Image-based Static Facial Expression Recognition using Bidirectional Neural Networks (BDNN), Deep Learning Convolutional Neural Network (CNN) and Transfer Learning using Pretrained ResNet Models

Feriati Nurdinasari[1]

[1] Research School of Computer Science, Australian National University
u7067895@anu.edu.au

Abstract. Facial expression detection using Static Facial Expression in the Wild (SFEW) close to the real-world dataset is still challenging. This research tried to use three different model architectures to classify human emotion from the SFEW dataset: Bidirectional Neural Network (BDNN), Convolutional Neural Network (CNN), and pre-trained Residual Network (ResNet) model. Based on our experiments, the pre-trained ResNet model outperforms the other two models and gives the highest 53.07% accuracy. We also discovered that the BDNN technique could give higher accuracy (27.61%) than the 19 % accuracy for the baseline non-linear SVM model using feature extraction LPQ and PHOG input data. This paper also found that weight decay can help prevent overfitting in the BDNN model, while data augmentation for pre-trained ResNet deep learning models can help avoid overfitting.

Keywords: Bidirectional Neural Network, Class Prototype, Static Facial Expression in the Wild, image classification, human emotion recognition, LPQ, PHOG, Deep Learning, Convolutional Neural Network (CNN), Residual Network, Transfer Learning.

1 Introduction

Recognizing human emotions is one of the most challenging tasks in human-computer interaction, which involves multidisciplinary fields such as computer vision, speech analysis, linguistics, cognitive psychology, robotics, and learning theory [1]. Using facial expression recognition, we can improve many real-world applications: detecting consumer perception of the product/service for marketing purposes and enhancing human-computer interaction in robotics. There are also many other practical applications in this field: affective computing, human behavior analysis, ambient environment, intelligent homes, pain monitoring in patients, stress, anxiety, depression analysis, lie detection, and medical conditions as autism [2].

In facial expression analysis research, realistic face data play a crucial role in enhancing the research in this area. Unfortunately, some of the early facial expression datasets were collected in a lab environment which is less representative of the real-world situation. Therefore, there is still a significant hurdle to do accurate facial expression recognition under an uncontrolled environment. In this research, we will use a static facial expression database named Static Facial Expressions in the Wild (SFEW), extracted from movies and closer to real-world data, to classify seven basic expressions/emotions (angry, disgust, fear, happy, sad, surprise, and neutral). SFEW dataset has 700 naturally occurring face images in many poses with different lightning and environment conditions, making it hard to classify compared to other constructed datasets like JAFFE and MULTI-PIE [2].

This paper experiment using different classification model and techniques to SFEW facial image data and examine if it can provide valuable insight into this challenging human emotion classification problem. In our first experiment, we train neural networks inspired by Bidirectional Neural Network and Class Prototype model (BDNN) [3] and inspect if the method could provide better accuracy than other techniques applied in the same database. For the BDNN technique, we used two feature extraction methods, Local Phase Quantization (LPQ) and Pyramid of Histogram of Oriented Gradients (PHOG), to convert SFEW images into descriptors as the model data input. The main idea behind the BDNN is to train neural networks to produce plausible input values given the output data. The model will be enabled to remember either input patterns or output vectors and can be used to construct more accurate rules from extracted results [3]. We can use the BDNN technique as associative memories and cluster center finder to enhance classification or prediction in real-world problems. Based on [3], the BDNNs have been used in two real-world datasets, Student Final Marks (SFM), to classify the marks, and Gross National Product (GNP) from UNCTAD to classify countries labels as Very Poor, Poor, Middle, and Rich Countries. A previous experiment with those two data shows that BDNN can provide sufficient capabilities of associative memories and cluster center finders.

Our subsequent experiment used a real image as an input for our SFEW image classification problem. We build a simple Convolutional Neural Network (CNN) to investigate whether the deep learning model could give good result performance using more complex image data. This simple CNN model will also used as a baseline model for other deep learning model in our experiment. We noticed that given the small SFEW dataset, the model with very deep layers can lead to overfitting and unable to generalized. To overcome this problem, we employed a transfer learning technique which gives promising result. We choose a sophisticated pre-trained model train on a similar but different dataset and used for a similar task with our research problem. We build a pre-trained ResNet18 and ResNet34 model, which already trained on the much bigger dataset ImageNet and also tackle the image classification problem [6]. Deeper neural networks are more challenging to

train because of the vanishing gradient problem, and ResNet architecture was designed to overcome this specific problem. ResNet model provides a residual learning framework to ease the training of the networks that are substantially very deep (used many layers). The model won 1st place on the ILSVRC 2015 classification task. Refer to the ResNet paper [6], there is empirical evidence showing that these residual networks are easier to optimize and gain better accuracy from considerably increased depth. ResNet explicitly reformulates the layers as learning residual functions regarding the layer inputs instead of learning unreferenced functions. With all the advantages and competitive results on the image classification task of the ResNet model, we want to examine if this pre-trained model can provide improved results to our human emotion classification task.

