

2006532-machine learning

by Yinan Yan

FILE	24276_YINAN_YAN_2006532-MACHINE_LEARNING_316598_303146609.DOCX (1.3M)		
TIME SUBMITTED	18-JAN-2021 07:52PM (UTC+0000)	WORD COUNT	2864
SUBMISSION ID	142601607	CHARACTER COUNT	15458



Application of Machine Learning (ACCA7025)

Facial Expression Recognition Model Building Based on CNN

Submitted by:

Yinan Yan

Student ID: 2006532

Concents

1. Introduction	2
2. Reviews on key development of DNNs and CNN	3
2.1 LeNet.....	3
2.2 AlexNet.....	4
2.3 ZF-Net.....	5
2.4 VGGNet.....	5
2.5 GoogLeNet.....	6
2.6 ResNet.....	7
2.7 DenseNet	8
3. Choice of parameters/hyperparameters.....	8
3.1 Batch_size	9
3.2 Epoch.....	10
3.3 Optimizer.....	10
3.4 Learning_rate	10
3.5 Activation function.....	11
3.Description of experiment.....	11
4. Experiment evaluation and comment of final model	12
Conclusion	12
Reference.....	12
Appendix	13

Facial Expression Recognition Model Building Based on CNN

Abstract: In this report, the convolutional neural network (CNN) in machine learning is used to complete the data set and its requirements about the Facial Expression Recognition data set given in the course. A relatively good model for facial expression recognition is constructed.

Key word: machining learning, facial expression recognition, CNN

1. Introduction

FACIAL expression is one of the most powerful, natural, and immediate means for human beings to communicate their emotions and intentions [1]. Studies have shown that humans have the same perception of certain basic emotions and can be perceived, recognized and understood by people of different cultural backgrounds. These basic emotions include anger, fear, disgust, happiness, surprise, and sadness. Facial expression recognition research has extensive application prospects in many fields such as social robots, classroom concentration evaluation, and driver fatigue monitoring [2].

In recent years, with the rapid development of deep learning technology, various researches on facial expression recognition using deep learning models have made great progress. Convolutional neural networks (CNN) have good effects in the field of image processing, such as Resnet-Net, etc. In summary, for the corresponding requirements mentioned in this course, the CNN is an appropriate solution, and the CNN can solve these problems reasonably and effectively.

2. Reviews on key development of DNNs and CNN

2.1 LeNet

The LeNet network structure was proposed by Lecun in 1998. It was created to solve the problem of handwritten digit recognition. It is a solution for 10 classification tasks. The Figure 1 below is a basic network structure [2].

