CSE514 Data Mining: Final Report of Human Emotion Classification

Team member
Zhibo Liang: ID: 467574 Email: zliangaf@gmail.com
Zubin Zhang: ID: 466774 Email: zhangzubin123@outlook.com

1. Introduction

Problem Description

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. The project is divided into five parts:
(1) Store the data in database.
(2) Use data points to generate features for each picture
(3) Train classifier
(4) Test its accuracy
(5) Make data visualization

The state of the art in addressing the problem

Facial expressions are visually observable non-verbal communication signals that occur in response to a person's emotions and originate by the change in facial muscle. They are the key mechanism for conveying and understanding emotions. Ekman and Freisen [1] postulated six universal emotions with distinct content and unique facial expression, which are anger, fear, disgust joy surprise and sadness. Much of the efforts have been made to classify facial expression with various facial feature by using machine learning algorithms. For example, Anderson et al. [2] developed a FER system to recognize the six emotions. He uses SVM and Multilayer Perceptron and achieved a recognition accuracy of 81.82%. Lia et al. [3] used k-nearest neighbor to compare the performance of PCA and NMF on Taiwanese and Indian facial expression databases. They attained above 75% recognition rate by using both techniques.

The method we use

In the human emotion classification project, we use four methods. They are Random Forest, Support Vector Machine (SVM), CNN (Convolutional Neural Networks) and Autoencoder+SVM.

2. The existing methods and related work

Existing method of human emotion classification

There are many methods that can be used in human emotion classification, like Convolutional Neural Networks (CNN), Bayesian Network (BN), rule-based classifiers, Support Vector Machine (SVM), Adaboost and Random Forest. Hidden Markov Models (HMM) has also been applied to facial expression recognition as one of the most popular classifiers among spatio-temporal methods.

Related work

Recently, Xing et al. [4] used local Gabor features with Adaboost classifier for FER and achieved 95.1% accuracy with the 10-time reduced dimensionality of traditional Gabor features.Tang [5] reported a deep CNN jointly learned with a linear support vector machine (SVM) output. His method achieved the first place on both public (validation) and private data on the FER-2013 Challenge. Liu et al. [6] proposed a facial expression recognition framework with 3D-CNN and deformable action parts constraints in order to jointly localizing facial action parts and learning part-based representations for expression recognition. In 2011 Xufen Jiang made research in HMM based facial expression recognition, in which he proposed a new method for facial recognition. Lu Tai, Pu Xiaorong, Tan Heng and Zhou Zhihu, published a paper in that they proposed an HMM for partially hidden face recognition. Wang et al. [7] combined HOG and WLD features to have missing information about the contour and shape. The proposed solution attained 95.86% recognition rate by using chi-square distance and the nearest neighbor method to classify the fused features.

3. Experiment setup and the data used

Experiment

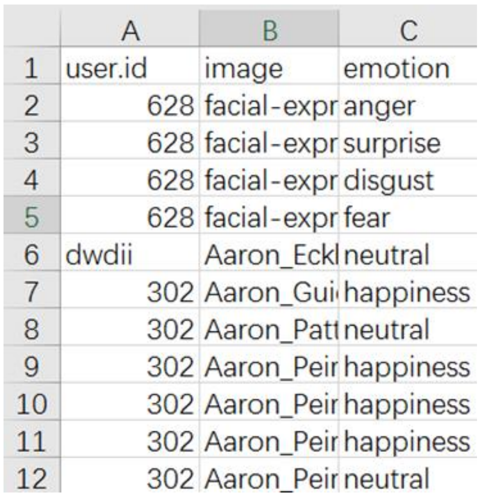
Language: python 3.6

After discussion with professor, we decide to use Keras and sklearn to help us build the classification models. Sklearn is used to build SVM and random forest. Keras is build based on TensorFlow and it is used to build auto-encoder and CNN.

We will also use OpenCV and Dlib to do feature selection and preprocessing. Matplotlib and tensor-board is used to build data visualization.

Data collection

We will use an image database with total 13690 different images. The images are divided into three different sets: training set, validation set and test set. These percentages may change according to our actual training result. Besides, we have a csv file named label. It includes the user.id, the name of images and the kind of emotion in images. Here is the screenshot of our csv file:



	A	B	C
1	user.id	image	emotion
2	628	facial-expr	anger
3	628	facial-expr	surprise
4	628	facial-expr	disgust
5	628	facial-expr	fear
6	dwdii	Aaron_Eckl	neutral
7	302	Aaron_Gui	happiness
8	302	Aaron_Patt	neutral
9	302	Aaron_Peir	happiness
10	302	Aaron_Peir	happiness
11	302	Aaron_Peir	happiness
12	302	Aaron_Peir	neutral

Figure 1: the screenshot of our csv file

We divide 7 kinds of emotion, which is surprise, happiness, neutral, and anger, disgust, fear, sadness. After the training, we use validation set to optimize the parameter of each model. 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. We will compute the overall training accuracy and testing accuracy.

