Classification face emotion with SFEW by Neural Network and improved by adjust threshold and compare with traditional neural network

Yanlong LI

Research School of Computer Science, Australian National University, Canberra, Australia
U5890571@anu.edu.au

Abstract. In recent years, with the rapid rise of artificial intelligence technology and the substantial improvement of computing technology, people have put forward higher requirements for human-computer interaction tasks. At the same time, in person-to-person communication, not only language signs, emotion and other body language are also components of conveying information. Facial emotion recognition technology refers to the extraction of human facial emotion from static images or dynamic videos, which can further identify the psychological emotions of objects. In this article, we will be using the artificial intelligent technique and train an artificial neural network model to recognize emotion by given SFEW library with real face picture captured from movies. While training our model, we found it is difficult to build a perfect model that library hence it is from the real world rather than ideal images from the lab. As a result, after we preprocessed images and tuned our model. We get around 50% accuracy rate on train set and 25% on evaluating set.

Keywords: Neural network, Face emotion recognize, image process

1 Introduction

Face emotion recognize is vital in the AI area in recent years and robotics in the future. Face emotion can be categorized and help computer/robot understand the human being at a high level.

According to the research from paper [1] has shown that tuning the threshold can be effect the model and results dramatically and find the correct threshold can significantly improve the result. In this paper, I will establish a fixed bias CNN network to analyze the real images in the SFEW library and also trained a traditional neural network for comparison.

2 Data

2.1 Data

In this research, I am using the SFEW library, which contains 675 images and has been labelled [3]. The images are directly captured from movies with size 720 × 576. Images are labelled in seven labels: Angry, Disgust, Fear, Happy, Neutral, Sad and Surprise. For the easy to read by the computer, I use 0 – 6 to represent those labels. For the dataset, all images in this library are split into 2 sets randomly by radio 8:2. 80% data is used for train and 20% data is test set.

2.2 Preprocessing data

In this research, I am using a group of images with real faces from movies. Thus, there are lots of noise in each image. Such as background, cars, other people in an image but not facing the camera. Moreover, the target faces usually are small, not directly facing and assimilate into the background in some time. Hence, our model is very difficult to abstract features for images without preprocessing.

For this research, I extract the faces from images by using OpenCV and keep the face only. This step remove all irrelevant object in the image. Thus, the noise of each image can be removed. In the library, faces have 3 directions: forward, left forward and right forward and by the limitation of OpenCV which only able to recognize forward and right face[4]. Hence, I split three steps to extract. Firstly, extract all face which forward. Secondly, extract all face which right forward. Thirdly, flip all left forwarded face then extract as right forward.

After extract face from images, I shrink the image size to 48 × 48. Because almost faces in the library after extract is small. Hence it is unnecessary to keep the image big, also shrink the image size also increased training speed. Finally, images will be converted into a grey image which only one channel.

2.3 Data for comparison

As a comparison, I used the same library, but only extract the two main features: LPQ and PHOG. LPQ stand for local phase quantization, which t is a novel texture descriptor that makes use of the invariance of the phase spectrum [5]. PHOG stand for pyramid histogram of the oriented gradient, which are calculated features at different scales for a fixed size image [6].

The data set contains the NaN data which lost the entire PHOG features. To improve the training accuracy and reduce the impact by bad data simple, I remove the lines

which contain the NaN data. Due to only one line is bad in the data set. There is no loss accuracy or negative impact by remove data in the set.

To reduce the complexity for train and the impacts from data, I normalized the data set by using min-max normalization technique. Min-max normalization transform each feature in range 0 to 1 by using the formula below [7].

$$x' = \frac{x - min}{max\ -\ min}$$

The x' is the new value for each feature, x is original and max, min are the maximum and minimum value for each data.

3 Neural Network

3.1 CNN

For this dataset, I used the Convolutional Neural Network (CNN) as a model to train. At convolutional part, that models take an image with $48 \times 48 \times 1$, which means height 48 px, weight 48 px and 1 channel. All images will pass a kernel with size $3 \times 3 \times 64$, 1 padding and 1 stride then pooling by 2×2. After that, the model will get a feature map FM_1 with size $24 \times 24 \times 64$. Then put FM_1 to a kernel with size $3 \times 3 \times 128$ then pooling by 2×2. Then get another feature map FM_2 with size $12 \times 12 \times 128$. Put FM_2 into a $3 \times 3 \times 256$ sized kernel with 2×2 pooling to get a new feature map with FM_3 size $6 \times 6 \times 256$.

After the convolutional layer, flat the FM_3 to one-dimension 9216 long tensor. Then put that tensor into a fully connected layer. The fully connected layer contains 3 hidden layers of neural. Firstly, the tensor will be dropout then into the first hidden layer. The first hidden layer contains 4096 neural. Secondly, the output of the first layer will be dropout then into the second hidden layer which 1024 neural. Thirdly, the output of the second hidden layer will into the third hidden layer which contains 256 neural. Finally, the output from the third hidden layer will into the output layer which has 7 different results related to the labels for images. The image of the structure shown below Fig. 1.

