Bottleneck Feature Extraction-Based Deep Neural Network Model for Facial Emotion Recognition

Tian Ma[1(✉)], Kavuma Benon[1], Bamweyana Arnold[1], Keping Yu[2,3], Yan Yang[1], Qiaozhi Hua[4], Zheng Wen[5], and Anup Kumar Paul[6]

[1] College of Computer Science and Technology, Xi'an University of Science and Technology, Xi'an 710054, China
matian@xust.edu.cn
[2] Global Information and Telecommunication Institute, Waseda University, Shinjuku, Tokyo 169-8050, Japan
[3] Shenzhen Boyi Technology Company Ltd., Shenzhen 518125, China
[4] Computer School, Hubei University of Arts and Science, Xiangyang 441000, China
[5] School of Fundamental Science and Engineering, Waseda University, Tokyo 169-8050, Japan
[6] Department of Electronics and Communications Engineering, East West University, Dhaka 1212, Bangladesh

Abstract. Deep learning is one of the most effective and efficient methods for facial emotion recognition, but it still encounters stability and infinite feasibility problems for faces of different races. To address this issue, we proposed a novel bottleneck feature extraction (BFE) method based on the deep neural network (DNN) model for facial emotion recognition. First, we used the Haar cascade classifier with a randomly generated mask to extract the face and remove the background from the image. Second, we removed the last output layer of the VGG16 transfer learning model, which was applied only for bottleneck feature extraction. Third, we designed a DNN model with five dense layers for feature training and used the famous Cohn-Kanade dataset for model training. Finally, we compared the proposed model with the K-nearest neighbor and logistic regression models on the same dataset. The experimental results showed that our model was more stable and could achieve a higher accuracy and F-measure, up to 98.59%, than other methods.

Keywords: Emotion recognition · Deep neural network · K-nearest neighbor · Haar features

1 Introduction

Image detection is increasingly and widely applied for different purposes, such as expert evidence, plant identification, tumor recognition, and facial recognition, among others. Due to the new massive rollouts of 5G internet services around the world and the increase

S. W. Loke et al. (Eds.): MONAMI 2020, LNICST 338, pp. 30–46, 2020.
https://doi.org/10.1007/978-3-030-64002-6_3

in demand from previous versions of the internet, we expect to see a steep mobile application usage boom in the global markets, which could generate mobile data traffic and make it possible to mine mobile intelligence [1]. Images are currently becoming one of the most important forms of mobile data. We could do more with such free data, which are generated through the activities in which users engage. In this paper, we focus on facial emotion (FE) recognition in mobile images [2] of different races. As a special field, FE recognition has attracted the attention of several research teams, and many related methods have been proposed.

For example, as suggested by SL Happy [3], salient patches that had discriminative patches were used in each classification of one pair of facial expressions. Similarly, we also used the CK+ data set, but the difference is that our group decided to use the Haar cascade method with randomly generated masks that we matched with the scores of the image. At times, we could also detect the same image via image edge and edge enhancement, as referenced by Xin Wang [5]. Unlike the facial patches method of S L Happy, B. Ryu [6] used a local directional ternary pattern method that has selected connections to edge-based methods in smooth regions of an image. However, if we detect the image, the stability is important in such a procedure [6]. With this in mind, we attempted to refer to a gridding-based algorithm for stability and endeavored to solve infinite feasibility problems, as that histogram-based algorithm uses grid like features and stability levels using that algorithm. And this would make them feasible because we consider infinite pixels instead of one pixel [5]. To obtain the extracted object, even with many different objects of all shapes, color, and posture, we suggested the K-nearest neighbor (KNN) method. However, their emotion recognition (ER) is better, which we only realized by considering the key points. Certain expressions have relative similarity, but the emotion might be different depending on the person. For example, a person with large cheeks might have a similar facial expression while crying, depending on the moment of the image capture. Because facial patches read only the bloated cheeks, this is where we have a different suggestion. With all of these differences, the need exists to find the best parameters or points to be used, create a group of combined subsets with the extracted components, and not use all of the points because it could affect the time computation of the suggested algorithm.

Certain great methods in machine learning have been used to address problems such as detection of facial emotions. In the field of image recognition, the deep neural network (DNN) has become a powerful tool [22] and has also initiated a large research base that attracts many participants. For example, the deep CNN network and new modern advanced techniques have been used in advanced image feature detection developments. From the VGG16 experimental results, it was found that together with continuous deep learning of the convolutional layer learning, the feed-forward neural network that uses the error backpropagation algorithm could learn more object features. These convoluted characteristics gradually become coarser in the process of model training, and the inadequacy of VGG16 in learning convergence occurs because the degree of convergence is weak. In other words, the phenomenon of gradient disappearance is obvious in the latter layers. The feature data of certain convolutional layers do not reflect the extraction of edge detailed features. The final output port directly uses the featured data of the fully connected layer, which is overly dependent on the extraction of edge detailed features.

Certain of those great methods include transfer learning with VGG16 for extraction of features by relying on a 3 × 3 layer filter with a stride. However, in obtaining precise results, because we are using supervised learning methods, we already know the expected results.