The rest of this paper's structure is as follow: the second part introduces the research methods concept, basic principle, and dataset; The third part introduces the experimental result and analysis. Finally, the fourth part summarizes and provides future work of this research.

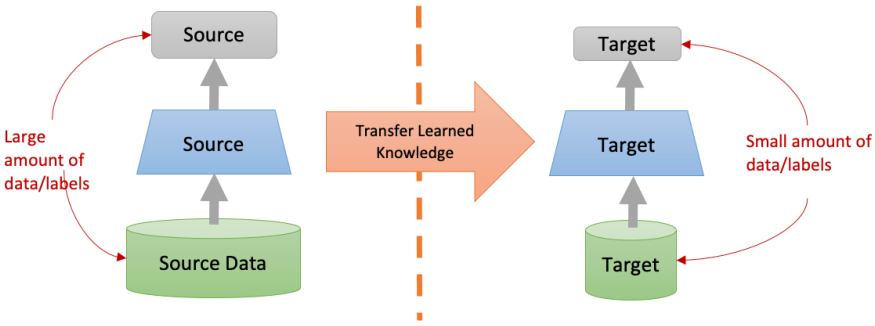

Fig. 0. Illustration of Transfer Learning

2 Method

2.1 Dataset

Facial expression analysis methods can be divided into image-based and video-based. Video-based approaches consider more robust since they can capture more human facial expression dynamic in nature. Still, there are scenarios when temporal data is not available, and we should use image-based facial expression analysis methods. Movies can provide more realistic close to natural environments than lab-recorded datasets. To address the issue of temporal and static facial expressions in though conditions, [2] propose two datasets extracted from movies as follow:

- Acted Facial Expressions in the Wild (AFEW)

AFEW is a data corpus consist of dynamic temporal facial expressions extracted from movies. The database contains 957 video clips labeled with six basic expressions: angry, disgust, fear, happy, sad, surprise, and neutral. It was collected based on Subtitles for Deaf and Hearing-impaired (SDH) and Closed Caption (CC) to search expression-related content. Data stored in an XML schema with the subjects in the clips annotated with attributes like Name, Age of Actor, Age of Character, Pose, Gender, Expression of Person, and the overall Clip Expression [2].

- Static Facial Expressions in the Wild (SFEW)

SFEW is a dataset which builds by selecting frames from AFEW to build static image-based facial expressions. The database covers unconstrained facial expressions, varied head poses, large age range, occlusions, varied focus, different resolution of the face, and close the sequence. The SFEW contains 700 images that have been labeled for seven basic classes - angry, disgust, fear, happy, neutral, sad, and surprise; six emotions and one neutral emotion (675 images used in this paper as 25 samples of the emotion' disgust' haven't been included). We used two versions of the SFEW dataset as follow:

a. The first version contains the first two principal components (each dimension 5) of the LPQ (Local Phase Quantization) and PHOG (Pyramid of Histogram Oriented Gradients) features of each image. Based on the dataset reference paper as a baseline, we used LPQ and PHOG descriptor in the strict person independent (SPI) protocol and combined those two sets for better performance. A principal component analysis is applied to reduce dataset input complexity, and 98% of the variance is kept [2]. The baseline classification accuracy calculated by averaging the accuracy for the two sets is 19.0% using a non-linear Support Vector Machine (SVM). We used this first version as an input for the BDNN technique.
b. The second version of the dataset contains the RGB images itself of the seven emotions. The width and height of each image are 720 and 576 pixels, respectively. We used this second version for Convolutional Neural Network (CNN) and pre-trained ResNet18 and ResNet34 models.

2.2 Data Preprocessing

- For version 1 of the dataset, the data was processed to cleaned missing(nan) values and outliers. We prepared the dataset into a training and testing set with 80% and 20% proportions. We used Z-score to find outliers using a filter for rows that have Z-score between -3 and 3. Then we implement standardization using each feature's mean and standard deviation in the training set, which also applied for the testing set.
- For version 2 of the dataset, the dataset was split into 80% training, and 20% testing sets proportion. The input images normalized in mini-batches of 3-channel RGB images of shape (3 x H x W), where H and W are set to 224. The images have to be loaded in to a range of [0, 1] and then normalized using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]. In order to prevent overfitting, we also applied data augmentation techniques in our experiments, including random resized cropped and flipping the images horizontally.

