Facial Expressions Classification by Convolutional Neural Network

Fengdan Cui

Research School of Computer Science,
The Australian National University, Canberra, Australia

U6589143@anu.edu.au

Abstract. Based on the excellent performance of CNN in image feature extraction and classification, I am committed to implementing a CNN model to classify facial expressions of the face images in the subset of database Static Facial Expressions in the Wild (SFEW). In this paper, I discuss the approach for image pre-processing and the architecture of the CNN model for extracting features and classifying. After training 50 epochs and testing every 5 epochs, the highest accuracy of the model can reach 49.26%, which is higher than BDNN model with the accuracy of 16.14% and another research, both of which focus on classifying images in the same dataset. At the end of the paper, I propose further work that can be done to improve the performance and universality of this CNN model.

Keywords: Convolutional Neural Network, Bidirectional Neural Network, Facial expressions, Classification

1 Introduction

As a nonverbal signal of human beings, facial expressions contain rich personal and social communication information, it also conveys some cognitive behavior and psychological information, which can't be replaced by other expression methods. Therefore, accurately recognizing the expressions of others is essential for successful interpersonal communication. It is obvious that humans can quickly recognize face emotions of others through a powerful brain system, and the rapid development of computer vision makes it also possible for machines. A large number of scientists are devoted to the study of facial signals. For example, researchers took advantage of multilayer perceptions to process the extracted facial images from frames and obtain the sequence of classified emotions [1]. This reflects the importance of facial signal recognition and analysis in the field of human-computer interaction.

Facial data can come from both static images and dynamic videos. As the MMI facial expression database that is explained in a report, it contains up to 1500 static facial images showing various expressions and can be accessed online for researching on facial information [2]. There are also scientists who implemented a model to automatically extract and recognize human facial expressions from real-time video on time basis [3]. In this paper, I selected two format of dataset which are extracted from dataset Static Facial Expressions in the Wild (SFEW) , which is collected from various movies and close to the real world [4]. In this case, the results of the experiment are more meaningful in reality and helpful for further research. One is the data for training BDNN obtained by processing the original image, which contains 675 lines and 10 columns which represent 2 high quality image features, and 1 column to represent the label, ranging from 1 to 7 to stand for seven different facial expressions. The other dataset contains 682 original images for training CNN which are stored in seven different folders according to different facial expressions, which are angry, disgust, fear, happy, neutral, sad, and surprise.

Neural networks perform distributed parallel data processing by simulating the behavior of internal systems in animals. Depending on the complexity of the system, the network can adjust a large amount of connection relationships between internal nodes so as to process the data effectively. In addition to forwarding data, some models also allow data to be processed in reverse direction, which is useful for processing serial data like sentences and videos, where the following data is related to the previous data. For example, by applying bidirectional LSTM to recognize and classify phoneme based on TIMIT speech corpus, the results showed that the bidirectional LSTM is better than unidirectional LSTM and simple RNN [5]. Research has also shown BDNN can be applied to recognize characteristics and find the center of clusters [6]. Therefore, bidirectional network model is effective in improving the neural network performance.

In the field of deep learning, convolutional neural networks are one of the representative neural networks, which are outstanding in image analysis and processing such like feature extraction classification and scene recognition. A deep convolutional network model was built up to achieve the effective classification of a large-scale images in the ImageNet LSVRC-2010 and the breakthrough progress made by the variant model was witnessed in 2012 [7]. Furthermore, it was also applied to automatically recognize human behaviors in surveillance video by implementing a 3D CNN model to extract features and analyze the motion information, which achieved better performance than a standard method [8]. It can be illustrated that the CNN can directly extract the features of the original image and then perform classification, avoiding the complicated preprocessing of the image by traditional image processing algorithms.

Therefore, I aimed to implement both BDNN model and CNN model to classify the data in the selected dataset into seven different facial expressions and emphasized the CNN model which is professional in image recognition and classification and achieved better result. In order to evaluate the performances of both models, I discussed and compared the classification results of both two models and compared with another existing research.