To obtained the desired results, in this paper, we used two baseline methods: KNN and the logistic regression method [7, 8]. In the third stage, we took the input data for the DNN, which were the Haar features that we extracted from the convolutional neural network, and subsequently came up with the results. We used all of these steps to improve the algorithmic time and accuracy. The entire process is a combination of traditional methods for detection and extraction that we used together to obtain the final results.

2 Related Work

For a long time, facial detection has gathered high interest from the research community. The major goals have been accuracy, speed in execution, and the amount of resources consumed in executing the task. All of these mentioned factors rely on the training set. The goal is to obtain an algorithm that aligns well with the training set and gives us the desired results accurately and in time, without consuming most of the hardware resources. It has been estimated that by 2024, 90% of the world's smartphones will be using biometric facial recognition features [9]. This trend is already emerging with the key players of private companies and governmental bodies. This effort has resulted in use of biometric technology. It should be noted that biometric facial recognition must be reinforced through both hardware and software. Via activation of sensors [9, 10], biometric facial recognition has different categories, as specified by Zualkernan and Aloul [11], and the main categories are verification and identification. Considering the importance of facial landmark detection, facial expression performance is attracting increasing attention in the field of image classification. The alignment of the face for detecting the position of the eyes has become an important issue, usually performed by horizontal positioning. For facial expression recognition, extraction of features from the face is followed by landmark detection, and feature selection plays an important role in prediction because it directly influences the model accuracy. The precise components of facial tracking were used in infrared illumination in which Kalman filters were applied. By observing the expression changes, Uddin reported a good performance [20]. In reference [21], a relative geometrical separation-based method was portrayed, which used computationally costly Gabor channels for landmark location and following. That method also used consolidated SVM and HMM models as classifiers. Different methods have been suggested such as the Haar classifier method, adaptive skin color method, and many more [12]. Facial recognition has become a fast-growing technology adapted by many governments and companies for different purposes. For example, identification is used by companies such as Alibaba and Facebook and by police departments in China, USA and Europe. Verification on the other side has been rolled out by certain governments in their different endeavors to collect data on their respective populations. However, each of those categories must fulfill the purposes defined as follows.

To explain this issue in our context, we need know the determinants or the parameters to apply for this matter. These parameters include the background of the picture, the

clarity of the picture, the position of the Haar-like features, and others. The contribution of the Voila-Jones algorithm framework face detection method obtained its peak point because of the speed of the implementation of Haar features. It is observed that the Haar classifier works quite impressively if the images contain a simple background. For handling large databases such as CK+, the Haar cascade shows its best detection accuracy. Images in which the target object is wearing glasses do not affect the Haar classifier accuracy. Even if images are confined with illumination, this method still works quite impressively. However, in our case, we had to compare with the existing trained dataset. In such a case, we used the linear regression method, which was the picture we input versus the trained dataset. Although certain shortcomings exist in this method, it gives values larger than 1 and less than zero (0), which are outside the bounds of the Haar classifier if we do not consider the gray areas. Therefore, we pivoted to logistic regression, which is similar to linear regression because the aim is to estimate the values of the parameter coefficients. We determined the algorithm that could show the connection between the no input value and the output value; it is the opposite of a linear function if we consider values that are non-linear. This method generally takes any given number and maps it between 0–1 without necessarily exceeding the limit, thus solving the problem mentioned with linear regression. We plotted the 2D graph using the XY plane and plotted the winners and the losers in terms of points. The losers are the ones tending to zero (0), and the winners are those tending to one (1), and we draw the line between them. From the plotted points, we can estimate whether the input picture matches the emotion that we want. With a given form of gradient descent characteristics that we noted with logistic regression, we are satisfied that the estimation can be used as a determinant for our research purpose. The number of iterations were few in our case, which could affect the algorithmic time consumption such that the desired results might not be achieved. This method was designed to use all of the data and compare with trained data by identifying which points are closer to the others. Whenever we need to find the predicted or chosen point (K), in our case, we already have a trained set of data CK+, and we need to import new images in the algorithm. Our method maps the dataset with the new image data and gives us the desired results. Our design was based on the emotions that we selected, which were sad, happy, angry, surprise, disgust and neutral. When we input the picture, it could predict the result although the result accuracy was slightly lacking. Using the Euclidean method of distance, we might apply both linear and nonlinear operations, and thus we chose to use logistic regression [8], which is applied by supplying the parameters of the logistics model, which is generally a binary model.

3 Proposed BFE Model

The name ‘Bottleneck’ comes from the fact that we use hidden multi-layer perceptions. The overview of the BFE model is shown in Fig. 1. Observation suggests that improving the feature extraction method can enhance the performance of the model, meaning that the first step was to feed the input images into the BFE model for extraction of the faces by the Haar classifier. The Haar classifier uses the AdaBoost and Viola-Jones detection algorithms to detect the important features of the faces by removing unnecessary components of the image. The extracted faces were fed into the VGG16 transfer learning model

for bottleneck features. The implantation was performed by removing the last layer of the VGG16 model. After extracting the features from the BFF model, a deep neural network with five dense layers was implemented for training of the model. Finally, the model was tested with random images downloaded from the internet.