Data processing

First, we set the number of training, validation and test data. Then, in order to prepare the training, validation and test data, we construct the database for model training. We resize all the detected faces into 350 X 350, load images and convert it to grey scale. After reading image name and label from the label.csv, we do the randomized perturbation in order to better overall accuracy.

The reason we need to perturbate is below:

As we checked the emotion label in label.csv. We have found the number of each emotion is that:

emotion	number
surprise	368
happiness	5696
neutral	6868
anger	257
disgust	208
fear	21
sadness	268

Form 1 The number of different emotions in label.csv

You can see that the number of “happiness” and “neutral” emotion is significantly more than other emotions. Therefore, the database in unbalanced, which does bad to overall accuracy. Our object is to make the number of seven emotion images are all the same. So, we need perturbate to increase the number of images in other six emotions. The step is that:

(1) Get the number of images with each specific emotion.

(2) Compare and get the emotion label which has the greatest number of images.

(3) Increase the number of images in other six emotion labels. The method we use is rotation. We set a random number representing the rotation degree from -45 to +45, and let each image belonging to other six emotion labels to rotate the random degree until the number of seven emotion images are the same.

After we rotate, we also use Horizontal mirror flip and GaussianBlur in order to increase the overall accuracy. We set a random number from 0 to 1. If the number is larger than 0.5, the image will have its horizontal mirror flip partner or GaussianBlur partner.

The comparing criteria of each method

(1) The testing accuracy (Standard: training number is 11000, testing number is 2000).

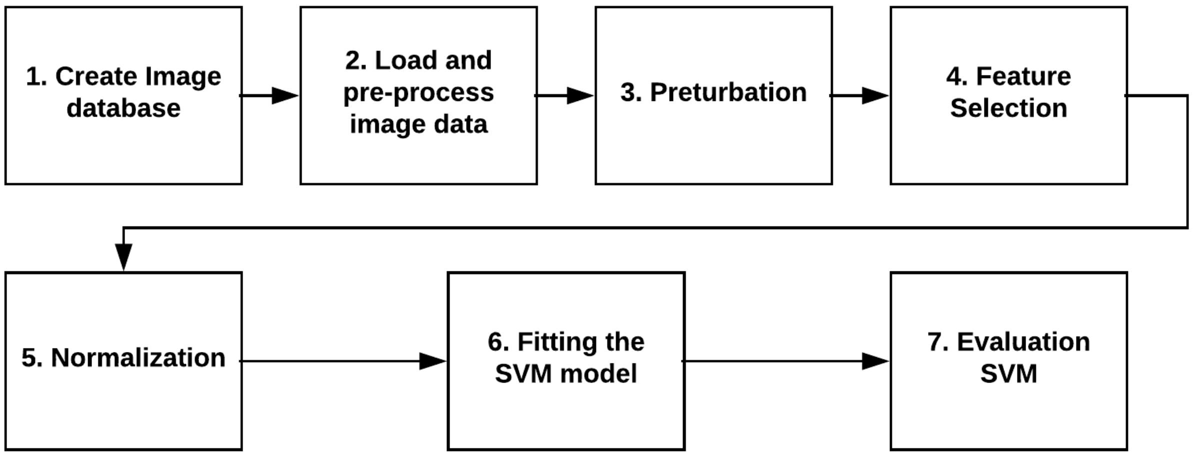
(2) The time to get the outcome of overall accuracy

(3) The problem of overfitting

4. Methods description and results

1) Support vector machine (SVM)

We use support vector machine to build our first emotion classification model. We use sklearn to build our SVM model, which is a very powerful machine learning method. The procedure of building an SVM emotion classification is as following:



Step 1: Create Image Database

Our image database has nearly 13000 images with 7 different emotion labels. The name and the label of image is stored in a file named “label.csv”. In this step, we separate the image into two different folders. One folder is used to store training dataset and the other is used to store test dataset. We copy 1000 images into training dataset folder and 500 images into test dataset folder.