2 Method

In this section, I described both techniques for emotion classification and emphasized the process of the CNN model, including the data preprocessing, model architecture, training and testing process, and evaluation method.

2.1 Bidirectional Neural Network (BDNN)

The given feature data is composed of two kinds of PCA values, each one represents a significant feature and contains five columns, which can't be as input for the network directly. I applied two different methods to preprocess the data. One is to calculate the means of each PCA and get the final two columns that can be seen as two features. The other is to calculate the convolution of the two-dimensional (2x5) data and obtain the final one-dimensional (1x5) data as input. Although I tried both two approaches and the results are similar, I would discuss the following sections based on the later method because the convolution result is more representative than the simple average theoretically. According to the requirements of the model, the size of input should be nx8 so that one more column with labels and two more columns with zero values should be added to form a dataset with 8 columns.

Then the data was randomly split into training data and testing data. As the requirements of BDNN model, the weight matrix should be inversible and the dimensions of input vector and weight vector should be the same in order to easily infer input value from output value. For such purpose, I grouped every seven-row of training data into two new two-dimensional arrays, representing the batches of input data and target data respectively. Therefore, the original training data with the shape of *(n, 8)* was changed to the input vector with the shape of *(batch_num, 7, 7)* and target vector *(batch_num, 7, 1)*.

The bidirectional neural network model is made up of multilayer completely connected neurons. Firstly, the data is processed from the layer to output layer. I chose cross entropy loss function to calculate the error between actual values and predicted values. Then, the model parameters were updated by applying back propagation algorithm and gradient descent. After 50 epochs, the direction for data transmission was reversed, meaning that the output was seen as input and transmitted to input layer using same weights. I used mean square loss function in the reverse direction and then the parameters were updated again by same way as forward direction. After 50 epochs, training the network work in forwarding direction again and so on.

2.2 Convolution Neural Network (CNN)

I selected 80% of dataset as training data and the remaining data was used to test the model. The original images were preprocessed and input to the CNN model for feature extraction and classification. I used stochastic gradient descent method to converge the loss function and backpropagation for updating parameters. After every 5 epochs, the test images were imported into the network to test the model performance. In order to observe the model convergence process, I plotted a graph to show the change of training accuracy and testing accuracy. Furthermore, I also calculated the confusion matrix of testing data which contains precision, recall, and f1-score and stored them into a file for evaluating the model. In the following section, I would discuss the details of my technic and investigation.

2.2.1 Data Preprocessing

In the process of preprocessing the image, I was committed to enhancing the image features so that the subsequent extraction of the image would be more accurate. First of all, the given dataset contains all images with width of 720 and height of 576, although convolutional layers have no size restrictions for input images, I resized the images to the size of 299x299, which is more suitable for solving image classification problem and the comparison of images is displayed in Figure 2 (a) and (b).

It can be seen in the Figure 1 (a) and Figure 2 (a), the gray level of the image is concentrated in the low brightness range and the original image looks dark overall so that I applied histogram equalization to increase the contrast of the image to

enhance image. The histogram of the grayscale distribution of the equalized image is shown in Figure 1 (b), where the distribution of pixel values is more uniform. The image after processing is displayed in Figure 2 (c), the characteristics of which are more obvious visually and easy to be detected and extracted.