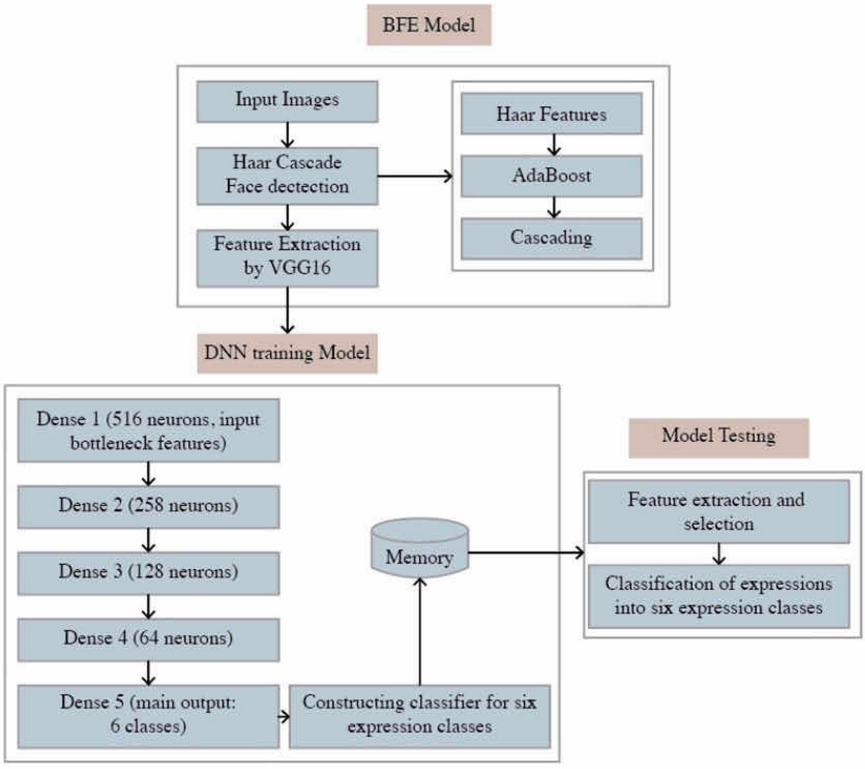

Fig. 1. Overview of the proposed BFE model.

3.1 Haar Cascade Face Extraction

The Haar cascade classifier is based on the ability to analyze pixels in the image into squares by a function. This method uses "integral image" concepts to group the different portions of the image and map the "features" detected. This method also uses the AdaBoost learning algorithm, which chooses a small number of useful features from a large set that have been stored to supply us with valid results for the classifier. The method also uses cascading techniques to detect a face in an image. The Haar cascade classifier is based on the Viola-Jones detection algorithm, which is already trained and has a wide range of faces and non-face components that were previously analyzed in its construction, can be easily accessed, and have a significant comparison with any image that we want to classify.

Haar Features

Haar features are refer to pixels, which in this case are in the form of 0 s and 1 s can also take on the value of −1 if we consider the gray areas [13]. Because we are interested in extracting the face, we apply these pixels in a piece-by-piece manner on the different components of the face, which we refer to as the ROI (region of interest) and is performed by moving them. And then move the pixel again. In this case, we extract the important features and remove the unimportant ones using a 24 × 24 window. As shown in Fig. 2, certain Haar representation are based on their shapes and values, where 0 s represent the white regions and 1 s represent the black shaded region.

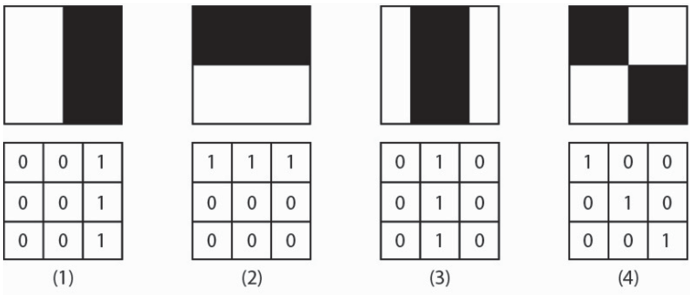

Fig. 2. Haar representations as shapes and numbers.

Although the number of pixels is usually large, we attempt to consider methods such as integral images [14], which reduce the pixels significantly and are applied on the next layer of analysis later.

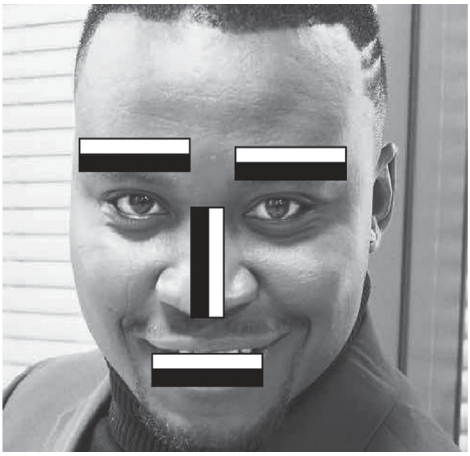

Fig. 3. Haar application on a face.

