Human emotion recognition based on facial expression

Zihao Wang
University of Miami
Department of Computer Science
Coral Gables, USA
zxw526@miami.edu

***Abstract*—This research proposes a method for training convolutional neural networks for human facial expression image classification. The trained networks may be used for human emotions analysis in many applications, such as ensuring safe driving and human-computer interaction. Commonly, a neural network can be trained by using whole face images. However, face outline can influence emotion recognition. Since the face outline can influence the emotion feature extraction. As a solution to this problem, we proposed to train a network on the key points of facial images. The average classification accuracy of the network is 33% (7 emotion classes), which shows it is a promising method.**

Keywords—Machine learning; neural network; facial expression; image classification

I. INTRODUCTION

Facial expression analysis is essential to human-computer interaction [1]. Facial expression analysis is important in social interaction as well as social intelligence [2]. Facial expressions are used for recognizing human emotions like happiness, anger, fear, etc. Facial expressions can be different among individuals.

There are four basic steps for facial expression analysis. The first step is face recognition, to get the face from an image. The second step is regulation, to normalize the size and reducing illumination effects. The third step is feature extraction, to obtain useful features and reduces irrelevant information. The fourth step is classifiying emotions into different categories.

There are two representation models for facial expressions. The one is continuous; the other is categorical [3]. The categorical model plays a more critical role than the continuous model. Continuous models represent each emotion as a vector in continuous space. The categorical model tags each facial expression image into one of the seven prototypical expressions which are neutral, angry, disgust, fear, happy, sad and surprise. Peoples' appearance also influences the human expression classification [4]. In a convolutional neural network, facial appearance may cause some noise for facial feature extraction in the convolutional layer. This project classifies facial images into discrete categorize by using the convolutional neural network (CNN) based on the facial expression feature map.

II. RELATED WORK

There are two parts of facial expression analysis. The first part is the facial feature extraction. The second part is the facial expression classification.

For facial feature extraction, Gabor coding is used to represent facial expressions [5]. To get the Gabor code similarity space, feature maps which have the same spatial frequency and orientation preference were compared at corresponding points by using normalized dot product. The 34 node grid is used to represent facial geometry (Fig. 1). The similarity of one image grid and another grid which is calculated the average points overall the corresponding images.

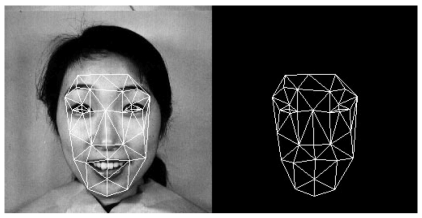

Figure 1: The 34 node grid used to represent facial geometry [5]

To extract the local information of facial images, facial images are divided into equal number of blocks [3]. A histogram is calculated for each of the blocks. These histograms are normalized and concentrated together to form the Accumulated Motion Descriptor (AMD) (Fig. 2).

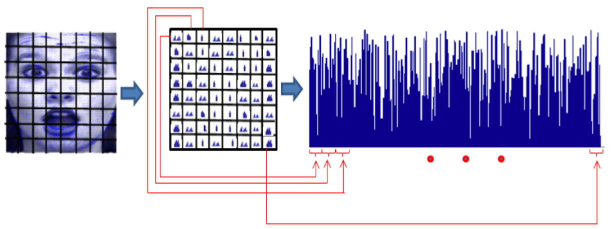

Figure 2: Partitioning and accumulated histograms [3]

For image feature classification, there are some classical CNNs, the earliest and the simplest is Lenet-5 [6].

Lenet-5 is used for MNIST digit image recognition. There are five layers, including two convolutional layers, two fully connected layers and one output layer. The input size is 32x32; the output is a one-dimension vector with a length of 10 (Fig. 3).