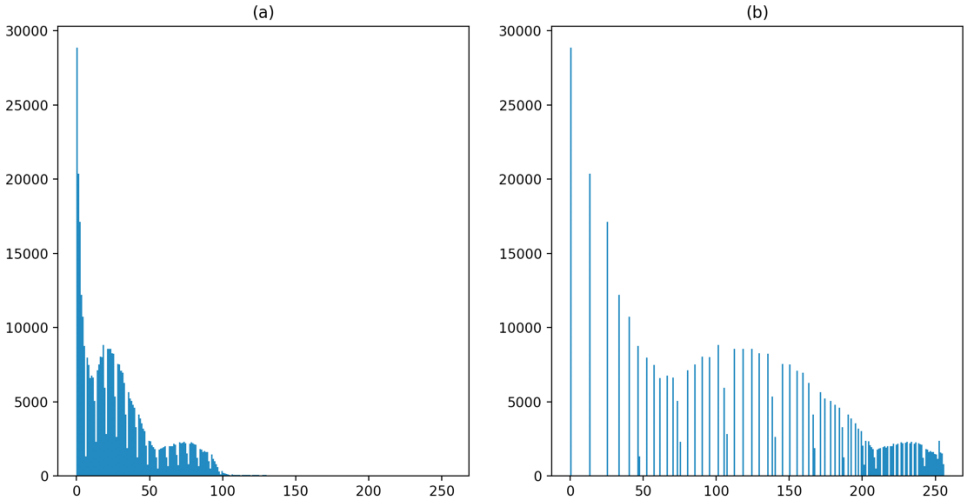

Fig. 1. Pixel distribution comparison. (a) Grayscale histogram of original image, (b) Grayscale histogram of the image after histogram equalization.

In addition, the image is further normalized before being imported into the network, meaning to convert UINT type image data that is 0 to 255 to the range of -1 to 1. In this case, images data is converted into standard mode to prevent the influence of affine transformation and speed up convergence. The comparison before and after image normalization can be seen from Figure 2 (c) and (d), illustrating normalization would not change the information of the image itself so that the results of feature extraction would not be influenced.

Fig. 2. Image processing. (a) the original image (720x576), (b) resized image (299x299), (c) after applying histogram equalization (d). after normalization.

Finally, I assigned images with labels according to the folder they belong to using numbers from 0 to 6 as labels to code the different expressions. So far, all processed image data and labels are stored in the form of arrays. In order to effectively train and test the network, I shuffled the dataset and split the it into a training set and a test set at a ratio of 8 to 2.

2.2.2 Network Architecture

In general, there are five kinds of 5 hierarchical structure in a convolutional neural network, including input layer, convolution layer, pooling layer, full connected layer, and output layer such like Figure 3, which shows a simple convolutional neural network containing the 5 layers. I built up a CNN that is composed of 3 convolution layers, 3 max-pooling layers, and a full-connected network with 3 hidden layers.

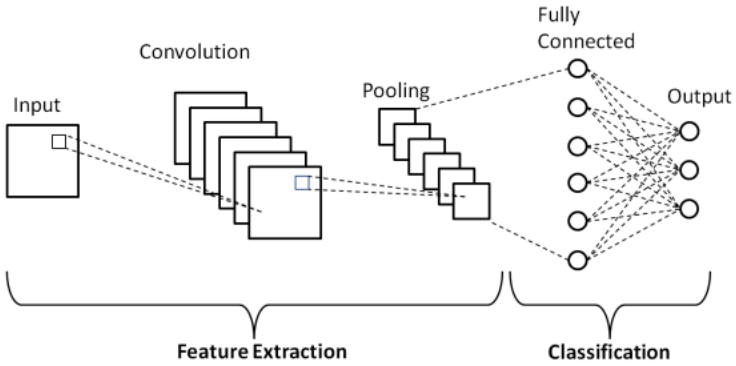

Fig. 3. A simple CNN architecture [9]

Convolutional layers of different levels are dedicated to extracting different information from the image through the principle of parameter sharing. For example, the first convolutional layer of the CNN mainly extracts the edges of the image. As the level of convolutional networks increases, more abstract information of the image will be extracted, which may indicate some common features. Every time the image is transferred to a layer of the network, some noise of the image would be removed so that the network can obtain pure common information represented by the image [10].