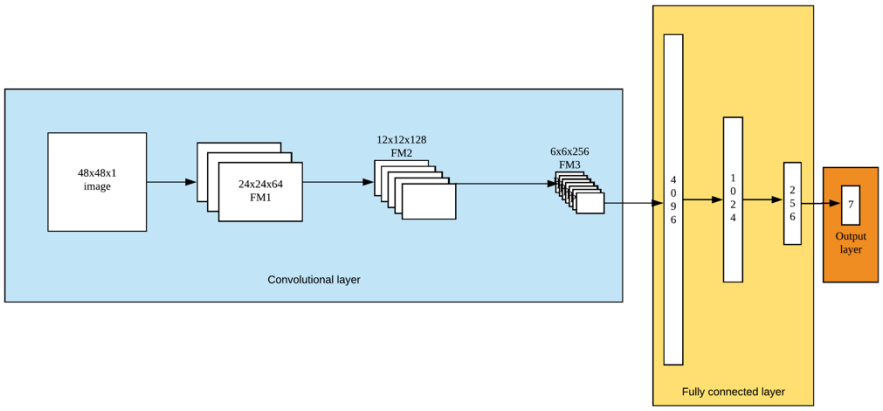

Fig. 1. The structure of CNN model.

3.2 Traditional Neural Network

For the classifying features to face emotions, I defined a feed-forward neural network with 3 layers, include input layer, hidden layer and output layer. Each layer of neural are fully connect with direct next layer. In the neural network, hidden layer using sigmoid as the activation function. Use MSE as loss function and Adam as the optimizer. The structure of the neural network is shown below in Fig 2.

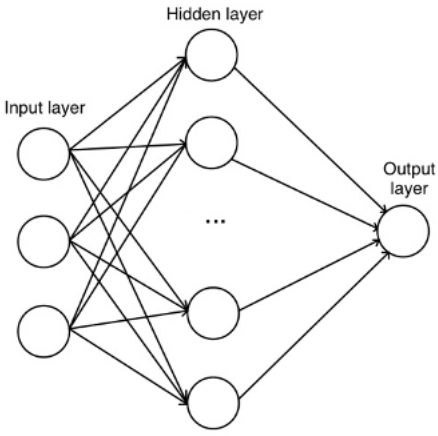

Fig. 2. The structure of traditional neural network

3.3 Optimizer, Loss function and activation function

In the CNN model, I use the RReLU as the activation function. RReLU is the randomized leaky ReLU. The function is defined as:

$$RReLU(x) = \begin{cases} x & if\ x \geq 0 \\ ax & otherwise \end{cases}$$

The a is picked randomly from uniform distribute (lower bound, upper bound)[8,9]. Pick RReLU for CNN activation function has the following reason: 1. Compare with sigmoid it will not affect by backwards. 2. The border is wider.

In the traditional model, I use sigmoid as the activation function. Because that model is simple with only a few layers, the dataset has been normalized all value between 0 and 1 and the range of sigmoid is 0 to 1, that is easy to calculate.

In the CNN model, I use the CrossEntropyLoss as the loss function. In the PyTorch, softmax calculation has been intergraded into CrossEntropyLoss. Hence, we do not need an extra softmax layer.

In the traditional model, the loss function is MSELoss. Because the punishment is heavier when the loss bigger.

The optimizer for CNN is SGD. It finding the optimal for the target. The optimizer for the traditional neural network is Adam. Because it is easier to calculate and it is not affected by the unstable target function.

4 Result and evaluation

4.1 Traditional neural network

Improving performance can be split into 2 parts. The first part is data preprocess which discussed above. Secondly is tune the hyperparameters for our machine learning model. In our model, hyperparameters are learning rate, number of neural in the hidden layer and the number epochs.

For the number epochs, we can see the Fig. 3, the loss turns to stable since 2000 epochs.

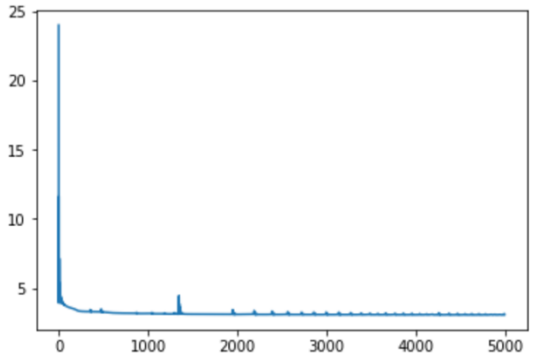

Fig. 3. The loss rate

To adjust the learning rate and the number of neural in the hidden layer can be based on the accuracy rate. The relation between the learning rate and the accuracy rate in our model is shown in Fig. 4 below. Based on the result, we can see the accuracy rate decreased at the beginning and increased to the highest when the learning rate is 0.4 then decreased.