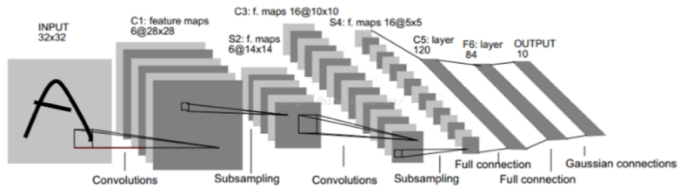

Figure 3: The structure of Lenet-5 [6]

To adapt the CNN into facial expression recognition, two parallel connected CNN was implemented [2]. One network is used to classify expression into seven categories; the other is used to classify people into ten different models (Fig. 4). The dataset is JAFFE which contains 213 images of 7 facial expressions (6 basic facial expressions + 1 neutral) from 10 Japanese female [7].

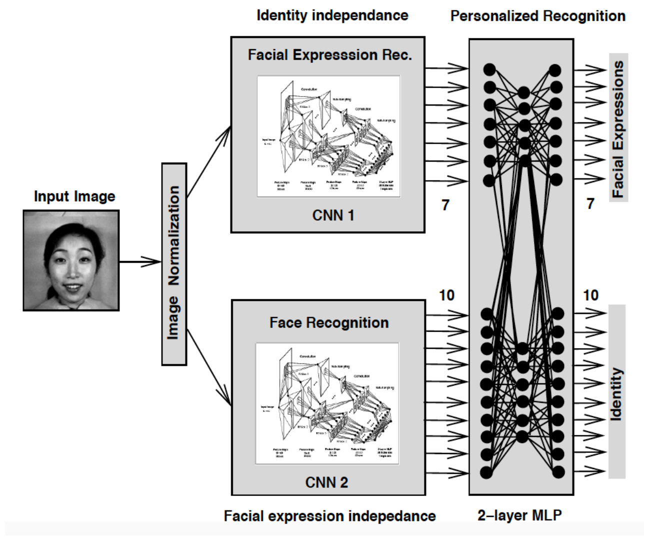

Figure 4: The structure of facial expression recognition CNN [2].

Based on this work [2], A CNN with two parts is designed in this research.

The organization of the paper is as follows. Section III describes the details of preprocessing of facial feature extraction and the architecture of the CNN. Section IV describes the results and outcomes of the methods proposed. The paper concludes in section V.

III. Methods of Use

This project uses a CNN to recognize facial expression in facial images. The research includes three parts, inclduing data capture, preprocessing, training and testing. The flow diagram is illustrated in Figure 5.

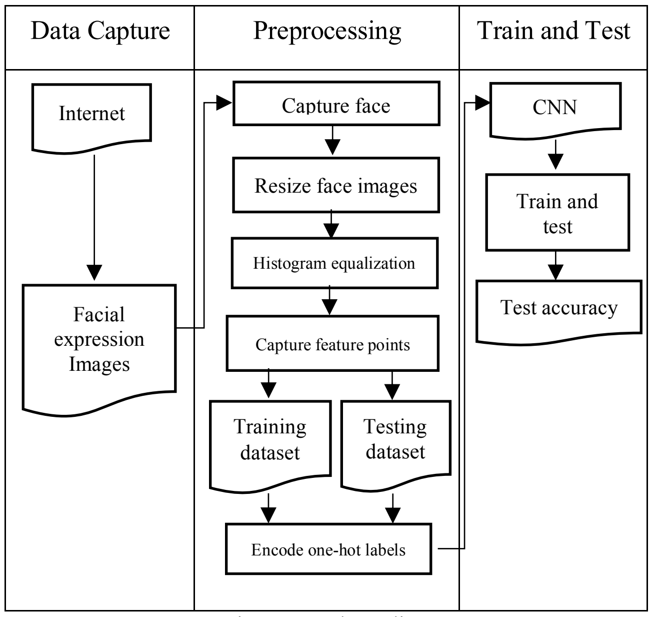

Figure 5: Flow diagram

A. Data Capture

The training dataset comes from CK+, KDEF and fer2013. The testing dataset is JAFFE. There are seven categories of expressions. Each expression corresponds to a number between 0 and 6, defined as follow: 0=Neutral, 1=Angry, 2=Disgust, 3=Fear, 4=Happy, 5=Sad, 6=Surprise

B. Preprocessing

There are five steps in preprocessing. First, capturing face from original images by using dlib library. Second, resizing face images into 64x64 pixels. Third, using OpenCV library to apply histogram equalization to images from the second step. Forth, capturing 68 feature points by using dlib library and get image feature maps. Fifth, changing decimal labels into one-hot labels.

