**CSE514 Data Mining: Review Essay of Human Emotion Classification**

October 29, 2018

**Team member**

Zhibo Liang: ID: 467574 Email: [zliangaf@gmail.com](mailto:zliangaf@gmail.com)

Xiangzhi Cao: ID: 465424 Email: [xcao22@wustl.edu](mailto:xcao22@wustl.edu)

Zubin Zhang: ID: 466774 Email: [zhangzubin123@outlook.com](mailto:zhangzubin123@outlook.com)

**Problem**

Emotion recognition has become one of the most important researching areas to enable effective human-computer interaction. Researchers have built sufficient fundamentals for us to study and succeed in this project. Ekman and Freisen [1] postulated six universal emotions with distinct content and unique facial expression, which are anger, fear, disgust joy surprise and sadness. We decided to build an Emotion Classifier to solve human emotion classification problem. Our classifier should be able to classify multiple human natural emotions using images as input.

**Objectives**

The main purpose of our project is identifying human emotions.

Our objectives based on the data categories:

1. Image Data. Successfully resize and grayscale the facial part of images. Also, be able to successfully convert image data to three-dimensional data points.
2. Data points. Be able to successfully extract the data of face landmarks and store into database in terms of three-dimensional coordinates (data points). Be able to successfully input data into trained CNN and get the results.

**Difficulties**

There are some difficulties that might occur while building our project:

1. Overfitting.
2. More than one emotion exists on an image.
3. Lab controlling of images in database. This may cause the problem of overfitting.
4. Unbalanced data. There may be too many similar data in database. (i.e. The number of ‘Happiness’ is a way greater than the number of ‘Sadness’)
5. Guaranteeing the accuracy of results after inputting data into trained CNN. This would be one of the biggest challenges for us since it is heavily relying on the training methods and data we will select.

**Methods**

In this section, we have proposed three ways to solve this problem. Then we will compare these three methods and choose one to develop in our project.

**Method 1: Multiple Deep Network Learning**

**Face Processing**

Face Processing is a crucial step for better recognition performance. It can remove irrelevant noise and integrate all faces to the same domain.[2] In our project, we plan to resize all detected faces to 48 X 48 and transform them to grayscale. Then, we will use standard histogram equalization to process these face images, then removing unbalanced illumination by a linear plane fitting. At last, the image pixel values are normalized to a zero mean and unit variance vector.

**Classification Module**

The classification Module we plan to use is the ensemble of multiple deep convolutional neural networks (CNN). Each CNN model is initialized randomly and pre-trained on the dataset.[3] The diagram of our CNN architecture lies below:

