 into a subset.
- Calculate the mean of subset as M2, the standard deviation as std2.
- Remove those patterns whose loss are greater than M2 + k*std2 (k =1)
- Repeat these steps until the new subset's standard deviation < 0.01

2.4 Evaluation method

In this paper, I use the simplest way to evaluate network performance: accuracy. Accuracy here equals correctly classified patterns divided by total patterns. The formula shows as below:

$$\text{accuracy} = \frac{\text{the number of correctly classified patterns}}{the\ number\ of\ total\ patterns} \qquad (3)$$

3 Result and Discussion

3.1 Improved ResNet without BDR

I trained it for 31 epoch, since batch = 64 and the number of training set is 819, there will be 14 steps for each epoch. Thus there will be 434 steps during the whole training process. figure 4 shows the training loss and validation loss during the training process.

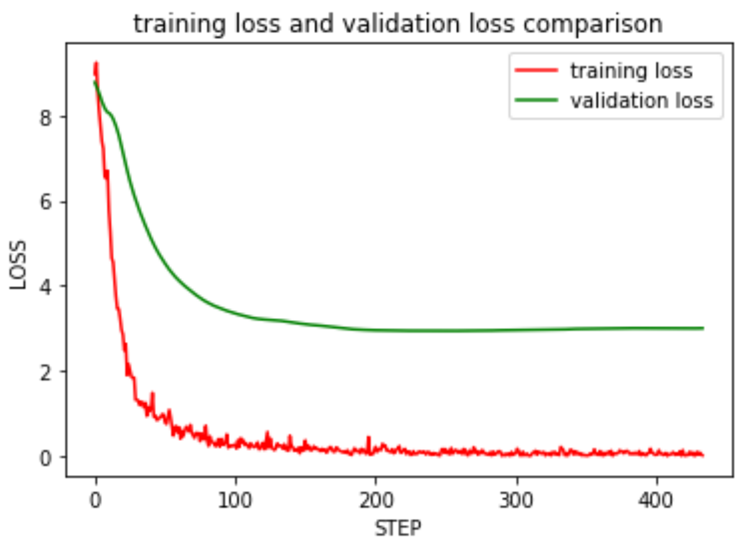


Fig. 5. Training loss and validation loss of ResNet

As can be seen in Fig.5, validation loss and training loss decrease sharply at the beginning. After 100 steps, both training loss and validation loss begin to converge slowly and the trend is alleviated. You may notice the slight fluctuation in training loss which is caused by dropout. There is a gap between training loss and validation loss, which may be caused by the inevitable gap between the training set and the verification set. The following Fig. 6 shows the accuracy of training set and validation set.

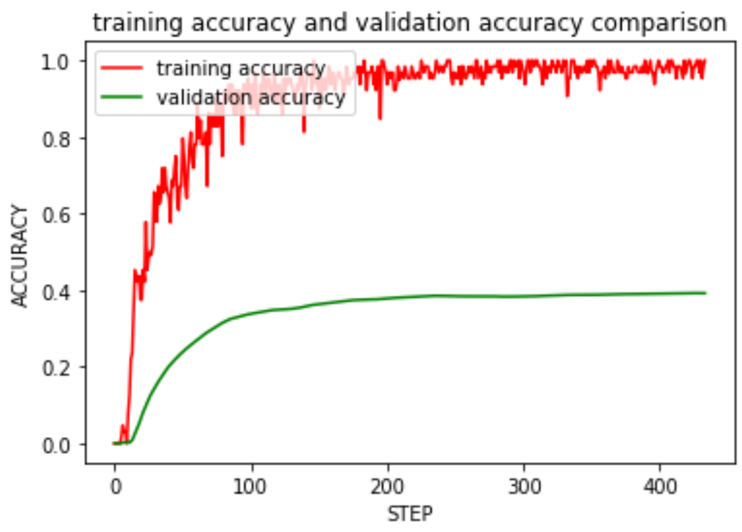


Fig. 6. The accuracy of training set and validation set of ResNet

As we can see from Fig.6, the accuracy of both the training set and validation set increase sharply in the first 100 step. Accuracy of the training set experience slight oscillations after 200 step, it reaches convergence and becomes stable at 100% after 200 step. As for the validation set, accuracy has been increased slowly since 100 step, it begins to converge to 38% around 350 step.