The purpose of the pooling layer is to ignore changes in the relative position of the target to improve accuracy of feature extraction, such as tilt and rotation. It can also reduce the dimension of the feature set and avoid overfitting to a certain extent. Finally, fully connected layer plays a classification role in the CNN, which map all extracted features distributed to the labels.

Therefore, I alternately connected three convolutional layers and three pooling layers and input the final extraction result into a fully connected network with three hidden layers. The whole CNN architecture is displayed in Figure 4, where the role and size of each layer of the network are marked.

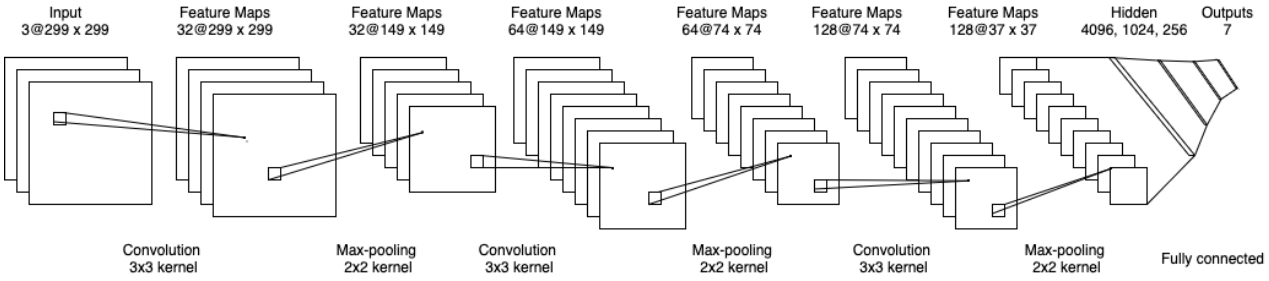

Fig. 4. CNN architecture for facial expression classification

Between the convolutional layer and pooling layer, there is a batch normalization layer to normalize the data of each channel, so that to prevent the network performance from unstable due to excessive data. In addition, I also added an activation layer to suppress negative values to achieve a nonlinear effect. Therefore, the data flow structure from input to output is shown in Figure 5.