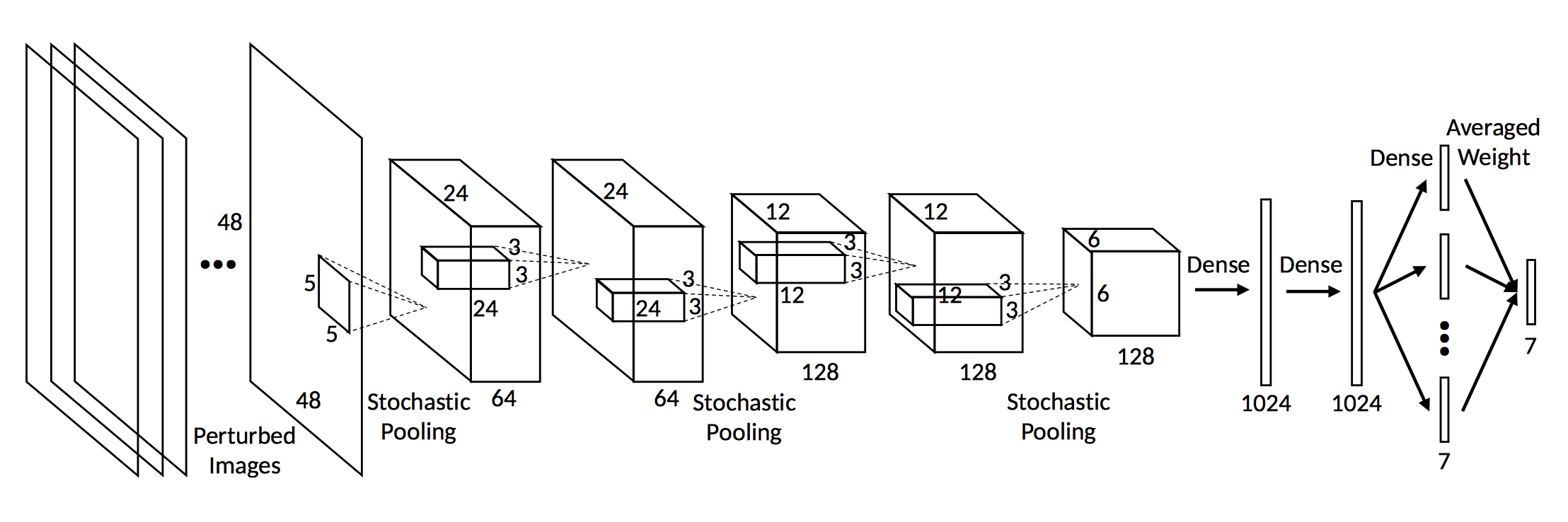


Figure 1: CNN architecture

You can see that this neural network contains five convolutional layers, three stochastic pooling layers and three fully connected layers. The input to the network is the preprocessed 48 × 48 faces. Convolutional layers are used to feature extraction. And in our project, we use stochastic pooling instead of max pooling. Stochastic pooling is randomly sampling a response based on the probability distribution obtained by normalizing the responses, which means the probability of being selected with a large element value is also large. The nonlinear mapping functions for all convolutional layers and fully connected layers are set as rectified linear unit (ReLU):

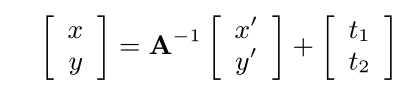


By the way, the fully connected layers contains dropout, which is another method of randomization.

Each CNN model will be pre-trained on the dataset. We will try to set an initial network learning rate and a minimum learning rate. Samples from each training epoch are randomly selected from the training set and with random perturbation.[3] The learning rate can be adjusted actively through the learning period. (Specific number to adjust learning rate can be discussed again) After all epochs are finished, we select the neural network from the epoch with the best training accuracy as our final pre-trained model.

What’s more, we need to fine-tune our network on the training set. In order to avoid overfitting, we may try to freeze the parameters of all the convolutional layers and only allow the update of parameters at the fully connected layers. We know that a slightly larger learning rate helps to reduce the risk of trapping at local minima and benefits the fine-tuning performance. So, in our project we can slightly increase the value of learning rate.[4]

In order to improve classification performance, we introduce random perturbations and voting in our convolutional neural networks (CNN) architecture. The random perturbation essentially generates additional unseen training samples and therefore makes the network even more robust to deviated and rotated faces. Because of the difficult and wild nature of the training dataset, the detected faces may contain a wide reality of different poses, cropped scales and deviations. We consider a comprehensive set of perturbations through the randomized affine image warping:



Where A is the composition of the skew, rotation and scale matrices. The input x’, yare the pixel coordinates of the warped image. t1and t2 are two translation parameters whose values are sampled from (0,) and is a randomly sampled integer on [0,4]. As the computed mappings mostly contain non-integer coordinates, bilinear interpolation is used to obtain the perturbed image pixel values. For pixels mapped outside the original image, we take pixel value of its mirrored position. We train multiple convolutional neural networks and these CNN structures may not be the same. We output their training responses to learn the ensemble weights w. There are two optimization frameworks to consider in our project:

1. Optimal Ensembled Log Likelihood Loss

Here is the objective function:

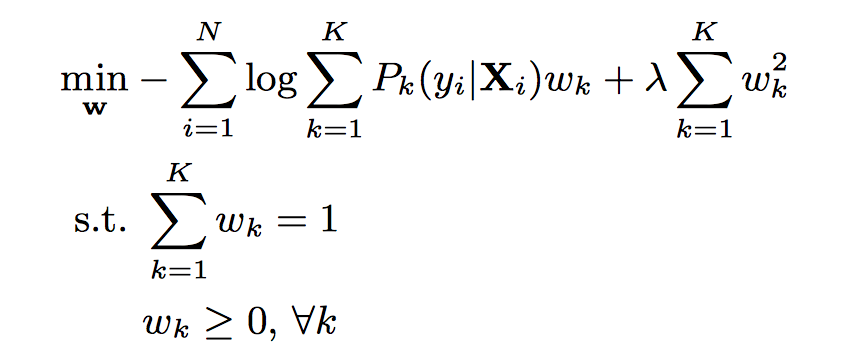
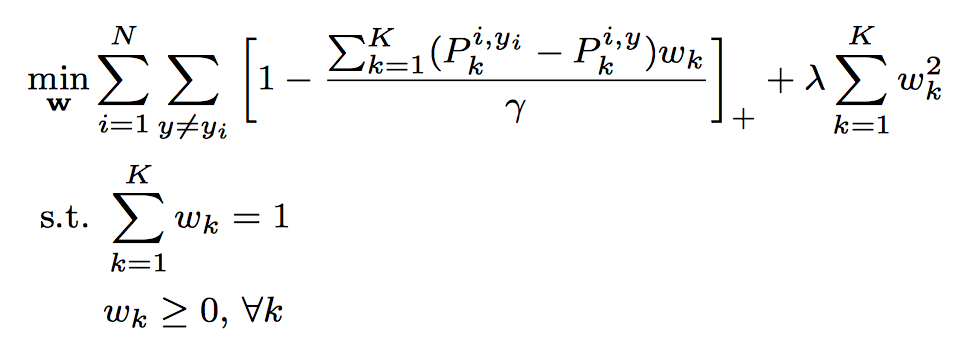


Figure 2: Optimal Ensembled Log Likelihood Loss

N is the number of training samples, and K is the number of networks. Pk (y|Xi ) is the kth network output response on the yth category given the set of perturbed samples Xi, λ is determined by maximizing the validation accuracy.

2. Optimal Ensembled Hinge Loss

Here is the objective function:



where

Figure 3: Optimal Ensembled Hinge Loss

The intuition is that the ensemble output response corresponding to ground truth should be larger than others with a margin γ. With the hinge loss, any case where the response difference is larger than γ will not introduce any penalty. Again, both γ and λ are determined with respect to the accuracy on validation set.

**Data**

We will use an image database with total 13718 different images. The images are divided into three different sets: training set (60%), validation set (20%) and test set (20%). These percentages may change a little according to our actual training result.

For each image in training set, we will generate some randomized perturbation of it because of two benefits. First, including randomized perturbation will make the network more robust to the deviated and rotated faces. Second, we can control the number of perturbation of each image to make the database more balance. In the database we used, there are more images with label “HAPPINESS” and “NEURAL”, and fewer images with the rest labels. So, we will produce less perturbation for images with label “HAPPINESS” and “NEURAL”. This can solve the problem caused by imbalance database. The training episode is expected to set to 20.

After the training, we use validation set to optimize the parameter of CNN. After optimization, we use both training data and validation data to train the final model. Finally, we will use test data to test our final model. The evaluation criteria is accuracy. We will calculate the overall accuracy, the recall as well as the false positive rate of different labels.

In order to deal with the overfitting issue, we will use the technique such as weight regulation, dropout and early stop.

**Method 2: Deep Autoencoders + Multiple SVMs**

**Image face processing**

Using face detection and extraction of face region to eliminate redundant regions. Using most discriminating parts of the face when facial expression changes.