3.2 Improved ResNet with BDR

The figure below shows the variance distribution of loss for each pattern in each epoch, in order to determine the starting point of BDR.

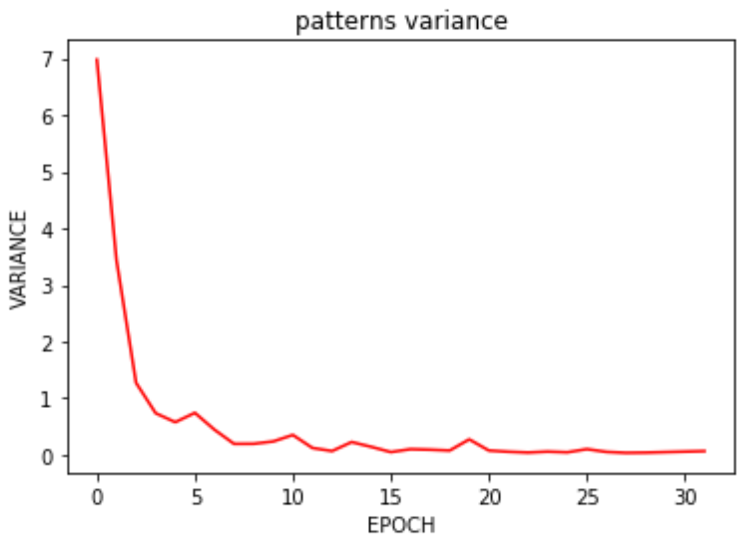


Fig. 7. The distribution of variance of pattern loss.

We can see that the variance began to stabilize after epoch 6, so I chose to start using BDR in epoch 6(step 14). The following figure shows the pattern’s loss distribution before BDR.

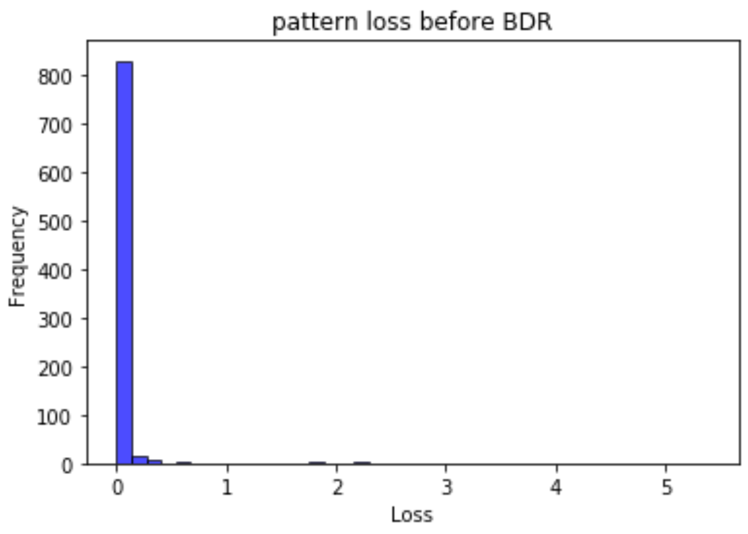


Fig. 8. Pattern loss distribution before BDR

Figure 8 shows that the pattern loss distribution occurs bimodal distribution. Few pattern losses are distributed on 3,4 and 5. The figure above is difficult to see that point due to the image proportion, but it does exist, otherwise, 3,4 and 5 wouldn't appear on the horizontal coordinate. It indicates that the SFEW data set does contain some noise. Therefore, the BDR algorithm will remove that part and continue the training process. Figure 9 shows the pattern loss distribution after BDR, we can see that pattern losses near 3,4,5 are dropped.