Step 2: Load and pre-processing image data

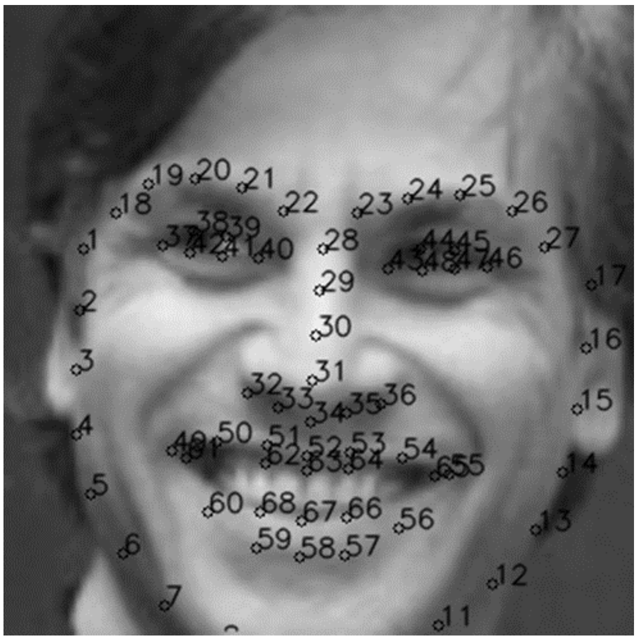
In this step, we load the image in training and testing folder and do some preprocessing to them. OpenCV is used to do these works. All the pictures are resized to 350 * 350 and convert to grey scale image.

Step 3: Perturbation

After some study on our dataset, we found the dataset we use is imbalance. There are more picture with label ‘Happiness’ and ‘Neural’ and less with labels such as ‘Disgusted’ and ‘Surprise’. So, we perform the perturbation [1], which means that we produce new images based on the current images we have to make our database balanced. We use image rotation (between -45 degree to 45 degree), image flipping, and image blurring to produce image. After this step, the number of image in training dataset becomes nearly 3500 and the number of testing image becomes nearly 2100.

Step 4: Feature Selection

Now, we have a balanced image dataset, we will select feature from these images and these features will be used to train the SVM classifier. This feature selection part is divided into two parts: human face landmarks detection and feature generation. OpenCV and Dlib is used to do landmarks detection. We detect 68 landmarks on human face. The following picture shows the landmarks that we used.



After getting the landmarks, we can use them to generate the feature we want. There are two types of feature: normalized distance between two landmarks and the angle between three landmarks. We use the distance between point 2 and point 16 to perform our distance normalization. We generate 65 different features and 60% is normalized distance feature.

One thing need to point out is that the face landmarks detection program can not recognize some pictures that are produced in perturbation step. For these images, we assign random value to their feature. There are about 3% such images, we believe it is acceptable.

Step 5: Normalization

This is a very tradition data pre-processing step. We normalize our data to make their mean to be 0.

Step 6: Fitting the SVM model

We use sklearn to perform this step. We set gamma to be 'scale' and decision_function_shape to be 'ovo'. For the rest setting, we use the default one. We use nearly 3500 images to train our SVM. The average training time is about one hour.

Step 7: Evaluation SVM model

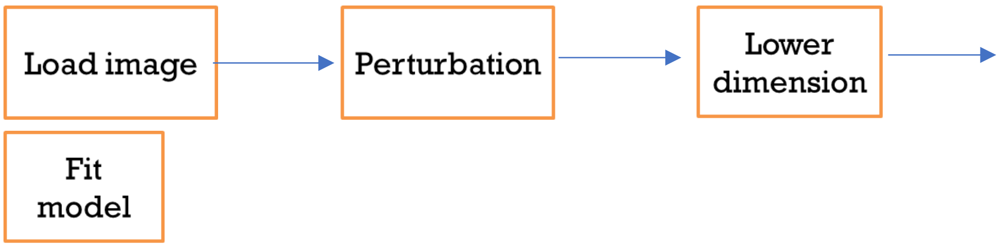
First, we directly use our training and test dataset to evaluate our model. The accuracy on training data is 0.634 and accuracy on test data is 0.505. Sklearn also provide cross validation method. We perform 5-corss validation and the accuracy is [0.57692308, 0.6039604, 0.60606061, 0.6122449, 0.60204082]

2) Random forest:

One of the classification methods we use is random forest.

Random forest is an ensemble learning method for classification, which is an extension to bagging. The difference between random forest and bagging is that random forest uses a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features.

In the human emotion classification project, the step of random forest is that:



Here is the detailed description of each step of building random forest.

1. Load image: The aim of loading image is to let our system read the file including image data and the label of emotion.
2. Perturbation: As we have shown in the data preparation part, the aim of perturbation is to balance our dataset. We would like to make the number of seven emotion images all the same.
3. Lower dimension: Lower dimension is a special part in random forest comparing to other methods we use for the human emotion classification project. We need it to fit the RandomForestRegressor function in the sklearn package. Therefore, we need to transform the dimension of image data from 4 to 2. For example, initially we set the dimension of image data as a four-dimension array (the number of image data, 350, 350, 1). After the step of lower dimension, the dimension of image data is a two-dimension array (the number of image data, 350*350*1=122500).
4. Fit model: After all the steps before, we need to train our dataset and compute its training accuracy and testing accuracy.

The result of random forest method:

Here are two graphs showing the training accuracy with different values of feature selection and testing accuracy with different values of feature selection. (Standard: training number is 11000, testing number is 2000).

