Face Emotion Detection Final Report

CS 273A Introduction to Machine Learning
Prof. Sameer Singh

Team Name: XYZ
Xuan Liu, Yaxi Lou, Zhiying Zhou

Fall 2018

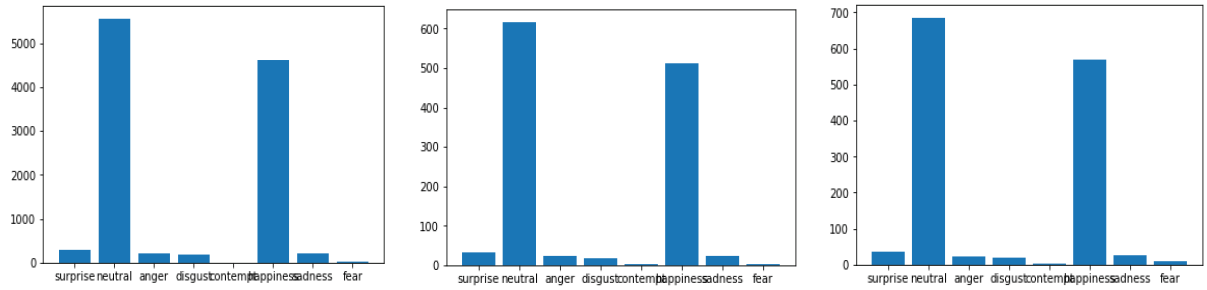
Contents

1 Introduction

Emotion recognition is an essential part of quantitative studies of human behavior. Many applications gain benefits from an accurate emotion recognizer. This project was interested in recognizing emotions and our goal was to find the correct emotion for images. Provided a dataset contained images in different emotions, we organized the images first, built up Neural Network model and some other models, and applied our models to data. Cross validation had been done to find the best parameters.

2 Data Preparation and Exploration

Provided by two folders-one contains images, the other was csv file with corresponding emotion labels, firstly, we randomly split the original dataset into training data, validation data, and test data(approximately, 80%, 10%, 10%) with each image path and its label. We totally had 1,3690 images, which were labeled in 8 different emotions: happiness, surprise, sadness, anger, contempt, disgust, neutral, and fear. For convenience, we changed the emotion labels into numbers, like surprise = 0, neutral = 1, anger = 2...



Training Validation Test

The histogram we plotted above shows the number of images with each emotions in each data. Noticed that the data are markably imbalanced: the images of "neutral"and "happiness"were much more than that of the other emotions, also there were only 9 images of "contempt"emotions, which may lead to difficulty of telling "contempt"emotion when we train our model. We also did some data processing to handle some errors in the data. For example, deleting those repeated images.

3 Model Training and Performance Measurements

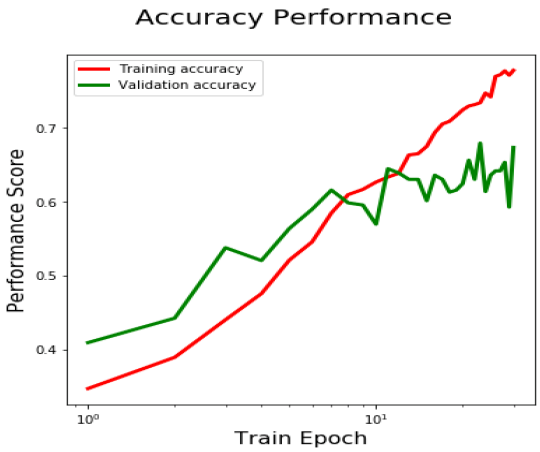
3.1 Convolutional Neural Networks (CNN)

Deep learning is well-applied in computer vision, so we explored the convolutional neural networks (CNN) algorithm which can be built by Keras, a high level neural network API uses backend TensorFlow. A typical CNN model will contain an input layer, some convolutional layers, some dense layers, and an output layer. In Keras, the model is created as Sequential( ), and more layers are added to build architecture.

In order to select the model architecture, we tried several combination of different types, sizes and numbers of layers to train the model. Based on the validation accuracy, we chose the best model below.

We started with convolutional layers which generate feature maps that represent how pixel values are enhanced. We created three relu layers with three poolings. Pooling is a dimension reduction technique usually applied after one or several convolutional layers. We used MaxPooling2D with (2,2) windows across the feature map only keeping the maximum pixel value and the pooled pixels from an image with dimensions reduced by 4. The dropout layer was added to avoid model overfitting. The output layer used sigmoid activation, which provides a probability for each emotion class.