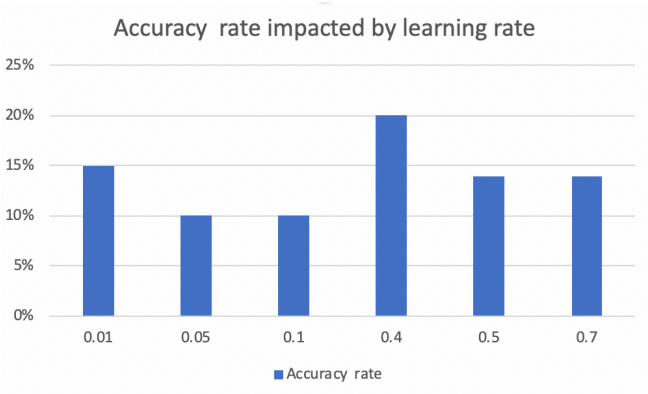

Fig. 4. Accuracy rate impacted by learning rate

For the accuracy rate impacted by the number of neural in the hidden layer are shown in Fig. 5 below. We can see that result is like the impact by learning rate above. When the number of neural in the hidden layer is 15, the accuracy rate is the highest.

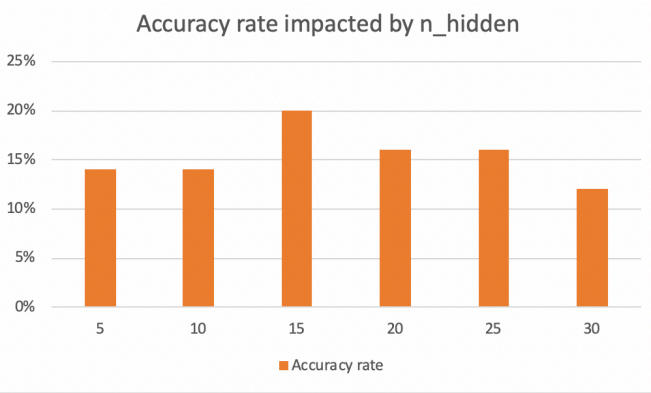

Fig. 5. Accuracy rate impacted by the number neural in hidden layer

Overall, the best in this model is shown in **Table 1**.

Table 1.

	Accuracy rate (train set)	Accuracy rate (test set)
traditional neural network	30%	20%

4.2 CNN

According to the paper[1]. The result will be changed by adjust threshold and find a good threshold can get a better result. In my model, I find the result is acceptable when the threshold is 0.2. Thus, I choose 0.2 as the baseline and try multiple values of the threshold. The results are shown in **Table 2**.

Table 2.

θ	Accuracy rate (train set)	Accuracy rate (test set)
0.10	52%	25%
0.20	55%	30%
0.25	43%	25%
0.30	37%	20%
0.40	36%	23%

Overall, we can see when the threshold is 0.2, the result for both the train set and evaluate set are best, and the accuracy rate is decreased on both set when threshold increase or decrease.

4.3 Compare traditional neural network and CNN

As a result, the best result of CNN is 55% correct on train set and 30% correct on the test set. Compare with the traditional neural network, the CNN model performed dramatically better than a traditional neural network. The main reason for that result is a more advanced model, the dataset is preprocessed before training.

5 Conclusion and future work

I had used the SFEW dataset to classifying face emotion by neural network techniques, which contains 675 images captured from movies. I also compared that model with a traditional neural network which classifying face emotion for the same dataset with 5 LPQ features and 5 PHOG features.

As a result, Our model preformed the 55% accuracy rate on train set and 30% accuracy rate on the test set. That is significantly better than the traditional neural network. However, it still can be improved by increase the quality and quantity of dataset by removing the irrelevant object in each image, make the face easier to recognize and increase the size of the dataset.

Reference

[1] L.K. Milne, T.D. Gedeon and A.K. Skidmore, “CLASSIFYING DRY SCLEROPHYLL FOREST FROM AUGMENTED SATELLITE DATA: COMPARING NEURAL NETWORK, DECISION TREE & MAXIMUM LIKELIHOOD”

[2] Dhall, A., Goecke, R., Lucey, S., & Gedeon, T. (2011, November). Static facial expressions in tough conditions: Data, evaluation protocol and benchmark. In 1st IEEE International Workshop on Benchmarking Facial Image Analysis Technologies BeFIT, ICCV2011.

[3] Goecke, R., “Static Facial Expressions in Wild” [Online]. Available: https://researchdata.ands.org.au/static-facial-expressions-wild-sfew/2729 [Accessed May 2020].

[4] Stephen, M., Richard, B. (2010). Multi-View Pose and Facial Expression Recognition. British Machine Vision Conference, BMVC 2010 - Proceedings.

[5] Ojala, T., Pietikainen, M., Maenpaa T. (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. Pattern Anal Mach Intell IEEE Trans 24(7):971–987

[6] Bosch, A. , Zisserman, A. and Munoz, X. Representing shape with a spatial pyramid kernel Proceedings of the International Conference on Image and Video Retrieval (2007)

[7] S. Patro, K. K. Sahu, "Normalization: A preprocessing stage," arXiv preprint arXiv:1503.06462, 2015.

[8] Pytorch., “RReLU” [online]. Available: https://pytorch.org/docs/stable/nn.html#rrelu [Accessed May 2020].

[9] Xu, B., Wang, N., Chen, T., & Li, M. “Empirical Evaluation of Rectified Activations in Convolutional Network”