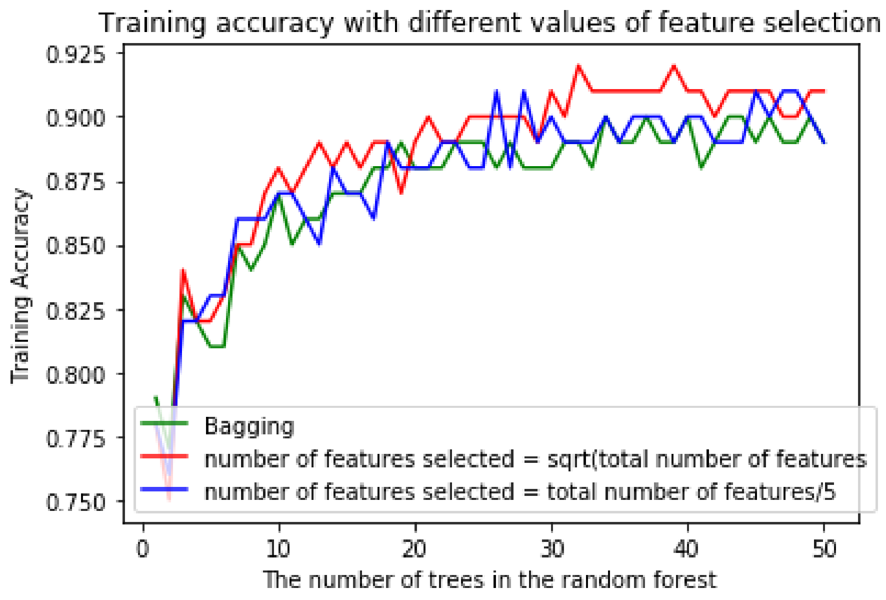


Figure 1 Training accuracy with different values of feature selection

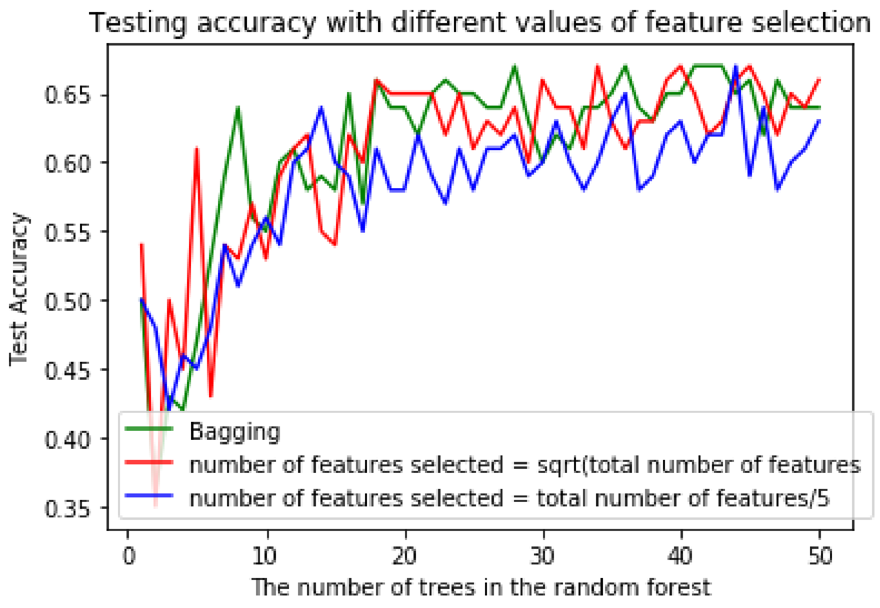


Figure 2 Testing accuracy with different values of feature selection

The analysis of the result in random forest method:

As you can see in these two graphs, the training accuracy and testing accuracy will increase when the number of trees in the random forest increase. For the training accuracy graph, the training accuracy line is nearly parallel when the number of trees in the random forest is over 20. The value of training accuracy is nearly 0.9 when parallel. For the training accuracy graph, the testing accuracy line is kind of dramatic change when the number of trees in the random forest is small. However, the testing accuracy line is nearly parallel when the number of trees in the random forest is big, especially over 20. The value of training accuracy is nearly 0.65 when parallel.

The pros and cons of random forest method in this project:

Pros:

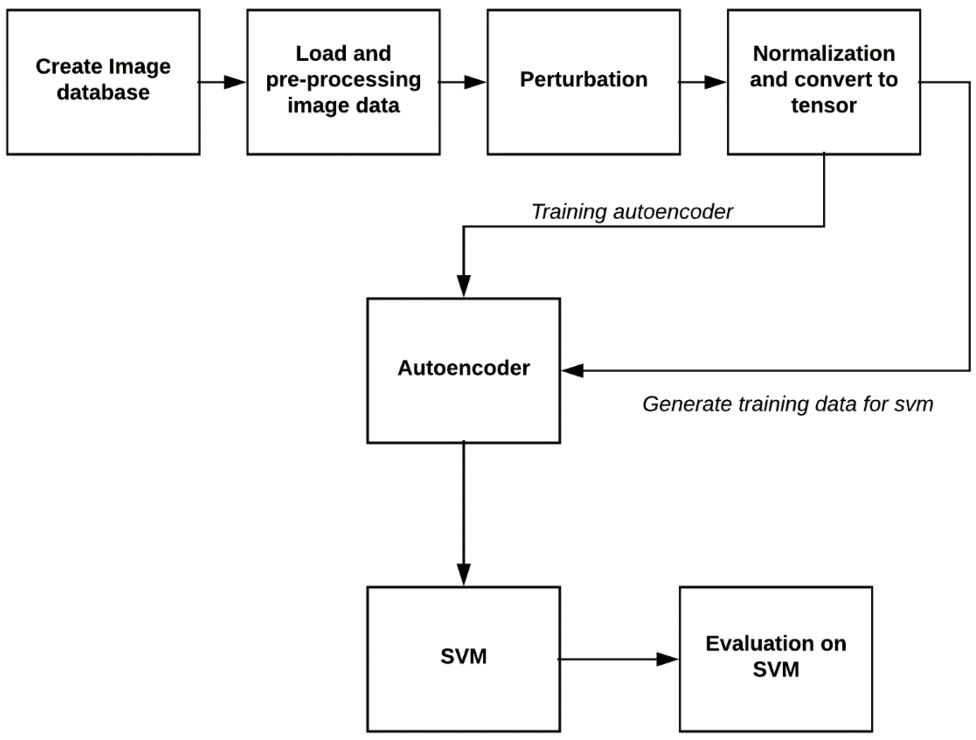
1. It uses less time to get the results of training accuracy and testing accuracy comparing to other method.
2. Random decision forests correct for decision trees' habit of overfitting to their training set.
3. The result visualization of random forest is easy to identify.

Cons:

1. Although correcting the overfitting problem, the overfitting situation is still existing. The difference between training accuracy and testing accuracy is higher than that of other method used in the human emotion classification project. We think the optimization of random forest method is not well enough, so this is one of our future work.

3) Autoencoder + SVM

After using two traditional machine learning method, we want to do something different. The first change we want to make is that we do not want to generate features by ourselves. We


Create Image database
Load and pre-processing image data
Perturbation
Normalization and convert to tensor
Training autoencoder
Autoencoder
Generate training data for svm
SVM
Evaluation on SVM

choose Autoencoder as our feature extraction method because Autoencoder is good at feature extraction, especially on image data. The following is the procedure of building an Autoencoder + SVM emotion classifier.
The first three steps in image data processing is the same as the previous models. After the Perturbation step, we can get a balanced image database suitable for building autoencoder. In normalization step, we make sure all the value data in the image are between 0 and 1 to prevent bias problem.
Then we use the image data to fit autoencoder directly. The encoder part has three layers: one input layer(input size is 350 * 350) and two hidden layers. The hidden layers has 2048 perceptron and the second layer has 128 perceptron. This means that we transfer the 350* 350 image to 128 features through encoder. Since our autoencoder is symmetry structure, the decoder part has similar structure. We use 10000 images to train our autoencoder and the final loss drop from ten thousand to nearly 0.05.
Then we take out the encoder part separately and use it to generate the features of images. Then we use these features to build SVM classifier. The procedure of building SVM is the same as the previous one. Then we use test data to evaluate the performance of SVM. The accuracy on testing data is 0.691, which is bigger than only use SVM(0.505). This means that autoencoder can really help extract features from the image.

4) Convolutional Neural Network

The last model we used for building emotion classifier is Convolutional Neural Network (CNN). As we have known, CNN is very good at dealing with image classification problem and that’s why we choose this method.
We use Keras, a library built based on TensorFlow, to develop our model. Compared to TensorFlow, Keras is easier to program and debug. What’s more, since Keras is built based on TensorFlow, it is as powerful as TensorFlow. We have also used some famous libraries in our project to do data pre-processing, such as Numpy, Pandas, Matplotlib and OpenCV.
The procedure of building a CNN emotion classification is as following:

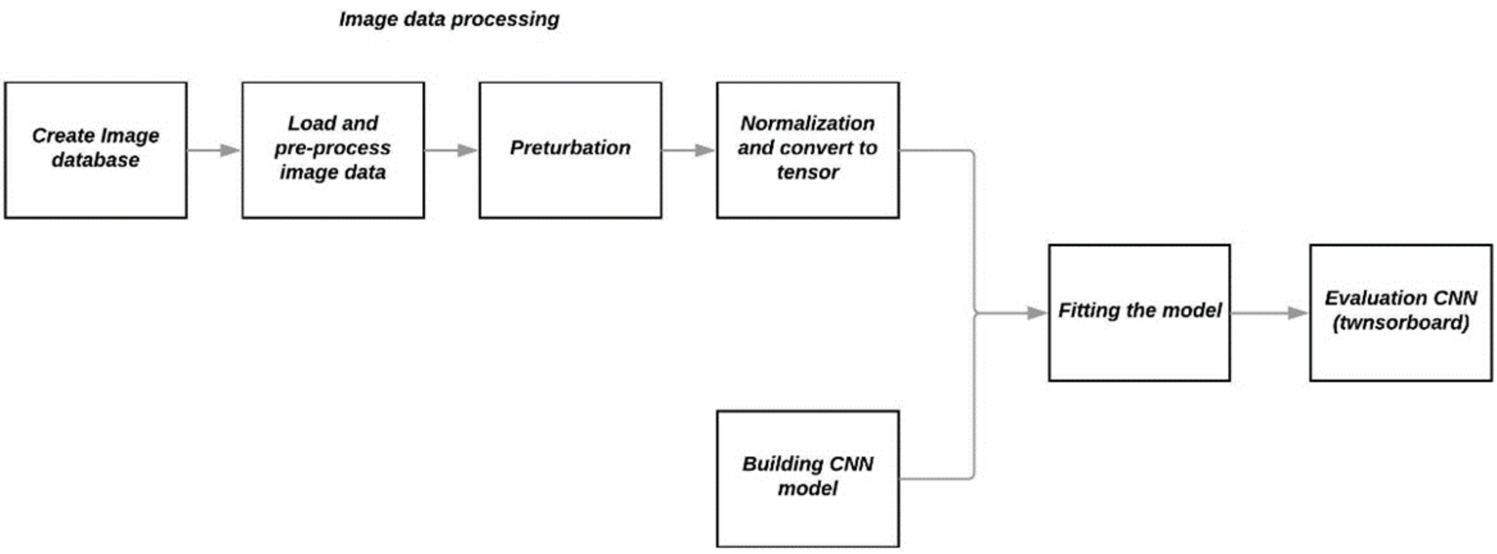
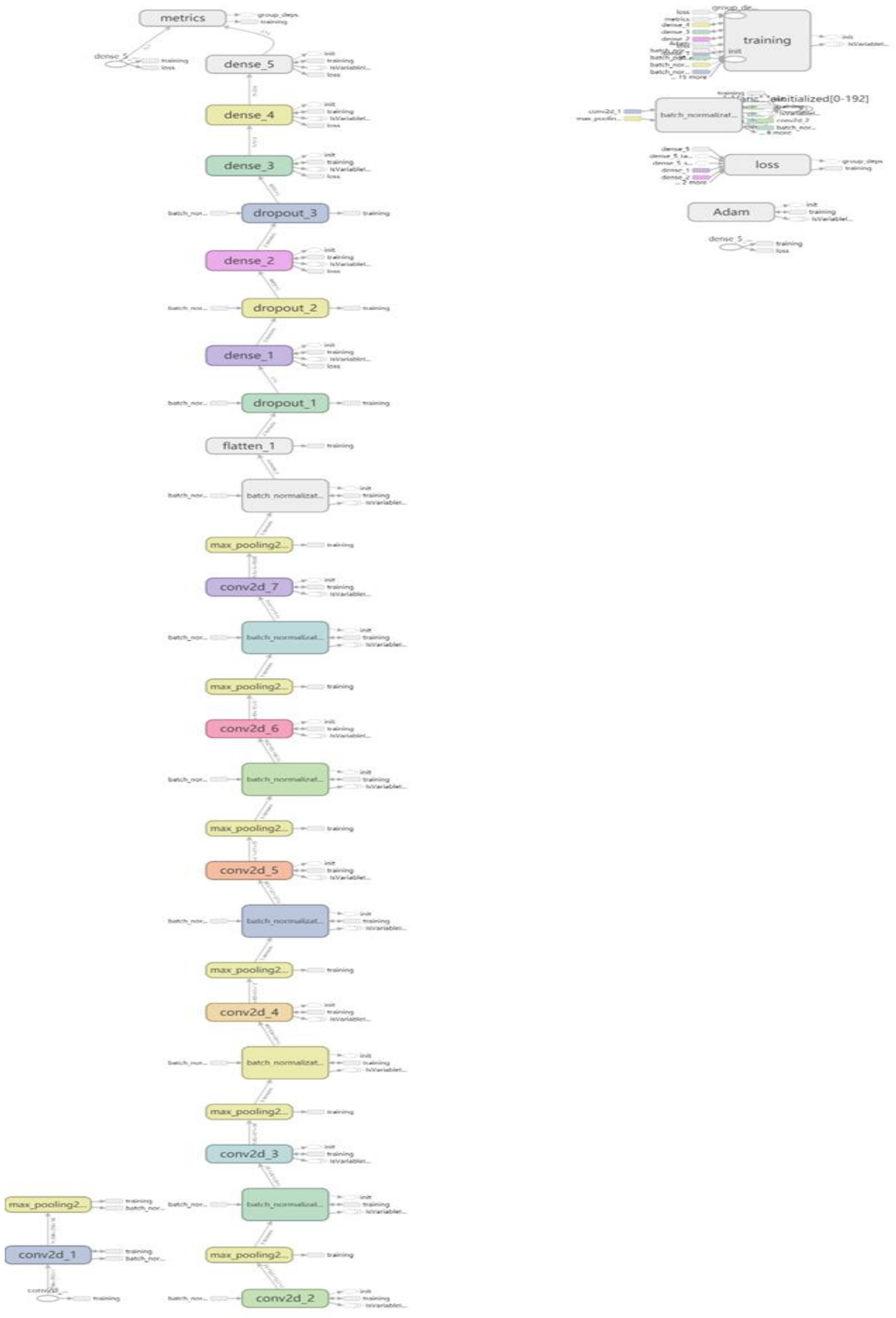