When we use the Haar algorithm, the windows repeatedly take different shapes of the 0 s and 1 s as they move towards the other pixels. In this case, we used an algorithm

that also relies on the integral image in computer vision, which has been suggested by Jones and Viola [1, 15], in which we have had to reduce the pixels significantly but with maximum effect. We applied this approach to differentiate the cat face from the person's face, even when we place them together in the same picture. The major advantage of this method as a base method for any facial recognition algorithm is that it is not affected by certain small factors such as the age of a person or the background behind a person. The Haar features are the eyes, nose, and mouth, as shown in Fig. 3, which exclude people wearing masks and images with background. This method is one of the pioneering methods used in facial detection because it is quick, and the processes it uses to classify the image reduce the algorithmic time. The applications can be found in common technology gadgets such cameras because it is widely used in almost all new facial-recognition phone cameras.

AdaBoost

As discussed above for the Haar features of 24 × 24 base resolution windows, there are approximately 160,000 features that need to be calculated. Additionally, a few features might be useful for more facial recognition systems. Therefore, such notably large features are needed for training, and use of all features can lead to poor results and high GPU and processing times. To remove these unnecessary features, AdaBoost was selected to choose the important relevant features that are needed and used in training. AdaBoost is used to select the best features generated by the Haar method [23]. When the relevant features are found, a weighted combination of all of these features is put to use in assessing and deciding any given window and whether it contains a face or not. Every relevant feature selected that was named as a weak classifier by AdaBoost is treated as fine if it can perform better than random guessing [24]. For detection of a face segment, each of these weak classifiers is relevant. The weak classifier output is binary if the classifier can detect a portion of the face or not.

Cascading

By breaking down a large problem into small and easily manageable modules, the Viola-Jonas algorithm is a famous algorithm that has long been in use and aids in the cascade for the Haar classifier. Boosting calculation requires strong classifiers to make the strategy function well, but for certain situations, we might encounter weak learners, which we can consolidate, and this situation encourages us to characterize that the closer the output, the closer the value to 1. When we used the Haar classifier for almost all cases, we removed the background of the images because we are most concerned with the human faces, especially for emotion recognition. For a 24 × 24 image window, 2,500 relevant features are needed and are processed by AdaBoost. AdaBoost aids in the process of obtaining these important 2,500 features for all 160,000 features. In this method for an input image, the fixed 24 × 24 windows must move all over the image to obtain 2,500 features for each partial window. A simple linear combination of all outputs is built and used to check whether the thresholds of the outputs are inside or outside of a limit. The Viola-Jones method follows a simple principle of scanning the detector through the similar image, and the detector scans with a new size every time. Although an image contains one or more faces, it is obvious that an excessively large amount of the scanned subwindows will still be negatives (non-faces). Therefore, the algorithm should focus

more heavily on discarding non-faces quickly and spend additional on time on probable face regions. Hence, a single strong classifier formed out of the linear combination of all best features is not satisfactory for evaluating each window because of computation costs. Instead of calculating 2,500 features for each single window, we use the idea of cascading and perform a sampling of 2,500 features into x different cascades. In this manner, we can linearly detect whether a face appears or not in different cascades. If a single cascade finds a face in an image, then the image is passed on to the next cascade. If no face is found in a cascade, we can move to the next window after dropping that window. This process reduces the time complexity. The job of each stage is to determine whether a given subwindow definitely contains or does not contain a face. The Viola-Jones face detection algorithm is trained, and the weights are stored on the disk. The next step is to take the features from the file and apply them to our image. If a face is detected, we obtain its corresponding location.

3.2 Feature Extraction

In our quest to recognize the face, the current method available is the Siamese network, which can sometimes translate to "connected". In this case, we already have a trained set that is relatively small, and thus the computation time is not as high, and all we have to do is to compare the input with the existing trained set. We must take the embedding of each image and put it through a neural network to obviously compare their Euclidean distance. This is an improvement over the standard CNN because it requires many images to be trained in the training set. However, on the positive side, certain existing datasets have been supplied [3, 4] to cater to such issues, although they might not be as large as those that are secretively used by large companies in the game. The algorithm requires us to enter two images and compare them by obtaining the sigmoid [8], which can be translated to the similarity between those two images. However, in certain cases, we can pivot to the triple loss [16], where we can input two images that are slightly similar and attempt to find the difference between the three images by first comparing the almost similar images. For example, image A, B and C are the chosen images. We consider A as an anchor image, B as a positive image, and C as a negative image. By applying the Euclidean distance, we can compare the anchor image and the positive image and subsequently compare the anchor and the negative image. Our aim is to plot these images and retain a secret number, or what we would consider a fingerprint, for the collection of images that have been well compared. The major disadvantage is that if we are given a notably large dataset, we might run into problems with computational time. If we need to train 3 images per time, then if we have a 240 million image dataset, that process can take a long time. For this reason, Google introduced an algorithm that can maximize the positive images and minimize the negative images in the process of comparing them to the anchor image. Given the base formula for triple loss d(a, p) and d(a, n), where a is the anchor image or what we consider as the true image on which we base our judgement, and P and n represent positive and negative, respectively, if we rely only on this algorithm, we could find ourselves in more trouble if we need to compute a larger dataset. The solution is to maximize the positives and minimize the negatives.