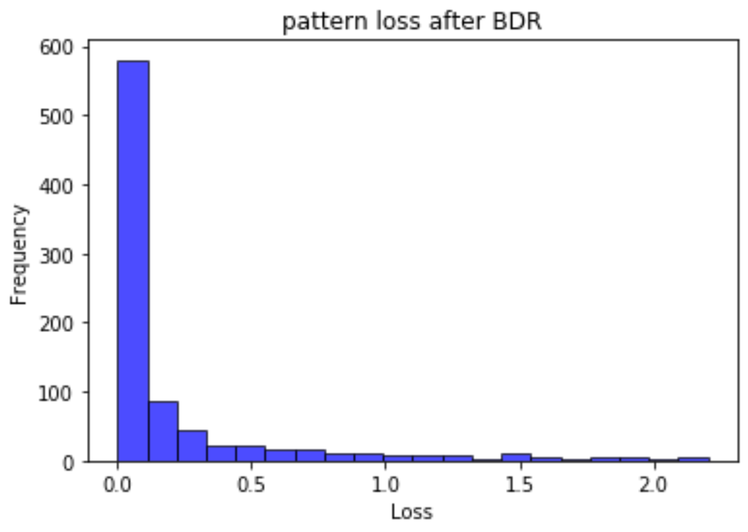


Fig. 9. Pattern loss distribution after BDR

With the BDR, the data volume of training set is reduced from 819 to 867. The following figure 9 shows the distribution of training loss and validation loss with BDR

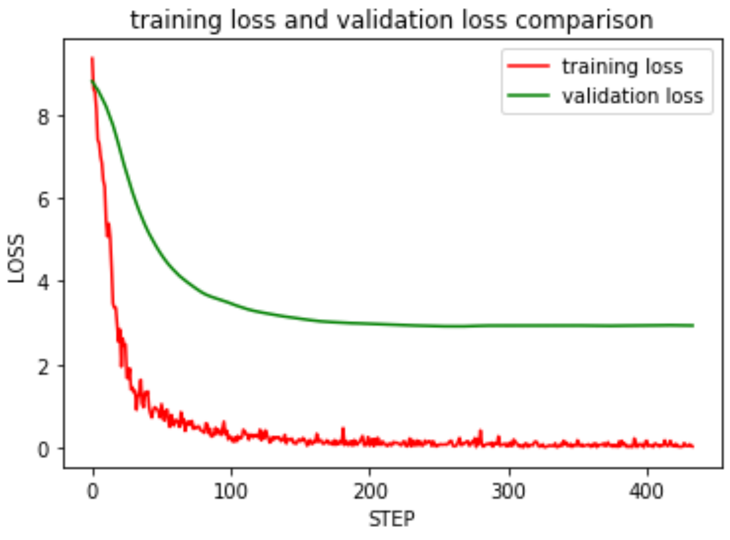


Fig. 10. Training loss and validation loss of ResNet with BDR

There is little effect on the training set and verification set. The below figure 11 shows the distribution of accuracy. The accuracy of validation improved by 1% which is 39%

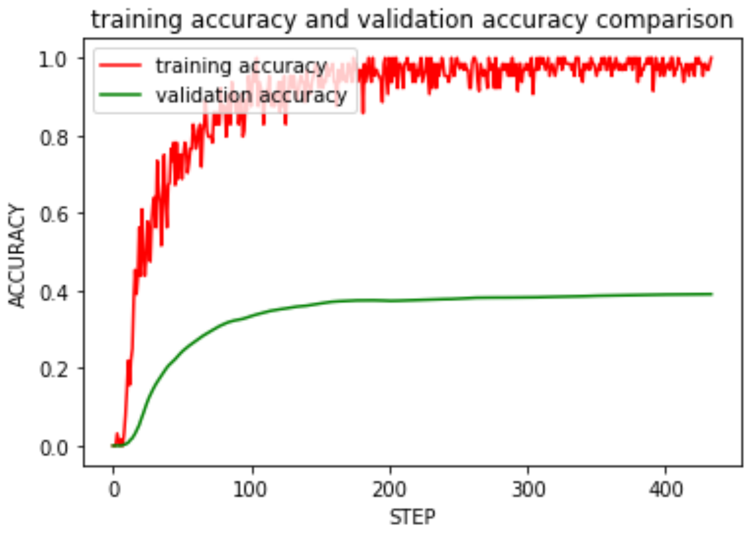


Fig. 11. The accuracy of training set and validation set of ResNet with BDR

The following table shows the results of the whole experiment. We can see that ResNet achieves a better performance compared with previous paper. With BDR, the accuracy of test set has been improved by 4% which is 46%

Dataset	Method	Accuracy
SFEW(image)	ResNet50 without BDR	42%
SFEW(image)	Resnet50 with BDR	46%

3.4 Comparison with other Approach

There are many other algorithms for facial expression recognition in current literature. For example, a non-neural network method uses SVM classification can achieve accuracy of 43.71% in LPQ features and 46.28% in PHOG features [9]. There is also a deep Convolutional Neural Network (CNN) architecture can achieve accuracy of 48.5% in validation sets and 55.6% in test sets [10].