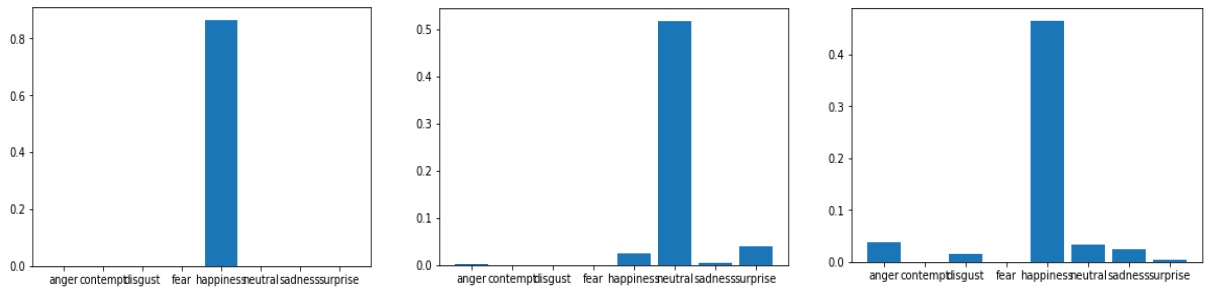
```
# Build models
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
 model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten()); model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5)); model.add(Dense(8))
model.add(Activation('sigmoid'))
# Configure learning
model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
    metrics=['accuracy'])
# Augmentation configuration for training
train_datagen = ImageDataGenerator(rescale=1. / 255, shear_range=0.2,
    zoom_range=0.2, horizontal_flip=True)
# Augmentation configuration for testing
test_datagen = ImageDataGenerator(rescale=1. / 255)
# Then, define train_generator, validation_generator, predict_generator, using
    model.fit_generator, model.evaluate_generator, model.predict_generator to
    train, evaluate and predict
```



We run this model by setting epochs = 30, batch_size = 16. The plot of training and validation accuracy with respect to different epoch is shown left. Based on the accuracy performance, we noticed that both the accuracy of training and validation increases with the increase of epoch. Also, the validation accuracy is fluctuating around 0.65 after 20 epoch. If we keep increasing epoch, we might see the accuracy of validation become flat first and then decrease afterwards – overfitting.

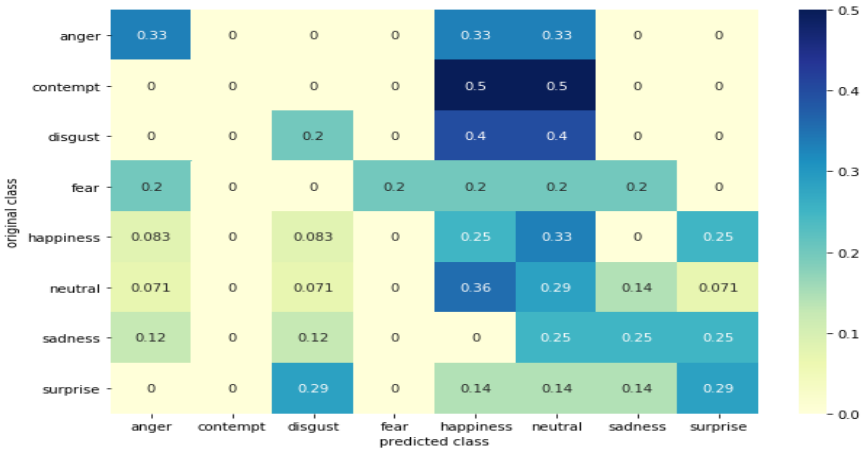
In 30th epoch, the final CNN had a training accuracy of 77.77% and a validation accuracy of 67.34%. We then could classify the test images into the emotion class which had the highest probability. We randomly chose three test images to show the probability of each class as below.

The first two faces predict right while the third face predict wrong.



Happiness $->$ Happiness Neutral $->$ Neutral Neutral $->$ Happiness

Based on the result of the prediction of test images, we can compute the confusion matrix. If X is in cell (i,j) of the confusion matrix, it represents x number points of class i are predicted as class j. The normalized confusion matrix is plot below.



Normalized confusion matrix for CNN

As you can see here that the number in the diagonal is larger than other number, which indicates the efficiency of the model. But there are also some interesting results worthy noticing. Frist, no "contempt"images are predicted correctly, it may be caused by the few numbers of "contempt"images in training data. Secondly, there are some errors between "neutral"and "happiness", which means these two classes are hard to tell by the model.

3.2 Feature Extraction

Although CNN is the most used methods in the study area of facial expression, we still attempted some other models and wanted to compare their performance.

We were intended to get the features from the face images using CNN first, then the extracted features are given to classifiers like KNN, decision tree etc to classify the emotions.

We used a pre-trained model – ResNet50 to convert images into vector representations. ResNet50 is a 50 layer Residual Network used for image classification. We input n images in the model, and got an output of n*1000 array in which the ith row represents the feature vector of the ith images. Due to the large numbers of features, an svd process of dimensional reduction is needed.