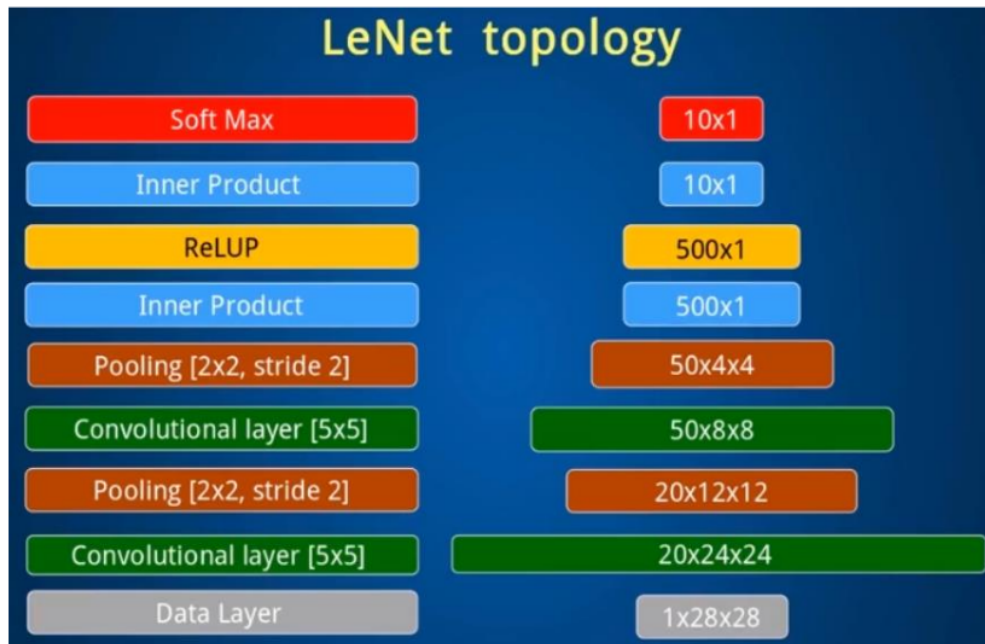


Figure 1 LeNet network structure

The LeNet network includes two convolutional layers, two down-sampling layers, two fully connected layers and an activation layer, and a softmax classification layer. It can be said that the LeNet network is a complete structure [6]. It includes all the units of the basic convolutional neural network.

2.2 AlexNet

In 2012, Alex Krizhevsky and Ilya Sutskever designed a deep convolutional neural network AlexNet in Geoff Hinton's laboratory at the University of Toronto, which won the 2012 ImageNet LSVRC championship, and the accuracy rate far exceeded the second place (top5 error rate 15.3%, the second was 26.2%), which caused a great sensation. AlexNet can be said to be a historically significant network structure. Prior to this, deep learning has been silent for a long time. Since the birth of AlexNet in 2012, the subsequent ImageNet champions are all done with convolutional neural networks (CNN). And the level is getting deeper and deeper, making CNN the core algorithm model in image recognition and classification, which has brought a great explosion of deep learning [3]. The Figure 2 below is a basic network structure.

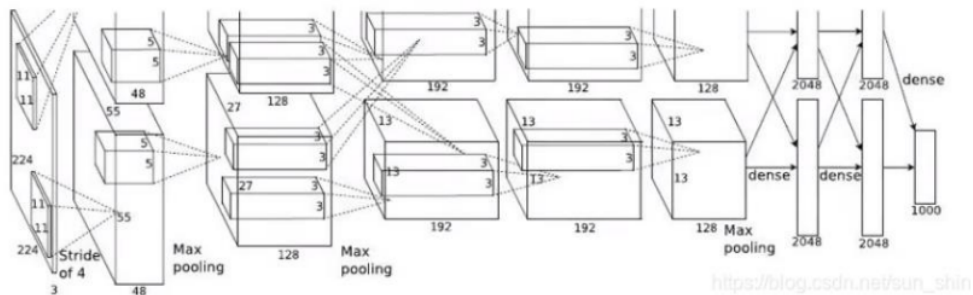


Figure 2 AlexNet network structure

The success of AlexNet is related to the characteristics of this model design, including:

- (1) Using non-linear activation function used: ReLU
- (2) Methods to prevent overfitting: Dropout, Data augmentation
- (3) Others: multi-GPU implementation, use of LRN normalization layer

2.3 ZF-Net

The ZFNet network has made small improvements on the basis of AlexNet network:

- a) From AlexNet's dual GPU to single GPU training;
- b) Visualize each layer of the neural network, which is the main contribution.

The Figure 3 below is a basic network structure of ZFNet.