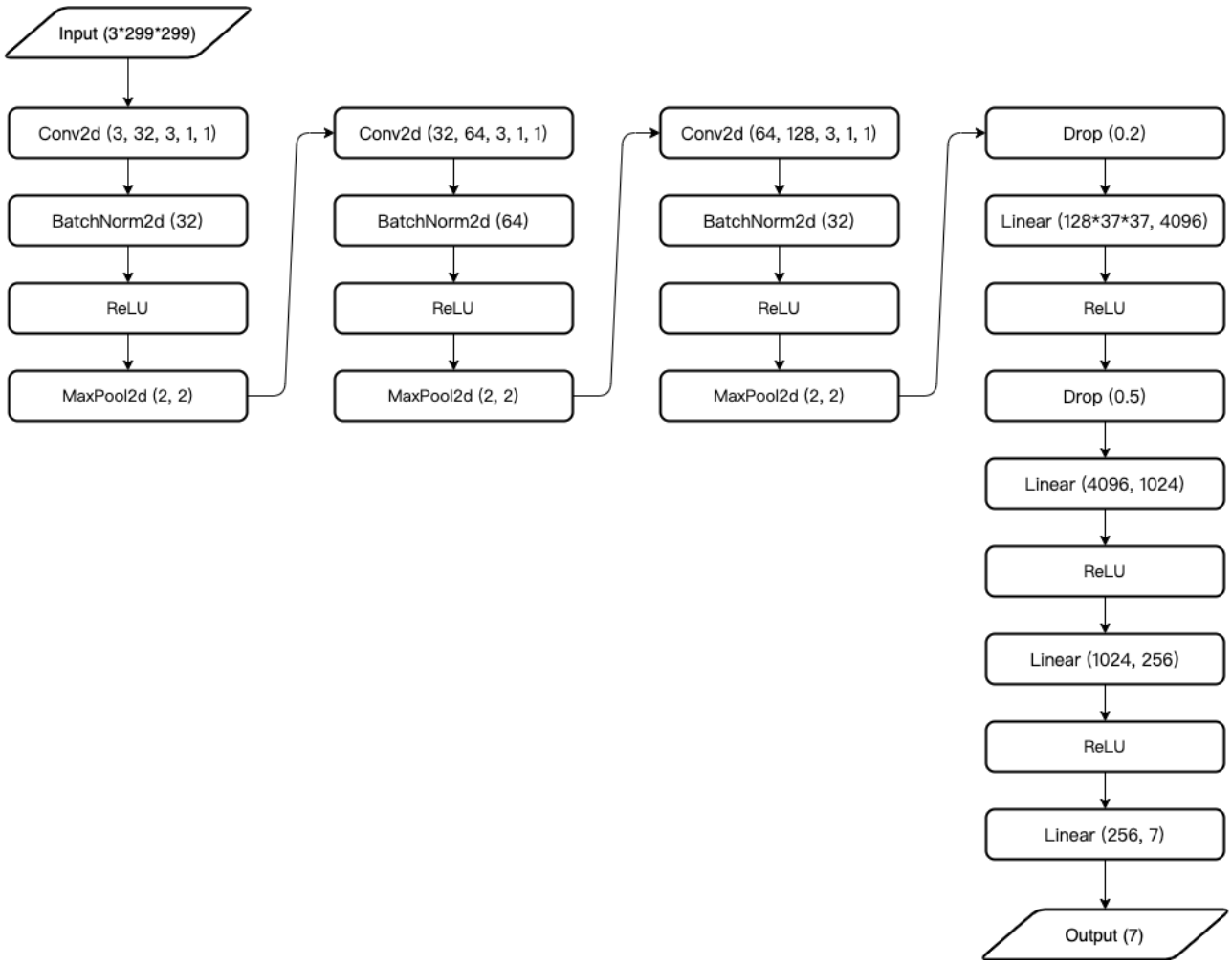

Fig. 5. Data flow in CNN with parameter values in brackets, e.g. Conv2d(32, 64, 3, 1, 1) means in_channels = 32, out_channels = 64, kernel_size = 3, stride = 1, padding = 1.

2.2.3 Training and Testing

For training the network, I shuffled the dataset and split 80% of the data as the training set and forwarded them, at the same time, using the learning rate of 0.05 and weight decay of 1e-8 to prevent overfit to an extent. Every time loading 16 images to predict their labels and using cross entropy loss function to calculate the loss. I also used the back-propagation algorithm to automatically update the model parameters so as to converge loss. After every 5 epochs, I used test data to evaluate the performance of the network, the overall procedure of model training and testing can be seen as follows:

Algorithm 1: CNN training and testing procedure

Input: dataset, batch size, epochs, learning_rate=0.05, weight_decay=1e-8
Output: training loss and accuracy after each epoch, classification report after every 5 epochs, model accuracy plot, a file stores all test records

```
shuffle and random split the dataset by 8 to 2 into train_data and test_data;
initialize model cnn_model = FaceCNN() ;
initialize loss function loss_fc = nn.CrossEntropyLoss() ;
initialize stochastic gradient descent optimizer = opt.SGD() ;
for e = 0 to 50 do
    all_loss = [];
    all_acc = [];
    for imgs, labels in train_data do
        optimizer.zero_grad();
        out = cnn_model.forward(imgs);
        loss = loss_fc(out, labels);
        pred = max(out, 1);
        all_acc.append((pred == labels) / len(labels));
        all_loss.append(loss.item());
        loss.backward();
        optimizer.step();
    end
    avg_loss = sum(all_loss) / len(all_loss);
    avg_acc = sum(all_acc) / len(all_acc);
    print avg_loss and avg_acc;
    if (e + 1) % 5 is 0 then
        actual = [];
        predict = [];
        for test_imgs, test_labels in test_data do
            pred = model.forward(test_imgs);
            predict.extend(pred);
            actual.extend(test_labels);
        end
        cr = classification_report(actual, predict);
        acc = accuracy_score(actual, predict) print cr;
        open file 'evaluation.txt';
        write cr, acc to 'evaluation.txt';
    end
    plot accuracy graph
end
```