The original face images size has 256x256 pixels (Fig. 6). There is some irelevent information need to be removed from the images, such as hair and outer space on the boundary.

Figure 6: Original face image sample

First, dlib is used to cut off the aeras of hair and outter space. dlib has a face detector method called get_frontal_face_detector which can be used to crop face in an image: (1) read original face images from the operating system; (2) create a detector object to hold face images; (3) get the face boundary coordinates parameters which are bottom, top, left and right; (4) finally, save images which only have facial information (Fig. 7). The process is summarized in Algorithm 1.

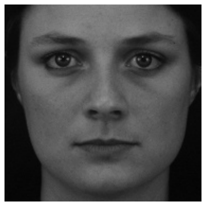

Figure 7: Image after catch face

Algorithm 1 Algorithm for catching faces

```
for filename in os.listdir(path_i):
   image_path=path_i+filename
   img = cv2.imread(image_path)
   detector = dlib.get_frontal_face_detector()
   face = detector(img, 1)
       for k, d in enumerate(face):
            height = d.bottom() - d.top()
            width = d.right() - d.left()
            face_matrix = np.zeros([height, width, 3], np.uint8)
           for m in range(height):
             for n in range(width):
               face_matrix[m][n][0] = img[d.top()+m][d.left()+n][0]
               face_matrix[m][n][1] = img[d.top()+m][d.left()+n][1]
               face_matrix[m][n][2] = img[d.top()+m][d.left()+n][2]
     cv2.imwrite(save_path, face_matrix)
```

Second, to resize the image into 64x64 pixels, a method in opencv called resize is used. The details are in Algorithm 2.

Algorithm 2 Algorithm for resizing images

```
for filename in os.listdir(path_i):
      src=path_i+filename
      dir=path_out_i+filename
      img=cv2.imread(src)
      img2=cv2.resize(img,(64,64))
      cv2.imwrite(dir,img2)
```

Third, to do histogram equalization, the opencv method equalizeHist is used. Algorithm 3 shows this process.

Algorithm 3 Algorithm for histogram equalization

```
for filename in os.listdir(path1_i):
      src=path1_i+filename
      dir=path2_i+filename
      img=cv2.imread(src,0)
      e=cv2.equalizeHist(img)
      cv2.imwrite(dir,e)
```

Forth, the dlib library is also used to get 68 feature points of an image face. A predictor with 68 feature points model needs to be loaded (Fig. 8). Then the predictor combined with the detector in the first step is used in this step. The details are in Algorithm 4.

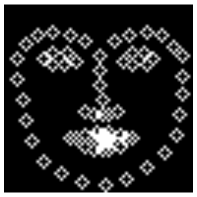

Figure 8: Facial feature map

Algorithm 4 Algorithm for getting 68 feature points

```
for filename in os.listdir(path1_i):
        src=path1_i+filename
        dir=path2_i+filename
        img=cv2.imread(src)
        img_gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        img_circle = np.zeros([64, 64])
        faces = detector(img_gray, 0)
        if len(faces) != 0:
            for i in range(len(faces)):
               landmarks = np.matrix([[p.x, p.y] for p in predictor(img,
                                  faces[i]).parts()])
                 for idx, point in enumerate(landmarks):
                       pos = (point[0, 0], point[0, 1])
                        cv2.circle(img_circle, pos, 2, color=(255, 0, 0))
                        cv2.imwrite(dir, img_circle)
```

Finally, each numerical label needs to be encoded into a one-hot label to deal with the multiclass classification problem. A one-hot vector is a 1 × N matrix (vector) used to distinguish each word in vocabulary from every other word in the vocabulary. The vector consists of many 0s and only a single 1 to uniquely identify the target class. For example, 0 (Neutral) is encoded as (1,0,0,0,0,0,0), and 1 (Angry) is encoded as (0,1,0,0,0,0,0). The steps are shown in Algorithm 5.