After extracting the face via Haar cascading, we need to extract the features. To find the bottleneck features, we opted to choose the transfer learning mechanism because

transfer learning obtains a better accuracy in taking on highly trained and weighted neural networks. We used VGG16 in Fig. 4 as the transfer learning mechanism, which was discovered in the ImageNet competition in 2015. The input given to the first convolution layer (conv1) is fixed to a 224 × 224 RGB image size. The image passes through a stack of convolutional (conv.) layers, where the filters were used with a small receptive field of 3 × 3 (which is the smallest size used to capture the notion of left/right, up/down, center). One of the configurations also uses selected 1 × 1 convolution filter, which can be viewed as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel. The spatial padding of each conv. layer input is set such that the spatial resolution is preserved after an assessment, i.e., the padding is 1 pixel for 3 × 3 conv. layers. Spatial pooling is applied by five max-pooling layers, which follow certain of the convolution layers (not all of the conv. layers are followed by max pooling). Max pooling is performed over a 2 × 2 pixel window with stride 2. Three fully connected (FC) layers follow a stack of convolutional layers (which have a different depth in architecture). It should be noted that we didn't remove any FC8 layers. The model just removes the top layer of the VGG16, as it is designed for 16 outcomes and the model has 6 outcomes. The first two layers have approximately 4096 channels each, and the third layer performs approximately1000-way ILSVRC classification, meaning that it contains a staggering 1000 channels (per class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks. All hidden layers are equipped with rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain local response normalization (LRN). Such normalization does not improve the performance on the ILSVRC dataset, but it can result in a large amount of memory consumption and additional computing time.

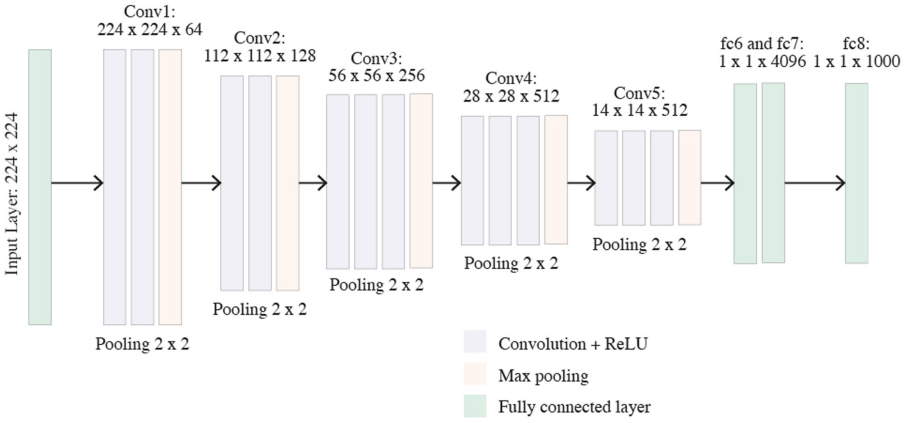

Fig. 4. VGG16 CNN transfer learning model.

A convolution neural net (CNN) was used in feature extraction from the face, and VGG16 was used as a CNN architecture. This method was used to win the ILSVR (ImageNet) competition in 2014 and is considered to be one of the most effective and efficient vision model architectures to date. The most unique characteristic of VGG16 is

that instead of using a large number of hyperparameters, it focuses on more convolution layers of a 3 × 3 filter with a stride 1, supplementing it with the same padding and max-pool layer of a 2 × 2 filter of stride 2. This method follows this style of convolution and max-pool layers consistently throughout the entire architecture. In the end, this method has 2 fully connected layers followed by a softmax layer for output. The '16' in the VGG16 refers to its 16 layers that have weights. This network is quite large and has approximately 138 million parameters. We took advantage of the VGG16 model for our facial recognition model by removing the last output layer of the model and replacing it with our model. Before feeding the training data into the VGG16 model for the bottleneck feature, we generated more image data extracted from Haar cascading by the Keras image data generator, which increases the size of the images for better accuracy with little variation. The generating systems worked by flipping the images horizontally and vertically or by applying variations in the brightness of the image. We used batch size 1, and after predicting the bottleneck feature, we reshaped the trained data into 568 × 51200 and saved the data features and labels as NumPy array files for training purposes.

4 DNN Training Model

After we obtained the features in Sect. 3, we implemented these extracted features into our deep neural network (DNN) model, which has five dense layers. The basic knowledge of DNN suggests that many dense layers should be present between the input and output [18, 19] to achieve a better-trained model. The following Table 1 shows the mathematical framework of the DNN model, where x represents the input bottleneck features extracted from the VGG16, and y denotes the emotion labels. In line 4, Ψi defines the mapping function of Di layers, and Wi indicates the random weights generated by the Keras framework library.