Image data processing
Create Image database
Load and pre-process image data
Preturbation
Normalization and convert to tensor
Building CNN model
Fitting the model
Evaluation CNN (twnsorboard)

The first four steps is the same as the procedures of building an autoencoder because CNN and autoencoder are both neural networks.
We have tried many different CNN structure. After some trying, we have decided the final structure of our CNN. The following figure is the structure of our CNN:

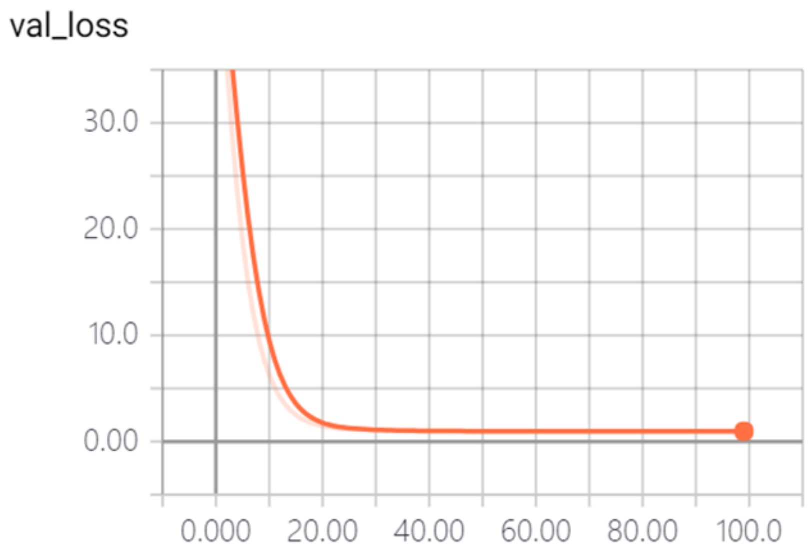

metrics
dense_5
dense_4
dense_3
dropout_3
dense_2
dropout_2
dense_1
dropout_1
flatten_1
batch_normalizat...
max_pooling2...
conv2d_7
conv2d_6
conv2d_5
conv2d_4
conv2d_3
conv2d_2
conv2d_1
training
loss
Adam
init
training
IsVariableI...
batch_nor...

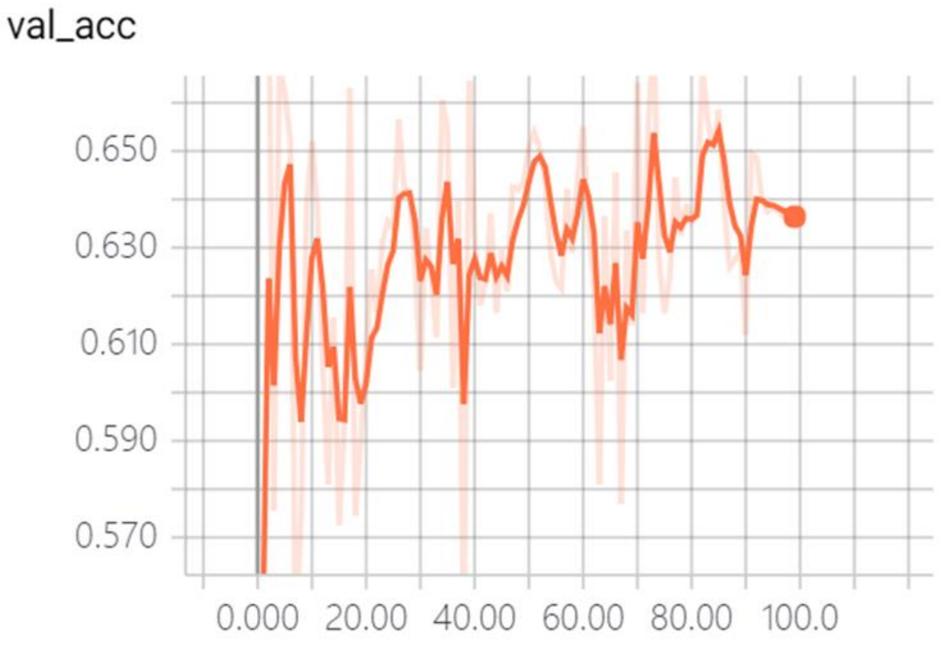
Our CNN has two parts, convolution layer part and dense layer part. In the convolution layer part, we add 6 convolution layers. Between each of them, we also add max-pooling layer and batch normalization layer. In the dense layer part, we add four dense layer part.