Algorithm 5 Algorithm for One-hot encoding

```
def onehot(labels,row, column):
   onehot=np.zeros([row,column])
   for i in range(row):
      index=labels[i][0]
      onehot[i][index]=1
   return onehot
```

C. *Training and testing*

The structure of CNN in this research is modified from LeNet-5. The CNN include four convolutional layers and two fully connected layer module. The CNN structure is shown in Table 1.

Table 1: The structure of CNN

layer	Input size	Number of filters	Filter size	stride	Outpu t size	Active function
Conv1	64x64	6	5x5	1	60x60	Relu
Pool1	60x60	6	2x2	2	30x30	null

Conv2	30x30	6	5x5	1	26x26	Relu
Pool2	26x26	6	2x2	2	13x13	null
Conv3	13x13	16	5x5	1	9x9	Relu
Pool3	9x9	16	2x2	2	4x4	null
Conv4	4x4	120	4x4	1	1x120	Relu
Full1	120	null	null	null	84	Sigmoid
Full2	84	null	null	null	7	Softmax

The output size is calculated by the formula:

$$output\ size = \frac{input\ size - filter\ size}{stride} + 1$$

To speed-up traning process, Adam Optimizer is used to adjust the learning rate automatically. The loss function is cross-entropy cost function:

$$C = -\frac{1}{n}\sum_{x}[yln\hat{y}]$$

n is the number of sample in one batch. x is one input sample, $\hat{y}$ is output from the CNN, and y is ideal output.

$$\hat{y} = \sigma(z),$$

$$z = \sum_{x} w_j x_j + b$$

The gradient of weight and bias are:

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_{x} x_j(\sigma(z) - y),$$

$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_{x}(\sigma(z) - y)$$

In cross-entropy cost function, the gradient descent is not dependent on the derivative of active function. It's influenced by the difference between the actual output and ideal output. The bigger difference they have, the large magnitude of the gradient.

The steps of training and testing are shown in Algorithm 6.

Algorithm 6 Algorithm for training and testing CNN

```
Input: Preprocessed training dataset
Output: Trained weights and bias
Step1: Load preprocessed training and testing dataset
Step2: Get images labels and One-hot labels
Step3: For each iteration, feed the minimum batch of training.
       dataset and calculate the training accuracy and loss
Step4: Feed testing dataset after feed one minimum batch of
       training dataset, getting testing accuracy and loss
Step5: Jump to Step3 and do next iteration until accuracy and loss.
       converge
```

IV. Experiment design and Results

The experiment is done on the training dataset CK+, KDEF and fer2013 and testing dataset JAFFE. After preprocessing, the number of valid data in training and testing dataset are 9690 and 210. The minibatch training method is used to train CNN [7]. The training dataset is divided in to 10 batches, each batch has 960 images. The total number of training epochs is 2000. The initial learning rate is 0.001.

Results analysis includes 3 parts: 1) Analyze the training loss 2) Analyze the training accuracy; 3) analyze the testing accuracy.

A. The training loss

During the 2000 training epochs, the training loss decrease visibly. The average loss of all batches in each epoch is illustrated in Figure 9. At the beginning, the training loss is close to 1.95. After 2000 epochs, the training loss is approximately 1.3

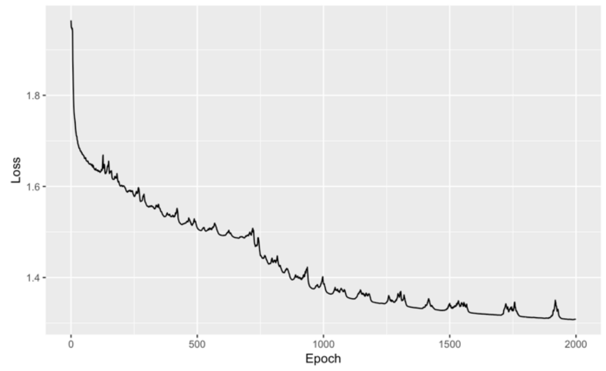

Figure 9: Loss trend

In the beginning, the loss decreased rapidly, then it slowed down after 1500 epochs.

B. The training accuracy

To calculate the training accuracy, we need to compare the actual output and one-hot label by judging if the position of 1 is the same in the label. The accuracy of one batch is:

$$accuracy = \frac{the\ number\ of\ correctly\ classified\ images}{batch\ size}$$

The average training accuracy of all batches over 200 epochs is illustrated in Figure 10.