In line 3, L represents the layer number (for this model, L = 5), and Φ (z_i) represents as the activation functions, where z_i obtain the activation (in our case, for the first four layers, we used "ReLU" activation, and in the last layer, we used "softmax" activation). In the first dense layer, we feed every bottleneck feature of each image into 512 neurons. We used dropout as a type of regularization that randomly eliminates certain units and their connections during training, with the intention of reducing the degree to which hidden units co-adapt, thus combatting overfitting. We feed the output of the first dense layer into three consecutive dense layers with 256, 128, and 64 output neurons. In the last layer, we used the "softmax" activation function, which turns the output of the fourth dense layer logits into probability distributions of a list of potential six outcomes. When the model outcomes are greater than two, we used categorical cross-entropy and the popular deep learning "adam" optimizer during the compilation process. In the fitting of trained data with labels, we used 40 number epochs with 100 batch size and saved the training model and weights in hierarchical data format files.

Table 1. Algorithm table for DNN model.

Algorithm: DNN model	
1:	**function** *DNN(x, y)*
2:	$x_1 \leftarrow x$
3:	**for** $i \in \{1, \ldots, L\}$ **do**
4:	$z_i \leftarrow \Psi_i (D_i (W_i, x_i))$
5:	$x_{i+l} \leftarrow \Phi(z_i)$
6:	**for** $i \in \{L, \ldots, 1\}$ **do**
7:	$W_i \leftarrow W_i$
8:	**if** $i = L$ **and** type = *classification* **then**
9:	$e_L \leftarrow x_{L+1} - y$
10:	$W_i \leftarrow W_i - \eta \nabla w_i J(x, y)$
11:	**return** y

5 Experimental Results and Analysis

To validate the proposed methodology, we referred to the Cohn-Kanade (CK) dataset. In the following subsections, we briefly describe the database and present the experimental results and discussion. In Sect. 5.1 we discuss the dataset and details of the images used and generated, and in Sect. 5.2, we show the results of the model and discuss details of the results found.

5.1 Dataset

In our experiment, we used 568 frontal face images of six different emotions, which are converted into grayscale and have a resolution of 350 × 350. Certain original images were printed, scanned, and digitized. We used the CK+ dataset for training, which was created using the faces of 100 university-level students [25]. This dataset consists of students ranging in age from 18 to 30 years in which 65% were female, 15% were African-American, and 3% were Asian or Latino. As shown in Fig. 5, the trained images with six emotions, i.e., anger, disgust, happy, neutral, sadness, and surprise, are shown from the top to the bottom of the figure. It should be however noted that we didn't use fear in Paul Ekman's 6 basic emotions. As we replaced it with neutral and the reason for that is when we have surprise and fear in images, it is hard to differentiate them. And this is because the reaction on the face is somewhat the same so we wanted to give a broader perspective by introducing a new facial emotion category where each row of the figure represents an emotion. For our study, we used the Haar classifier to extract the faces from these images and cropped the faces to the data directory files. The cropped dataset is saved in the training data directory. During VGG16 feature extraction, we generated more images to increase the data because it yields good accuracy by flipping the images horizontally and zooming in and out by the Keras image generator library.

Fig. 5. Trained images sample.

5.2 Experimental Results and Discussion

In this stage, we compared our BFE-based deep neural network (DNN) with two baseline methods, i.e., logistic regression and K-nearest neighbors (KNN) [17]. The accuracy and F-measure of the BFE-based deep neural network (DNN) model are quite impressive.

Table 2. Accuracy comparison.

Model	Accuracy	F-measure
BFE-based DNN model	98.59%	98.59%
K-nearest neighbors	*83.09%*	80.57%
Logistic regression	89.88%	89.81%

The accuracy and F-measure comparison among the models are shown in Table 2. From this table, it can be easily observed that our BFE-based DNN model produced satisfactory predictions over the two benchmarking models.

Table 3. Confusion matrix for DNN model.

Predicted Class / Original Classes	**Neutral**	**Anger**	**Disgust**	**Happy**	**Sadness**	**Surprise**
Neutral	100	0.0	0.0	0.0	0.0	0.0
Anger	0.0	92.0	0.0	0.0	0.0	0.0
Disgust	0.0	0.0	100	0.0	0.0	0.0
Happy	0.0	0.0	0.0	100	0.0	0.0
Sadness	0.0	8.0	0.0	0.0	100	0.0
Surprise	0.0	0.0	0.0	0.0	0.0	100

Table 3 shows the confusion matrix for the proposed deep neural network (DNN) model. The model predicts almost all of the emotions correctly at 100% accuracy. For the "Anger" emotion, the DNN model predicted with 92% accuracy, and it is the only predicted class in which the model fails to predict with full accuracy. During the training procedure, the training log loss was 21.77% and the testing log loss was 45.45% for alpha value 1. To compare the performance of the outcome, we also generated two other confusion matrices based on the K-nearest neighbors and logistic regression models. This method was designed to take all of the data and compare them with the trained data by finding which points are closer to the others. Whenever we need to find the predicted or chosen point (K), in our case, we already have a trained set of data CK+, and we need to import new images into the algorithm. The primary process was to map the dataset with the new image data and obtain the desired results. Our process was based on the emotions we selected, which were sad, happy, angry, surprise, disgust and neutral. Therefore, when we input the picture, it could predict the result, although the result accuracy was slightly lacking.