2.2.4 Evaluation Methods

In general, accuracy, precision, and recall can be used as criteria for evaluating a network model, which can be calculated by tp, fp, tn, fn as follows:

$$\text{Accuracy} = \frac{tp+tn}{tp+fp+fn+tn}$$

$$\text{Precision} = \frac{tp}{tp+fp}$$

$$\text{Recall} = \frac{tp}{tp+fn}$$

Where *tf* is true positive, meaning the correct number in the samples determined as positive, *fp* is false positive, meaning the number of errors in the sample determined to be positive, *tn* is true negative, meaning the correct number in the sample judged as negative, and *fn* is false negative, meaning the number of errors in the sample judged as negative. I calculated all three statistics criteria and compared the results with the BDNN model that I implemented before and an existing research to evaluate the performance of CNN model.

3 Results and Discussion

After training and testing the BDNN model, the accuracy only reached an undesirable rate of 16.14%. The main classification indicators are shown in Table 1, where there are missing data of some labels.

Table 1. Report of main classification indicators of BDNN.

Emotion	Precision	Recall
Angry	0.11	0.15
Disgust	0	0
Fear	0.13	0.29
Happy	0.16	0.26
Neutral	0.27	0.21
Sad	0.14	0.18
Surprise	0	0

On the opposite, the performance of CNN model is much better. To evaluate this model, I will discuss in three aspects, including the accuracy change during training the network, the classification result, and the comparison with other networks. First of all, the accuracy change of the training set and testing set during 50 epochs can be observed from Figure 6, which have been rising overall and begin to stabilize after about 34 epochs. At the end, when the train accuracy is near to almost 100%, the test accuracy can up to 49.26%.

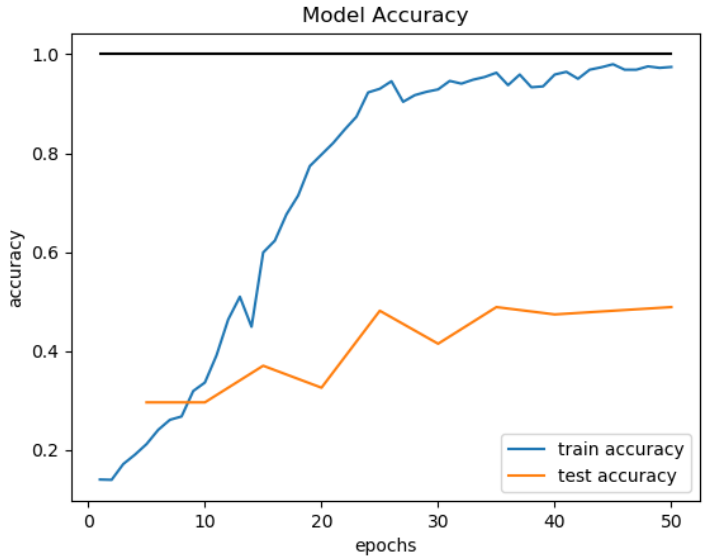

Fig. 6. The change of accuracy of training set and testing set over 50 epochs.

In addition, there are 135 samples for testing and the CNN model aims to classify them into seven classes and the number of samples assigned to each label is shown in Table 2. The Table 3 shows the indicator data used to analyze the classification results, including the accuracy rate, recall rate, and F1 value of different categories. It can be observed that images with fear expressions are the easiest to classify correctly, but images with neutral expressions are the opposite, with the lowest recognition precision.

Table 2. Statistics of model classification results of CNN.