**Feature Extraction**

This method uses Histogram of oriented gradients (HOG) to detect the variations of the face images, HOG is good for geometric and photometric transformations, but it does not work well for object orientation.[5]

**Dimension Reduction using Autoencoder**

Using HOG features may result in having high dimension as compared to the number of images. Autoencoder is used to reduce dimension in this method. An autoencoder is an unsupervised architecture that replicates the given input to its output. It uses an input feature vector X and learns a code dictionary by changing the raw input data from one representation to another. Backpropagation is applied to the autoencoder by setting the target values to be the equal to input values, as shown in the Figure 4.

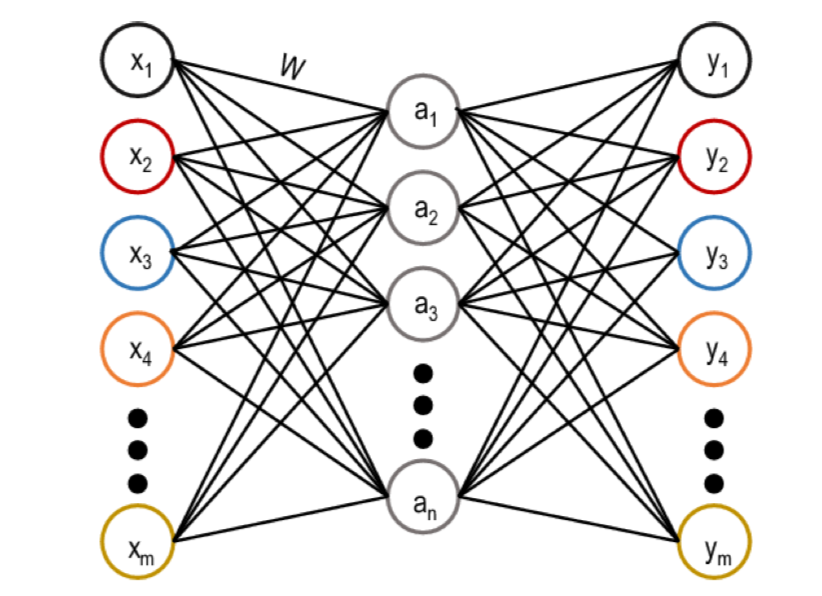
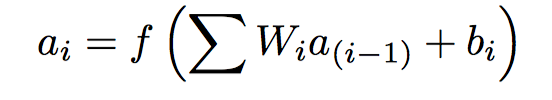


Figure 4: Architecture of an autoencoder network

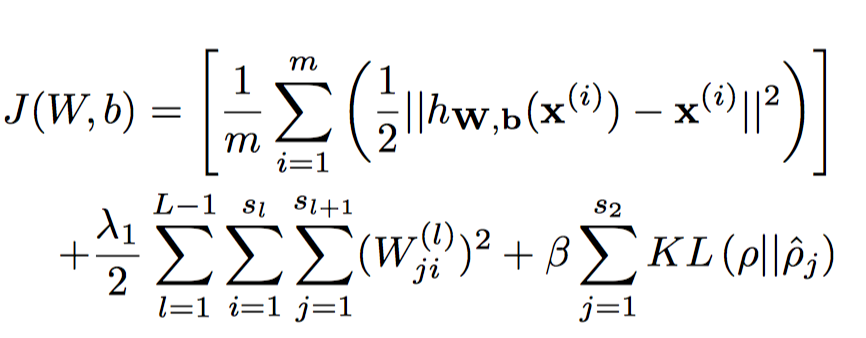
In an autoencoder, a lower dimension a can be represented as follow:



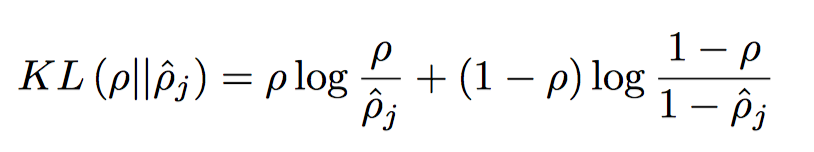
The stacked autoencoder can be represented as follow:



An autoencoder can construct a structure of data with large number of hidden units by imposing sparsity constraint on hidden units. This architecture is known as sparse autoencoder, and the cost function J(W, b) of a sparse autoencoder can be represented as follow:



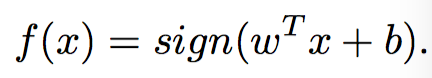
Where hW,b(x) is an activation function, W is weights and b is biases, Λ is the parameter for weight decay, L is the number of layers in autoencoder network, sl is the number of units for lth layer, W(l)ji represents the weight value between ith unit of layer l and jth unit of layer l+1, b(l)i is the bias related with unit i in layer l+1, β is the weight of third term, and ρ is a sparsity parameter. The first part of the equation is minimizing the difference between the input and output. The second part of the equation is the weight decay, and it is used to avoid overfitting. The third part of the equation is sparse penalty term, where KL is the Kullback-Leibler(KL) divergence that can be represented as follow:



The value of ρ is close to 0.

**Support Vector Machine**

One-vs-all SVM with Gaussian kernel function can be used to classify k-classes of data by constructing k separate binary classifiers. The mth class uses the positive values and the rest of k-1 number of classes use negative values when binary classifier is trained using the data from mth class. For binary classification with training data xi (i=1, 2, 3…, N)and corresponding labels yi = +-1, and the decision function can be represented as follow:



Where wTx + b = 0 is a separating hyper-plane, w is a weight vector normal to the separation hyper-plane and b is the bias of hyper-plane.

**Method 3: Cauchy Naive Bayes Classifier**

**Features for emotion recognition**

In this method, we introduce a wireframe model for the face. The model consists twelve facial motion measurements being measured for facial expression recognition. The combination of these measurements’ features define the 7 basic classes of facial expression---Neutral, Anger, Disgust, Fear, Happiness, Sadness and Surprise. The figure 4 will show that, and the arrow represents the motion direction away from the neutral position of the face.

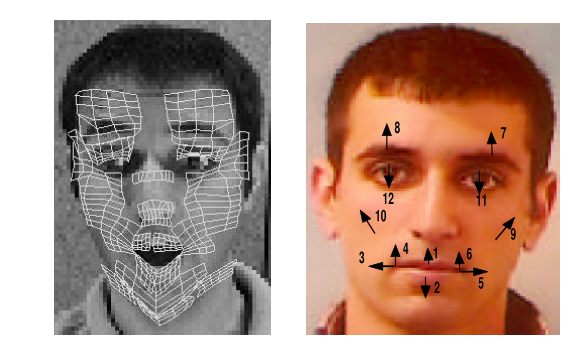


Figure 4 The wireframe model and the facial motion measurements

**Cauchy Naive Bayes**

The class label yand the feature vector for the observed data x. Under the maximum likelihood framework (ML), the problem is to model the probability of features given the label P(xi |y). At past, we usually use a Gaussian distribution and the ML is used to obtain the estimate of mean and variance. In our project, we use Cauchy distribution, We need to estimate the parameters of the Cauchy distribution. For a sample of size n sampled from the Cauchy distribution the likelihood is that:

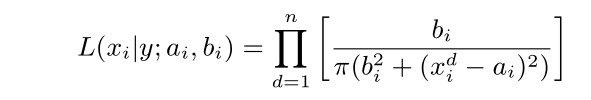


Figure 5 The likelihood of a sample of size n sampled from the Cauchy distribution

Where ai is the location parameter, bi is the scale parameter. Let ^ai and ^ bi be the maximum likelihood estimators for ai and bi, we need to solve the equations for ^ai and ^bi. In this project, we use a Newton-Raphson iterative method with the starting points given by the mean and variance of the data. And the maximum likelihood equations are:

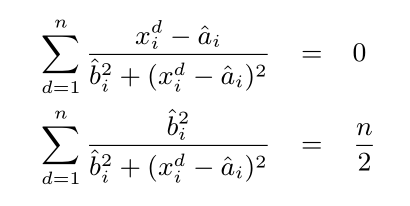


Figure 6 the maximum likelihood equations

**The algorithm we use**

1. For each class consider the corresponding training set and estimate the parameters of the model using the ML framework.
2. Extract a random sample from the training set and perform classification. The model which provides the best results will be assigned for each individual class in the classification step.
3. Perform classification using only the test set. [6]

**Method conclusion**

In this section, we have discussed three different methods for image class classification, which are:

1. Multiple Deep Networks Recognition: We build multiple CNNs and assign weights to each CNN to ensemble them together. We use the technique such as perturbation, pre-training, weight regulation, drop out and early stop to make each of the CNN more robust.
2. Deep Autoencoders + Multiple SVMs: Firstly, we do pre-process to image data. Secondly, we extract high-dimensional Histogram of oriented gradients as feature. Then, we use deep autoencoders to decrease the dimension of feature and train multiple SVMs for emotion classification.
3. Cauchy Bayes Classifier: Firstly, we extract pre-defined action units (AU) features for image data. Then we build a Bayes classifier based on Cauchy distribution.

After the researching, we decide to use Multiple Deep Networks Recognition to develop our project because of the following reasons:

1. Nowadays, Deep learning is one of the most popular machine learning method.
2. Powerful libraries, such as TensorFlow and Kearse can be used to build CNN model conveniently.
3. Feature extraction is done by CNN, we do not need to extract them by ourselves.
4. Multiple Deep Networks provide the best outcome among these three methods.

We will first build the multiple deep networks. If we have enough time, we will continue building the autoencoder and try to find whether combine autoencoder to CNN will provide a better prediction result.

**Work plan**

We will use GitHub to manage our project. Our project is mainly divided into four parts: data pre-processing, CNN building, optimization and visualization. There are totally three people in our project team (Zhibo Liang, Xiangzhi Cao, Zubin Zhang)

1. Data pre-processing: analysis the original data (the number of data that belongs to different labels), data rescaling, converting to grayscale and randomized perturbation. (Zubin Zhang)
2. Building the CNNs and multiple network learning framework. (Zhibo Liang)
3. Data visualization and most work of optimization. (The rest of two teammates will also take part in optimization) (Xiangzhi Cao)

**Final Goal**

Through this project, we want to find out what kind of Neural Networks works the best for emotion recognition and are these methods we implement all doable for emotion recognition. Furthermore, we want to analyze the important features of emotion recognition.

**Milestones**

11.5 finish pre-processing of image data (a smaller database)

11.8 finish building one neural network

11.11 finish optimization the neural network

11.14 finish building of the prototype of neural network

11.25 finish building multiple neural network

11.30 finish the assembling of multiple neural networks

If there is still time left, we will do more optimization of the whole model.

**References:**

[1] K. Anderson and P. W. McOwan, “A real-time automated system for the recognition of human facial expressions,” IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 36, no. 1, pp. 96–105, 2006

[2] Yu Z, Zhang C. Image based static facial expression recognition with multiple deep network learning[C]//Proceedings of the 2015 ACM on International Conference on Multimodal Interaction. ACM, 2015: 435-442.

[3] Dachapally P R. Facial Emotion Detection Using Convolutional Neural Networks and Representational Autoencoder Units[J]. arXiv preprint arXiv:1706.01509, 2017.

[4] Ng H W, Nguyen V D, Vonikakis V, et al. Deep learning for emotion recognition on small datasets using transfer learning[C]//Proceedings of the 2015 ACM on international conference on multimodal interaction. ACM, 2015: 443-449.

[5] Usman M, Latif S, Qadir J. Using deep autoencoders for facial expression recognition[C]//Emerging Technologies (ICET), 2017 13th International Conference on. IEEE, 2017: 1-6.

[6] Sebe N, Lew M S, Cohen I, et al. Emotion recognition using a cauchy naive bayes classifier[C]//null. IEEE, 2002: 10017.