For the K-nearest neighbor model, we used 99 n-neighbors, and the "sigmoid" function was also used in the classifier to calibrate the characteristics of the CK data sets. As shown in Table 4, we used the same data sets and features for these two models. It can be observed that the model can correctly predict "happy" facial expressions 100%, and the model we used to predict almost all of the emotions can predict correctly. As shown, this K-nearest neighbor failed to predict the "anger" and "sadness" emotions of these data. The other predicted class accuracy was less than 80% without the surprise emotions.

Table 4. Confusion matrix for K-nearest Neighbors model.

Predicted Class / Original Classes	**Neutral**	**Anger**	**Disgust**	**Happy**	**Sadness**	**Surprise**
Neutral	76.5	0.0	6.5	0.0	0.0	0.0
Anger	8.7	0.0	22.6	0.0	0.0	0.0
Disgust	4.9	0.0	64.5	0.0	0.0	0.0
Happy	1.1	0.0	0.0	100.0	0.0	0.0
Sadness	5.5	0.0	6.5	0.0	0.0	5.4
Surprise	3.3	0.0	0.0	0.0	0.0	94.6

Therefore, to evaluate the accuracy of the model more closely, we decided to test another model for the CK data set because the K-nearest neighbor models fail to predict two of six classes. Logistic regression predicted six out of six classes for the following dataset. The accuracy of the model is also quite impressive compared with that of K-nearest neighbors. For logistic regression, we used stochastic gradient descent learning and implemented regularized linear models. The confusion matrix of logistic regression is shown in Table 5. As shown in the table, the model correctly predicted "happy" facial expressions with 100% accuracy, similar to the two models discussed above, but its predictions of "sadness" are also less accurate. The model showed a bias for the "Neutral" facial expression because the logistic regression model predicted 50% of "sadness" and 2.3% of "surprise" face emotions as "neutral" emotions. We also tested random images from the Google image dataset, and the model perfectly categorized the emotions associated with the test images.

Table 5. Confusion matrix for logistic regression model.

Original Classes \ Predicted Class	Neutral	Anger	Disgust	Happy	Sadness	Surprise
Neutral	83.2	0.0	0.0	0.0	50.0	2.3
Anger	7.8	88.9	3.8	0.0	25.0	0.0
Disgust	3.0	0.0	92.3	0.0	0.0	0.0
Happy	0.0	0.0	0.0	100.0	0.0	0.0
Sadness	5.4	11.1	3.8	0.0	25.0	4.7
Surprise	0.6	0.0	0.0	0.0	0.0	93.0

6 Conclusion

Mobile data are useful for user behavior analysis [26] to enhance mobile intelligence, especially image data for emotion recognition. In this paper, we propose a novel bottleneck feature extraction pre-processing method, which improves on the traditional random facial emotion recognition method. In the BFE pre-processing model of bottleneck features, we used the Haar classifier to extract the faces of the images, the AdaBoost element algorithm was used to select important features, and the Viola-Jones detection algorithm was used in the cascade to focus on rapid discard of non-faces and spend more time on the possible face areas. In the feature extraction stage, we used the famous transfer learning VGG16 model. Finally, we used the bottleneck features generated by the BFE pre-processing method in a DNN training model with five dense layers.

Given the current mobile computing paradigm, most of the computing is done from the cloud, and the exchanging data is somewhat very busy or crowded. In such cases, we need to find the methods which are a little bit quicker. And the solution would be to shift from cloud to local devices. The cloud means is great but a little expensive in the long-run. So, it would be better ways, that we run some algorithms and work locally on the machine, and then send to the cloud when some tasks have been done locally.

We found that the BFE pre-processing method is suitable by simply looking at the training model results. The accuracy and F-measure of the model are outstanding, and the score on the test data is 98.59%. Although we used small-size data, the proposed model showed great feature selection ability and thus supplied good results. For benchmarking, we compared this model with the K-nearest neighbors and logistic regression neural network with the same features and dataset. The comparison results show that the test results of the two benchmark test modes are better. However, the natural images have a complex background, and it is difficult to extract faces from them. In the future, BFE is

expected to offer robust bottleneck features for complex input images. We also believe that this BFE technique can be used in any circumstance that works with or will work with images.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (61834005), the Enterprise Joint Fund Project of Shaanxi Natural Science Basic Research Plan (2019JLM-11-2), the Shaanxi Key Laboratory of network data analysis and intelligent processing, and the Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research (KAKENHI) under Grant JP18K18044.