As for my previous implementation, I used the simple neural network without BDR which achieved 25.2% accuracy in test sets, simple neural network with BDR which achieved 27.3% accuracy and neural network with BN & dropout which achieved 37.1% accuracy.

In this paper, the improved ResNet with BDR can achieve accuracy of 39% in validation sets and 46% in test sets. Overall speaking, the result is pretty promising. The detailed information can be seen in table 1 and Fig.12.

Table 1. Comparison with other approach

Dataset	Method	Accuracy
SFEW(image)	SVM,LPQ	43.71%
SFEW(image)	SVM,PHOG	46.28%
SFEW(LPQ feature and PHOG feature)	Simple neural network without BDR	25.2%
SFEW(LPQ feature and PHOG feature)	Simple neural network with the BDR	27.3%
SFEW(LPQ feature and PHOG feature)	Neural network with BN and dropout	37.1%
SFEW(image)	ResNet50 without BDR	42%
SFEW(image)	Resnet50 with BDR	46%
SFEW(image)	Deep Convolutional Neural Network	55.6%

4 Conclusion and Future Work

This paper uses two methods to accomplish faces emotion classification: improved ResNet50 without BDR, improved ResNet50 with BDR. Several data preprocessing operations are used in the paper, including data set generating, label processing, data augmentation, random sampling, etc. Results show that ResNet50 with BDR has the best performance, the BDR algorithm does improve the network by dropping some of the outliers in the training set. But even if without BDR, the ResNet50 can achieve good results compared with previous work. For future work, try to use other models like VGG and inception[17], and try to combine the advantages of several models for ensemble learning[18].

References

[1] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. Acted Facial Expressions in the Wild Database. In *Technical Report*, 2011.

[2] V. Ojansivu and J. Heikkil. Blur Insensitive Texture Classi- fication Using Local Phase Quantization. In Proceedings of the 3rd International Conference on Image and Signal Pro- cessing, ICISP'08, pages 236–243, 2008.

[3] A. Bosch, A. Zisserman, and X. Munoz. Representing Shape with a Spatial Pyramid Kernel. In Proceedings of the ACM International Conference on Image and Video Retrieval, CIVR '07, pages 401–408, 2007.

[4] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.

[5] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15(1):1929–1958, January 2014.

[6] L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621, 2017.

[7] Rubinstein, R.Y. and D.P. Kroese. (2004). The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning. Springer-Verlag, New York.

[8]. Slade, P., and Gedeon T.D.: Bimodal Distribution Removal, vol. 686 (1993)

[9] A. Dhall, et al., "Static Facial Expression Analysis in Tough Conditions: Data, Evaluation Protocol and Benchmark," Proc. IEEE Int'l Conf. Computer Vision Workshop (BeFIT), IEEE Press, 2011, pp. 2106–2112.

[10] H.-W. Ng, V. D. Nguyen, V. Vonikakis, S. Winkler, "Deep learning for emotion recognition on small datasets using transfer learning", Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, pp. 443-449, 2015.

[11] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015

[13] G. Klambauer, T. Unterthiner, A. Mayr, S. HochreiterSelf-normalizing neural networks
CoRR arXiv:1706.02515 (2017)

[14] Kingma, Diederik P and Ba, Jimmy Lei. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[15] A. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, A. A. Kalinin, *Albumentations: fast and flexible image augmentations*, 2018.

[16] H.-W. Ng, V. D. Nguyen, V. Vonikakis, S. Winkler, "Deep learning for emotion recognition on small datasets using transfer learning", *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pp. 443-449, 2015.

[17] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2015b. Rethinking the inception architecture for computer vision. arXiv preprint arXiv:1512.00567.

[18] Dietterich TG 2002 Ensemble Learning. The handbook of brain theory and neural networks Arbib M. A, (Ed.) Cambridge MA The MIT Press