2.3 Model Design

We implement BDNN as Content Addressable Memory and BDNN as Cluster Center Finders model, which uses version 1 of the dataset. The general concept and main idea about these two models explain as follow:

a. BDNN as Content Addressable Memory

This approach applied the error back-propagation technique in both reverse and forward directions to adjust the weight matrix of the network. The neural network will be trained from left to right; the weights on the connections between the input layer and hidden layer and the weights between the hidden and output layers are used. The same weights between the hidden layer and output layer, an input layer, and a hidden layer are used for training in the reverse direction. When function relating inputs to outputs is not invertible (many-to-one inputs outputs relation), the paper suggests adding one extra node in the output or using statistical techniques manipulation data preparation. Our experiment assigns a random value to this extra node to map the input vectors to the new output vectors with one-to-one relation.

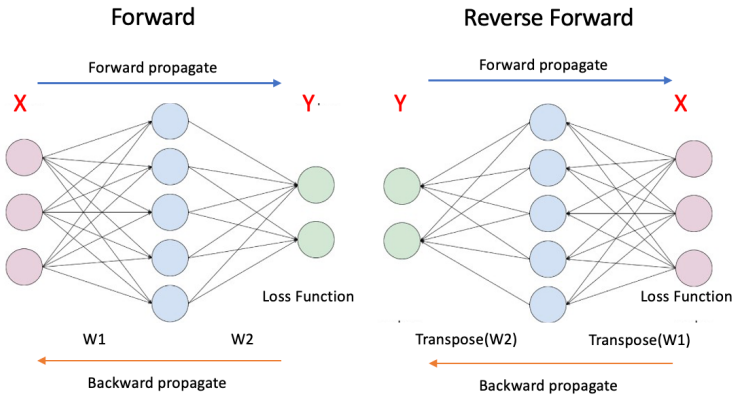

Fig. 1. Illustration of Bidirectional Neural Network

b. BDNN as Cluster Center Finders

According to [3], traditional neural networks trained by the back-propagation algorithm can't provide prototypes for the user, while BDNNs as cluster center finders give the advantage to provide such prototypes. These prototypes may then be used to enhance the learning and generalization ability of neural networks. In BDNN as Cluster Center Finders, we train BDNN with no limitation on the relation between input and output patterns. It will find class prototypes/cluster centers that represent the most expected input/feature value in the class. Then, we can use the cluster center to assign the most likely value when the input value is unknown and improve the learning process by explaining the properties of category classification. In this Cluster Center Finders architecture, we didn't use an additional node in the output of the forward pass.

The main architecture of these BDNNs has two hidden layers and uses different combination nodes for the experiment: 12/10 (first layer nodes/second layer nodes), 30/15, and 50/25. We use ReLU in each hidden layer for activation function, Softmax activation function in the output of forwarding pass, and linear function in the output of reverse forward (backward) pass. Mean Squared Error (MSE) and Cross-Entropy Loss were used as the loss metric and we applied Adam optimizer since they are the most widely used for classification problems. Furthermore, we simplify the methodology in [3] for bidirectional training and change the direction every 50 epochs. Finally, we evaluate the performance of training and testing data separately and using accuracy measure, which uses the formula below (tp is true positive, tn is true negative, fp is false positive, and fn is false negative).

$$Accuracy = \frac{tp+tn}{tp+fp+fn+tn} \quad (1)$$

For our deep learning models, we implement two different models for our version 2 real image dataset: a pre-trained ResNet18/ResNet34 model and a simple CNN as a baseline model. As we already mentioned in the first part, the main

problem in CNN could be the vanishing gradients or exploding gradient problem even though Batch Normalization ensures that the gradients have healthy norms. Thus, the deeper networks may fail to perform better than the shallow networks. The author of [6] proposes a solution to this problem in training deep networks using Residual Block. This Residual Block implements additions from the output of the previous layers (x) and the output of the next layers ($F(x)$). The illustration of the Residual Block is shown in Fig. 2.