References

1. Han, H., Liu, Z., An, J.: Mining mobile intelligence for wireless systems: a deep neural network approach. IEEE Comput. Intell. Mag. **2**, 24–31 (2020)
2. Hossain, M.S., Muhammad, G.: An emotion recognition system for mobile applications. IEEE Access **5**, 2281–2287 (2017). https://doi.org/10.1109/ACCESS.2017.2672829
3. Eleftheriadis, S., Rudovic, O., Pantic, M.: Joint facial action unit detection and feature fusion: a multi-conditional learning approach. IEEE Trans. Image Process. **25**(12), 5727–5742 (2016). https://doi.org/10.1109/TIP.2016.2615288
4. Happy, S.L., Routray, A.: Automatic facial expression recognition using features of salient facial patches. IEEE Trans. Affect. Comput. (2015) https://doi.org/10.1109/TAFFC.2014.2386334
5. Ryu, B., Rivera, A.R., Kim, J., Chae, O.: Local directional ternary pattern for facial expression recognition. IEEE Trans. Image Process. **26**(12), 6006–6018 (2017). https://doi.org/10.1109/TIP.2017.2726010
6. Celis, D., Rao, M.: Learning facial recognition biases through VAE latent representations. In: FAT/MM 2019 - Proceedings of the 1st International Workshop on Fairness, Accountability, and Transparency in MultiMedia, Co-Located with MM 2019, pp. 26–32 (2019). https://doi.org/10.1145/3347447.3356752
7. Shen, X., Gu, Y.: Nonconvex sparse logistic regression with weakly convex regularization. IEEE Trans. Sig. Process. **66**(12), 3199–3211 (2018). https://doi.org/10.1109/TSP.2018.2824289
8. Zhang, C., et al.: Multi-gram CNN-based self-attention model for relation classification. IEEE Access **7**, 5343–5357 (2019). https://doi.org/10.1109/ACCESS.2018.2888508
9. Chen, S., Pande, A., Mohapatra, P.: Sensor-assisted facial recognition: an enhanced biometric authentication system for smartphones. In: MobiSys 2014 - Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, pp. 109–122 (2014). https://doi.org/10.1145/2594368.2594373
10. Gu, Y., Wang, Y., Liu, T., Ji, Y., Liu, Z., et al.: EmoSense: computational intelligence driven emotion sensing via wireless channel data. IEEE Trans. Emerg. Top. Comput. Intell. (2019). https://doi.org/10.1109/TETCI.2019.2902438
11. Kiaee, N., Hashemizadeh, E., Zarrinpanjeh, N.: Using GLCM features in Haar wavelet transformed space for moving object classification. IET Intell. Transp. Syst. **13**(7), 1148–1153 (2019). https://doi.org/10.1049/iet-its.2018.5192
12. Sharifara, A., Rahim, M.S.M., Navabifar, F., Ebert, D., Ghaderi, A., Papakostas, M.: Enhanced facial recognition framework based on skin tone and false alarm rejection. In: ACM International Conference on Pervasive Technologies Related to Assistive Environments, Part F1285, pp. 240–241 (2017). https://doi.org/10.1145/3056540.3064967

13. Viola, P., Jones, M.: Robust real-time object detection. Int. J. Comput. Vis. **57**, 137–154 (2001)
14. Liu, F., Shen, C., Lin, G.: Deep convolutional neural fields for depth estimation from a single image. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2015). https://doi.org/10.1109/cvpr.2015.7299152
15. Facciolo, G., Limare, N., Meinhardt-Llopis, E.: Integral images for block matching. Image Process. Line (2014). https://doi.org/10.5201/ipol.2014.57
16. Karpathy, A.: CS231n convolutional neural networks for visual recognition. Stanford University (2016)
17. Brace, N., Kemp, R., Snelgar, R., Brace, N., Kemp, R., Snelgar, R.: Discriminant analysis and logistic regression. In: SPSS for Psychologists (2016). https://doi.org/10.1007/978-1-137-57923-2_11
18. Vasuki, A., Govindaraju, S.: Deep neural networks for image classification. In: Deep Learning for Image Processing Applications (2017)
19. Singh, V., Shokeen, V., Singh, B.: Face detection by haar cascade classifier with simple and complex backgrounds images using opencv implementation. Int. J. Adv. Technol. Eng. Sci. **1**(12), 33–38 (2013)
20. Valstar, M.F., Pantic, M.: Combined support vector machines and hidden markov models for modeling facial action temporal dynamics. In: Lew, M., Sebe, N., Huang, T.S., Bakker, E.M. (eds.) HCI 2007. LNCS, vol. 4796, pp. 118–127. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75773-3_13
21. Islam, M.F., Rahman, M.M.: Metal surface defect inspection through deep neural network. In: 2018 International Conference on Mechanical, Industrial and Energy Engineering, ICMIEE 2018, Khulna, Bangladesh, p. 258 (2018)
22. Ma, S., Bai, L.: A face detection algorithm based on Adaboost and new Haar-Like feature. In: 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS). IEEE (2016)
23. Wu, B., et al.: Fast rotation invariant multi-view face detection based on real adaboost. In: 2004 Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition. IEEE (2004)
24. Wang, Y., et al.: Real time facial expression recognition with adaboost. In: 2004 Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004. IEEE (2004)
25. Lemaître, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. J. Mach. Learn. Res. **18**(1), 559–563 (2017)
26. Yu, G., Zhang, X., Liu, Z., Ren, F.: BeSense: leveraging WiFi channel data and computational intelligence for behavior analysis. IEEE Comput. Intell. Mag. **14**(4), 31–41 (2019). https://doi.org/10.1109/MCI.2019.2937610