Emotion	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Label	0	1	2	3	4	5	6
Count	23	11	22	23	14	21	19

Table 3. Report of main classification indicators of CNN.

Emotion	Precision	Recall	F-score
Angry	0.62	0.43	0.51
Disgust	0.36	0.45	0.40
Fear	0.63	0.77	0.69
Happy	0.45	0.43	0.44
Neutral	0.27	0.21	0.24
Sad	0.50	0.57	0.53
Surprise	0.42	0.42	0.42

Finally, compared with the BDNN model which I implemented for facial expressions classification based on a different format of dataset with the accuracy less than 20%, this CNN model performs much better on solving the same problem with the accuracy of 49.26%. However, the dataset for training BDNN is the result of the original data after dimensionality reduction processing, which removed information of original images to some extent so that the dataset is in a small scale, which allows the network to be quickly trained and converged. On the contrary, the convolutional neural network directly acts on the image data, and the large data size slows down the training speed. Furthermore, there is an existing research which used an extension of local binary patterns (LPQ) and pyramid of histogram of oriented gradients (PHOG) to classify facial expressions based on the same database and the accuracy of such model is 43.71% and 46.28% respectively [4]. It is obvious that in terms of the accuracy of the classification results, the CNN model I developed has outstanding performance.

4 Conclusion and Future Work

In this paper, I implemented a bidirectional neural network and a convolutional neural network to classify the images in SFEW dataset into seven facial expressions and obtained accuracy rates of 16.14 and 49.34% respectively. Furthermore, I analyzed and compared the classification results of both two models, drawing a conclusion that the performance of CNN is much better. It is not suitable to apply BDNN model to solve some issues because of the limitation of this model.

Although I improved the performance of the model by adjusting the parameters of the CNN model, due to time and hardware limitation, I have no opportunity to observe the impact of different numbers of convolutional layers on the classification results. So that I would like to try to change part of the architecture of the CNN in the future to whether the performance of the model will improve. In addition, there are 682 images in the subset of SFEW, although the training results are acceptable, it is necessary to sample more data for training the model. Therefore, I will use larger scale of data to train the model so as to improve its performance. I will also work on apply this CNN model in different dataset that contains different format of images to improve the applicability and universality of the network.

References

1. Hu, T., L.C. De Silva, and K. Sengupta, *A hybrid approach of NN and HMM for facial emotion classification*. Pattern Recognition Letters, 2002. **23**(11): p. 1303-1310.
2. Pantic, M., et al. *Web-based database for facial expression analysis*. in *2005 IEEE International Conference on Multimedia and Expo*. 2005.
3. Cohen, I., et al., *Facial expression recognition from video sequences: temporal and static modeling*. Computer Vision and Image Understanding, 2003. **91**(1): p. 160-187.
4. Dhall, A., et al., *Emotion recognition in the wild challenge (EmotiW) challenge and workshop summary*, in *Proceedings of the 15th ACM on International conference on multimodal interaction*. 2013, Association for Computing Machinery: Sydney, Australia. p. 371–372.
5. Graves, A., S. Fernández, and J. Schmidhuber. *Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition*. in *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005*. 2005. Berlin, Heidelberg: Springer Berlin Heidelberg.
6. Nejad, A.F. and T.D. Gedeon. *Bidirectional neural networks and class prototypes*. in *Proceedings of ICNN'95 - International Conference on Neural Networks*. 1995.
7. Krizhevsky, A., I. Sutskever, and G.E. Hinton, *ImageNet classification with deep convolutional neural networks*, in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. 2012, Curran Associates Inc.: Lake Tahoe, Nevada. p. 1097–1105.
8. Ji, S., et al., *3D Convolutional Neural Networks for Human Action Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013. **35**(1): p. 221-231.
9. Phung and Rhee, *A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets*. Applied Sciences, 2019. **9**: p. 4500.
10. Pinheiro, P.O. and R. Collobert. *From image-level to pixel-level labeling with Convolutional Networks*. in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.