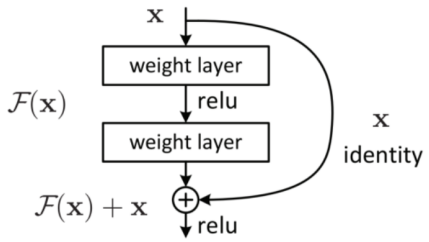

Fig. 2. Residual Block Illustration

The ResNet18/ResNet34 has been trained on the ImageNet dataset and provide by PyTorch for Transfer Learning purposes. ResNet18 has 18 layers while ResNet34 has 34 layers deep, and both take an image input of 224 by 224 (detail architecture example of ResNet18 in Fig. 3). For our baseline CNN model, we applied two 2D convolutional layers with the first convolutional layer applying 12 filters with 3 kernel size, 1 stride, and 1 padding. The second convolutional layer takes 12 input channels and generates 24 outputs, with a kernel size of 3, 1 stride, and 1 padding. We apply max pooling in the end with a kernel size of 2 and a drop layer which deletes 40% of the features to help prevent overfitting. The last layer is a fully connected layer with seven outputs. After each convolutional layer, we used the ReLU activation function and used SoftMax in the output layer. Our fine-tuning experiments used combination of different batch sizes (5 and 20), epoch (25 and 50), learning rate (0.01, 0.001, 0.005) and weight decay to find the best testing accuracy. Both CNN and the transfer learning ResNet models used Cross-Entropy Loss and Adam optimizer.

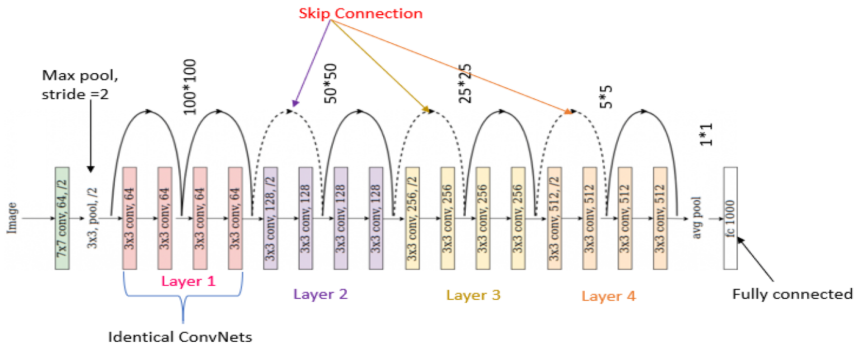

Fig. 3. ResNet18 Architecture

3 Results and Discussion

3.1 Comparison between BDNN models

For BDNN techniques, we experiment using a different combination of parameters in BDNN models as Cluster Center Finder and Content Addressable Memories. We also try two different loss functions, Mean Squared Error (MSE) and Cross-Entropy and implement weight decay to handle overfitting in some of the model experiments. In Table 1, we can see the overall performance of these models in different combinations of parameters; wd means weight decay, and h1 and h2 mean the first layer of hidden nodes and the second layer of hidden nodes as we use two layers neural network.

Table 1. Comparison of the accuracy (in %) with different hyperparameters between MSE and Cross Entropy Loss in BDNN Models

	12 h1 and 10 h2		30 h1 and 15 h2		50 h1 and 25 h2	
BDNN as Cluster Center Finders	**without wd**	**with wd**	**without wd**	**with wd**	**without wd**	**with wd**
	Training					
MSE Loss Function	41.93	22.05	54.92	30.91	73.82	35.04
Cross Entropy Loss	27.76	31.89	43.9	35.04	50.39	44.88
	Testing					
MSE Loss Function	28.36	14.18	26.87	23.13	31.34	23.88
Cross Entropy Loss	14.18	22.39	27.61	22.39	24.63	25.37
BDNN as Content Addressable Memories	**without wd**	**with wd**	**without wd**	**with wd**	**without wd**	**with wd**
	Training					
MSE Loss Function	35.24	23.23	37.8	27.17	50.39	27.36
Cross Entropy Loss	31.89	34.45	54.53	44.49	59.84	49.41
	Testing					
MSE Loss Function	25.37	20.15	23.13	22.39	25.37	23.13
Cross Entropy Loss	20.15	20.15	26.12	21.64	23.88	23.13