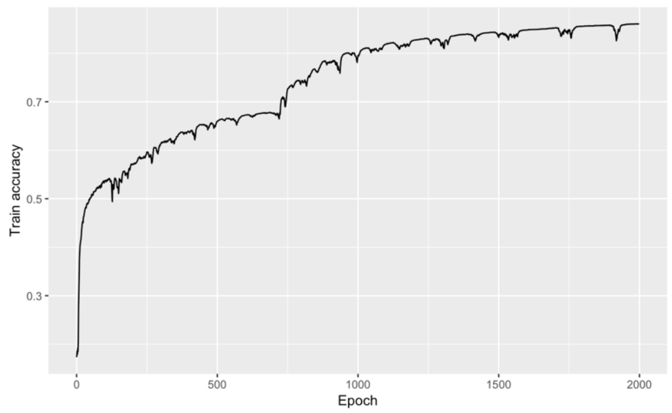

Figure 10: The training accuracy

The accuracy trend is smilar to the trend of loss function. In the beginning, the accuracy increases dramatically. After 1000 epochs, the training loss function converges. In the beginning, the training loss is approximately 0.15. After 2000 epochs, the training accuracy is close to 0.85.

C. The testing accuracy

The definition of testing accuracy is the same as training accuracy. The average testing accuracy of all batches in each epoch is illustrated in Figure 11.

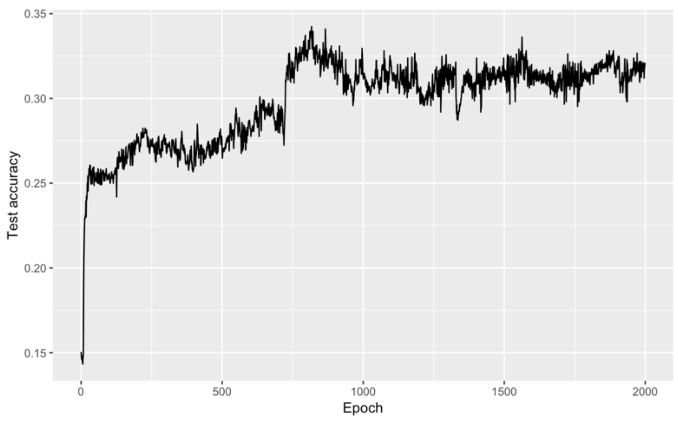

Figure 11: The testing accuracy

The testing accuracy increases dramatically at the beginning. After 500 times, the training accuracy converge. After 2000 times, the testing accuracy is close to 0.33. We note that the accuracy of random guess is 1/7 (0.1429).

V. Conclusion and future work

A convolutional neural network for human emotion recognition is developed. The testing accuracy still needs to be improved. The information of the facial feature points is important for emotion recognition. Futhur work will focus on how to utilize those feature points to improve testing accuracy.

References

[1] W. Liu, C. Song, and Y. Wang, “Facial expression analysis using a sparse representation based space model,” in *2012 IEEE 11th International Conference on Signal Processing*, 2012, vol. 3, pp. 1659–1662.

[2] H. P. Mal and P. Swarnalatha, “Facial expression detection using facial expression model,” in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, 2017, pp. 1259–1262.

[3] S. Agarwal and D. P. Mukherjee, “Decoding mixed emotions from expression map of face images,” in *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 2013, pp. 1–6.

[4] B. Fasel, “Robust face analysis using convolutional neural networks,” in *Object recognition supported by user interaction for service robots*, 2002, vol. 2, pp. 40–43 vol.2.

[5] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, “Coding facial expressions with Gabor wavelets,” in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 200–205.

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[7] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, “Coding facial expressions with Gabor wavelets,” in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 200–205.

[8] R. Shimizu, K. Asako, H. Ojima, S. Morinaga, M. Hamada, and T. Kuroda, “Balanced Mini-Batch Training for Imbalanced Image Data Classification with Neural Network,” in *2018 First International Conference on Artificial Intelligence for Industries (AI4I)*, 2018, pp. 27–30.