We use 11000 images to train the model, 2000 images for validation and 2000 images for testing.

The training epochs is 100. The accuracy on training data is 90.3%, on validation data is 81.1% and 79.0% on testing data.

Overfitting is a very important issue in training CNN model. In order to prevent overfitting, we add drop-out layer and l1&l2 regularizers. The following images are the loss on validation set with and without dropout layers and l1&l2 regularizers.





The first image is the model with drop-out layers and l1&l2 regularizers and the second one does not use these technic. We can see that the first model do not have obvious overfitting issue but the second one converge very quickly and have overfitting problem

5. Conclusion and beyond

summary

In conclusion, we use four different methods to build emotion classifier, which are SVM, Random forest, Autoencoder + SVM and Convolutional Neural Network. The result are as following:

Model	SVM	Random Forest	Autoencoder + SVM	CNN
Accuracy on testing data	0.505	0.65	0.691	0.790

Compared to SVM, random forest can give a higher accuracy on testing data. The reason maybe that the structure of random forest is more complex than SVM, which is more suitable for high dimensional image feature data. Adding the auto-encoder will increase the accuracy of SVM. This means that the autoencoder is good at feature extraction. Compared to other methods, CNN can give the highest accuracy on testing data, this implies that CNN structure works quite well on image classification problem compared to traditional machine learning method.

Lesson learned

Building a classifier includes mainly three different steps: data collecting, model building and model evaluation. For the first step, actually there are many emotion related database. Since we need to train an CNN model, we need a large number of images. The image database we choose has about 13000 images which we believe it is suitable for our project. However, the database is imbalanced, so we need to do perturbation to solve this problem. Perturbation really makes our models more accurate. For the second step, model building, we choose two libraries: sklearn and Keras. Both libraries is very powerful and easy to use. However, compared to Keras, TensorFlow is more flexible. So, I will choose TensorFlow to build CNN next time. For the final step, since our models solve classification problem, we use accuracy to evaluate our models. Matplotlib and tensor-board can also help us to evaluate the models and perform data visualization. This is the first time for us to build classifier, we really learnt a lot from this project.

Future work

After the project, we can do these three improvements to our project:

1) Improve the perturbation process: The perturbation process that we use now includes the image rotation, flipping and blurring. This perturbation process will generate many similar images. So, we need to use a more complex way to do the perturbation. We believe that it will increase the accuracy of our models.
2) Build autoencoder + random forest model: As what we have discussed before, random forest will give a better result compared to SVM. We also know that adding an auto-encoder will increase the accuracy of SVM. So, a model that combine autoencoder and random forest will give a higher accuracy.
3) Use CNN to build autoencoder: By now, our autoencoder use dense layer structure. As we know, the convolution neural network structure is designed to deal with image classification problem. So, we believe that using a CNN structure will make the training of autoencoder more efficiency.

Credit distribution

Zhibo Liang:

1) Build the SVM, autoencoder + SVM and CNN model.
2) Write the program of database generation, image loading and preprocessing.
3) Write the letter-of-intent and the last two parts of the final report.

Zubin Zhang:

1) Build the random forest model
2) Write the program of image perturbation
3) Write the midterm progress report and the first four parts of the final report

6. Reference

[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] 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.

[3] J. Li and M. Oussalah, “Automatic face emotion recognition system,” in Cybernetic Intelligent Systems (CIS), 2010 IEEE 9th International Conference on. IEEE, 2010, pp. 1–6.

[4] Y. Xing and W. Luo, “Facial expression recognition using local gabor features and adaboost classifiers,” in Progress in Informatics and Computing (PIC), 2016 International Conference on. IEEE, 2016, pp. 228–232.
[5] Y. Tang. Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239, 2013.
[6] M. Liu, S. Li, S. Shan, R. Wang, and X. Chen. Deeply learning deformable facial action parts model for dynamic expression analysis. In Computer Vision–ACCV 2014, pages 143–157. Springer, 2014.
[7] X. Wang, C. Jin, W. Liu, M. Hu, L. Xu, and F. Ren, “Feature fusion of hog and wld for facial expression recognition,” in System Integration (SII), 2013 IEEE/SICE International Symposium on. IEEE, 2013, pp. 227–232.