Based on the experiment result, BDNN models gives almost similar performance between MSE and Cross-Entropy loss function. The highest testing accuracy achieves by BDNN as Cluster Center Finders when using the medium hidden neuron (30 h1 and 15 h2) and Cross-Entropy Loss in 27.61%. Some of the experiments seem overfitting, so we implement weight decay in the optimizer. It successfully improves the model and lowers the overfitting problem. Compare with the non-linear SVM in the same data with an accuracy of 19% [2] as the baseline, these models still give better performance. One of the main reasons for the low accuracy, as stated by [2], is the complexity of the datasets that used the feature extraction LPQ and PHOG. The SFEW datasets prove to be more difficult to classify than other datasets as they capture real-world situations. By using a bidirectional neural network, we show improvement from baseline non-linear SVM. We also learn that the reverse forward mechanism in bidirectional models seems to reduce the overfitting in the model, especially in BDNN as content addressable memories that use extra node to make invertible function. For much simpler data like SFM and GNP data in [2], BDNN as a cluster center can provide the cluster center which can give more useful insight about the data as class prototypes.

3.2 Comparison between Simple CNN model and Pretrained ResNet models

We present our simple CNN model experiment in Table 2 which shows that simple CNN model gives good performance to classify human emotion class from real image input data. Our CNN model perform best in 50 epoch, with 0.001 learning rate , 0.001 weight decay and 20 batch size. The performance is not very different for smaller 20 epoch data, and we also found that 0.001 learning rate tend to make the model overfit. The model also performs well in 0.005 learning rate and achieve 28.15 % accuracy, and we can see that the model is slightly better and not overfit.

From the transfer learning model experiment in Table 3 and 4, we found that transfer learning technique using pre-trained model gives the highest accuracy in 53.07% for the ResNet18 model, using 25 epoch, 0.001 learning rate, 0.9 momentum, 0.001 weight decay, and 5 batch size. However, we also noticed oticed that these deep layers model seems to severely overfitting when we only applied minimal data preprocessing. We then experiment using data augmentation by randomly resized crop and horizontally flipping the input data, which we can see from the experiment significantly lowered the overfitting of the model and even improve the overall model performance. ResNet 34 model without data augmentation also still gives good performance in 51.85% using 25 epoch, 0.001 learning rate, 0.9 momentum, and 5 batch size. This pre-trained model gives better accuracy compared with our initial simple CNN model significantly and proof that transfer learning method is a potential option to train small dataset with deep learning model and still achieve good accuracy.

Based on our extensive experiments, we noticed several key findings in training our deep learning model. First, we found that a lower learning rate gives better performance to the pre-trained ResNet model and smaller batch size. Although we found that weight decay is useful for our BDNN models to prevent overfitting, the tricks didn't work in our deep learning model. In this experiment, we finally successful to reduce the overfitting by using the data augmentation. Moreover, we also found that the larger epoch doesn't always increase the model performance. Our experiment using 50 and 25 epoch results gives almost the same accuracy for the same hyperparameter settings for most of our deep learning model.

Table 2. Comparison of the accuracy with different hyperparameters in simple CNN

					Simple CNN	
No	**Epoch**	**Learning Rate**	**weight decay**	**batch size**	**training**	**testing**
1	50	0.005	0.001	20	87.03%	27.40%
2	50	0.001	0.001	20	98.51%	34.81%
3	20	0.005	0.001	20	50.19%	28.15%
4	20	0.001	0.001	20	96.85%	34.07%

Table 3. Comparison of the accuracy with different hyperparameters between ResNet18 and ResNet34 (**without data augmentation**)

					resnet18		resnet34	
No	Epoch	Learning Rate	weight decay	batch size	training	testing	training	testing
1	25	0.001	-	5	98.33%	49.63%	98.15%	48.89%
2	25	0.001	0.0001	5	97.96%	48.89%	99.07%	48.15%
3	25	0.001	0.001	5	98.15%	42.96%	99.44%	51.85%
4	25	0.005	-	5	57.59%	31.85%	20.19%	20.00%
5	25	0.005	0.001	5	22.59%	25.19%	25.56%	25.93%
6	50	0.005	-	5	36.11%	25.93%	26.85%	27.41%
7	25	0.01	-	5	16.11%	20.00%	20.19%	15.56%
8	25	0.001	-	20	100.00%	48.89%	100.00%	49.63%
9	25	0.005	-	20	100.00%	49.63%	100.00%	48.15%

Table 4. Comparison of the accuracy with different hyperparameters between ResNet18 and ResNet34 (**with data augmentation**)

					resnet18		resnet34	
No	Epoch	Learning Rate	weight decay	batch size	training	testing	training	testing
1	25	0.001	-	5	67.78%	53.07%	69.07%	45.19%
2	25	0.005	-	5	19.81%	23.70%	21.85%	25.19%
3	25	0.001	-	20	62.59%	44.44%	69.26%	48.89%
4	50	0.001	-	5	65.19%	46.67%	67.04%	45.93%
5	50	0.001	-	20	66.85%	45.19%	72.04%	47.41%