Since we will do cross-validation of different classifiers, we combined the training images and validation images.

```
from keras.applications.resnet50 import ResNet50
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input
import numpy as np
# integrate training and validation data
trva_path = tr_path + va_path; trva_label = tr_label + va_label
features = np.zeros([len(trva_path),1000]); y = np.zeros([len(trva_path),1])
```

```
# use pre-trained model-ResNet50 to extract features, store the features and
    according labels
model = ResNet50(weights='imagenet')
for i in range(len(trva_path)):
    img = image.load_img(trva_path[i], target_size=(224, 224))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    features[i] = model.predict(x)
    y = trva_label[i]
```

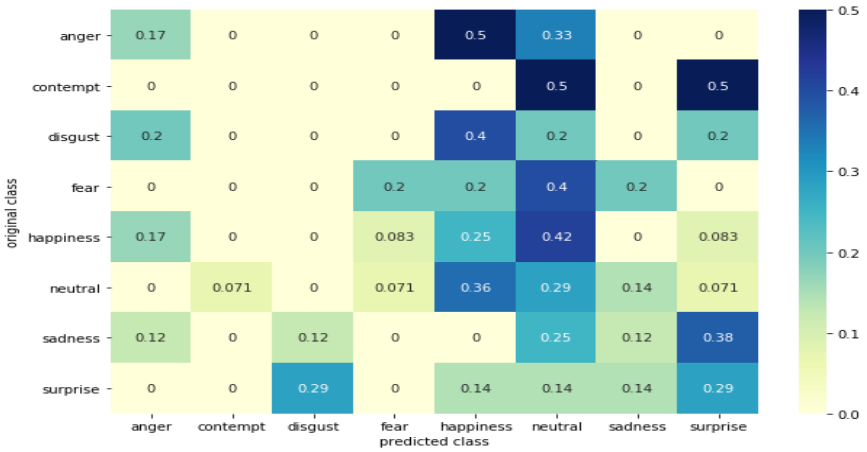
3.3 Performance Measurement

After getting the reduced features from the images, we splitted the whole output array into 3 equal folds to do cross-validation. We were intended to try KNN and Decision tree algorithm to classify these features. For the performance metrics, we computed the cross-validation accuracy and plotted the confusion matrix based on the classification of test images.

3.3.1 K-Nearest Neighbours

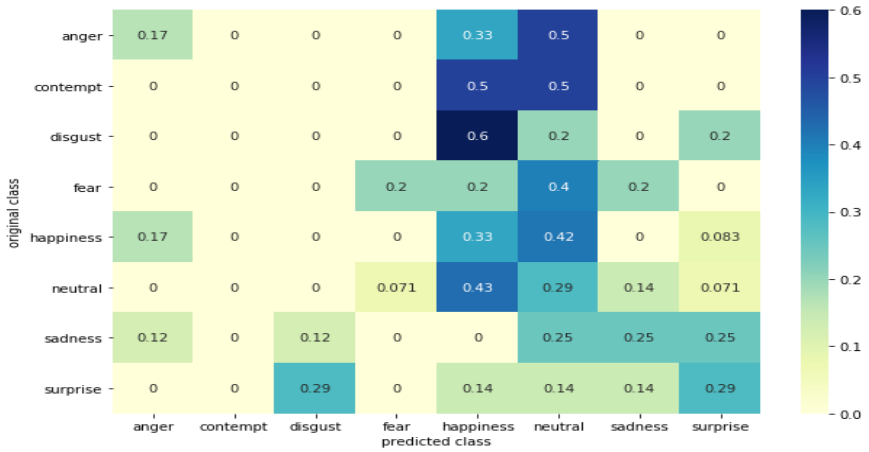
We applied a simple KNN model to classify images into the 8 categories. We chose K=11 to balance the precision rate and computation time.

The resulting cross validation accuracy is 0.61 and the confusion matrix is shown below.



3.3.2 Decision Tree

Decision Tree is widely used for classifications. By choosing specific parameters (maxDepth = 30, minParent = 21, minLeaf = 17), we obtained the best performance with cross validation accuracy of 0.64. The confusion matrix is:



4 Conclusion

After comparing classification results and cross validation accuracy from the three models (K-Nearest Neighbours, Decision Tree, CNN), we decided that CNN provided the best classification results. The final CNN model contains three convolutional layers and two dense layers, which gives training accuracy score of 0.7777 and validation accuracy score of 0.6734.

5 Future Study

In this project, we met several challenges which might be improved in a future study. Firstly, the original data has 8 categories, but the size of each class is uneven, which may have impact for model training and classification. Getting more images with equally sample sizes can improve the training model. In addition, for feature extraction, we only used the pre-trained model directly to output features without training it for our specific dataset. This will make the features have low accuracy, and it can also mainly explain why the accuracy of KNN and decision tree are low. Finally, it might be a good idea to bagging three models which reduces variance while keeps bias at a low level. More trainings and cross validation may be helpful for choosing the best parameters which we could not test this time.

6 Distribution of Work

Zhiying Zhou was responsible for data preparation and accuracy plots. Xuan Liu and Yaxi Lou were responsible for model training, and we three did testing and cross validation together. We collaborated to write and edited the final report.

References

[1] *Keras Documentation*
https://keras.io/

[2] *Jostine Ho, Facial Emotion Recognition*
https://github.com/JostineHo/mememoji

[3] *Hina Sharma, Facial Expressions Recognitions*
https://medium.com/@hinasharma19se/facial-expressions-recognition-b022318d842a