3.2 Comparison between BDNN and Deep Learning Model

From Table 2, 3, and 4, we can see that deep learning model performs well compare with BDNN model. The pretrained ResNet model gives significantly higher accuracy compare with the BDNN (almost two times higher). On the other hand, the BDNN model provide relatively simple architecture, which give more efficient training time in CPU infrastructure. In this experiment, we train our deep learning model mostly using GPU in large amount of time. This lower accuracy is actually acceptable given the more simplified input data from feature extraction and also BDNN simpler architecture. If we have more time and better infrastructure, then the CNN and pretrained transfer learning model will gives better result, especially in our deignated task to classify human emotion using closer to real-world data SFEW. From our experiment, we also found that using weight decay can prevent overfitting in BDNN model, while in deep learning model we can employ data augmentation.

4 Future Work and Discussion

This paper shows that human emotion classification problems using close to real-world SFEW datasets can be improved using different techniques. For BDNN inspired model as the associative memories and cluster center finders, we can improve model performance using LPQ and PHOG feature extraction input from the baseline 19% accuracy using non-linear SVM technique [2]. We found that BDNN as cluster center finders gives better overall performance and achieves the highest 27.61% accuracy.

We also used the real image version dataset and experimented with deep learning models. We experimented using two pre-trained ResNet18 and ResNet 34 models and built a simple CNN model as a baseline. In our experiment, pre-trained model ResNet18 and ResNet34 significantly improve the model accuracy. Moreover, using the deep learning model, we found that lower learning rates perform better in more complex models and help the network to converge. The smaller batch size in the ResNet model seems to also give a better result than the larger batch, and the larger epoch doesn't always give significantly better results in this model. We try to use weight decay to prevent overfitting in deep learning model, which didn't give good results. The overfitting problem of this model can be prevented by using data augmentation in the data preprocessing step.

We suggest some of the future works and improvements for this research by using optimization techniques to find the best hyperparameter settings and improve each model's performance. Evolutionary algorithms such as the Genetic Algorithm can be one of the potential alternatives to this optimization enhancement. Secondly, for BDNN models, we can experiment using other feature extraction techniques for the SFEW dataset. In [2], it is proven that LPQ and PHOG methods are clearly not suitable for facial expression analysis in an uncontrolled environment like SFEW data. We can explore different feature extraction for this dataset that more suitable and try to use the bidirectional model to see if it can give higher performance. Although we already achieve good result using ResNet pretrained model with ImageNet, we can experiment in the future using the same model trained with larger facial emotion recognition data like FER-2013 dataset to better improve the pretrained model for the image classification problem on human emotion recognition task. Finally, we also can used the face detection technique for data preparation which could be significantly improved the model performance given the SFEW data sometimes contains lot of noisy that give more difficulties for training process.

References

1. Yu Z, Zhang C.: Image Based Static Facial Expression Recognition with Multiple Deep Network Learning. Proceedings of the 2015 ACM on International Conference on Multimodal Interaction (2015). https://doi.org/10.1145/2818346.2830595
2. Dhall A, Goecke R, Lucey S, Gedeon T.: Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark. Proc IEEE Int Conf Comput Vis 2106–2112 (2011). https://doi.org/10.1109/ICCVW.2011.6130508
3. Nejad AF, Gedeon TD.: Bidirectional neural networks and class prototypes. IEEE Int Conf Neural Networks - Conf Proc 3:1322–1327 (1995). https://doi.org/10.1109/icnn.1995.487348
4. Dhall A, Asthana A, Goecke R, Gedeon T.: Emotion recognition using PHOG and LPQ features. 2011 IEEE Int Conf Autom Face Gesture Recognit Work FG 2011 878–883 (2011). https://doi.org/10.1109/FG.2011.5771366
5. Pontes-Filho S, Liwicki M.: Bidirectional Learning for Robust Neural Networks. Proc Int Jt Conf Neural Networks 2019-July: (2019). https://doi.org/10.1109/IJCNN.2019.8852120
6. He, K. Zhang, X. Ren, S. Sun, Jian.: Deep Residual Learning for Image Recognition. 2016 IEEE Conf on Comput Vis Pat Recog, pp. 770--778. Las Vegas (2016)
7. J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” *Adv. Neural Inf. Process. Syst.*, vol. 4, no. January, pp. 3320–3328, 2014.