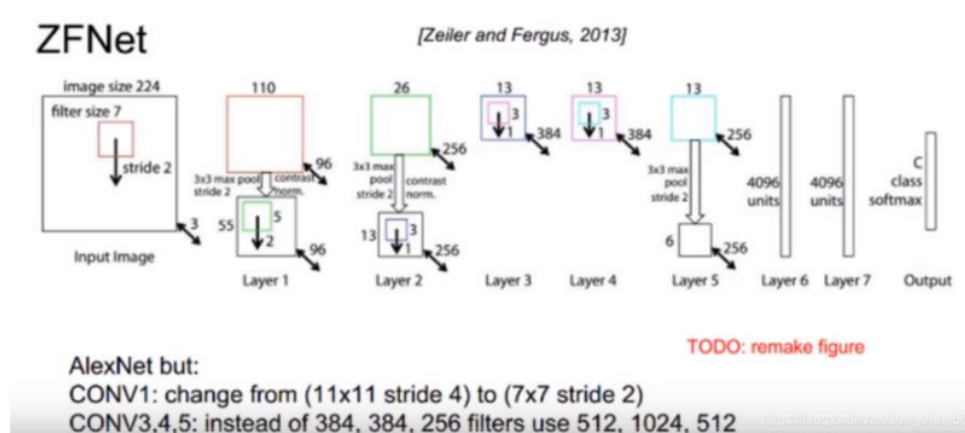


Figure 3 network structure of ZF-Net

2.4 VGGNet

2014, researchers from the University of Oxford's Visual Geometry Group and Google DeepMind developed a new deep convolutional neural network: VGGNet, and won the second place in the ILSVRC2014 classification project and the first place in the positioning project. VGGNet explored the relationship between the depth of the convolutional neural network and its performance, and successfully constructed a 16-19-layer deep convolutional neural network, proving that increasing the depth of the network can affect the final performance of the network to a certain extent. The errors rate has dropped sharply, and at the same time, the scalability is

very strong, and the generalization of migration to other image data is also very good. So far, VGG is still used to extract image features. VGGNet can be regarded as a deepened version of AlexNet, which is composed of two parts: convolutional layer and dense layer [4]. The Figure 4 below is a basic network structure of VGGNet.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64	conv3-64	conv3-64	conv3-64
maxpool					
conv3-128	conv3-128	conv3-128	conv3-128	conv3-128	conv3-128
maxpool					
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 4 network structure of VGGNet

2.5 GoogLeNet

In 2014, GoogLeNet and VGG were the winner of the ImageNet Challenge (ILSVRC14) that year. GoogLeNet won the first place and VGG won the second place. The common feature of the two types of model structures is that they have a deeper level. There are four versions of GoogLeNet, which are mainly improved in the width of the network. Unlike VGG-Net, which

simply increases the depth, it uses multiple convolutions of different sizes in the same layer to obtain different fields of view, and finally cascade (Directly superimpose the number of channels), this is the Inception module starting from v2, to further simplify the decomposition of the $n \times n$ module in the Inception module into a combination of $1 \times n$ and $n \times 1$, reducing the number of parameters, and v3 further reduces the initial 7×7 convolution and other non- 3×3 decomposition, v4 introduces the idea of ResNet residual. The Figure 5 below is a basic network structure of GoogLeNet [5].

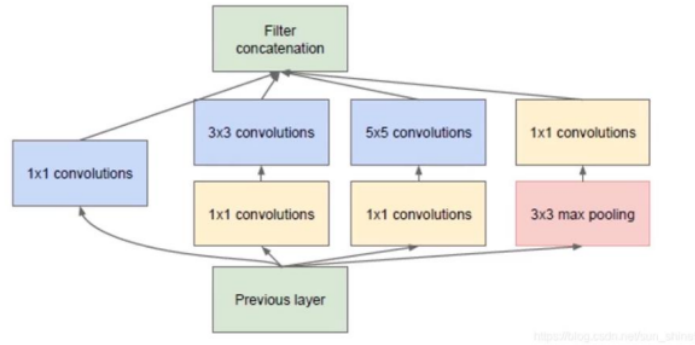


Figure 5 network structure of GoogLeNet.

2.6 ResNet

In 2015. The team of Kaiming He proposed the ResNet network that introduce a jump-connect structure to prevent the problem of gradient disappearance and increase the depth of the network. For the first time, the idea of residual error is proposed, which solves the problem of gradient disappearance caused by the network being too deep, and provides a powerful

direction for deeper networks. The Figure 6 below is a basic network structure of ResNet.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 6 network structure of ResNet.

2.7 DenseNet

As the Best Paper of CVPR2017, DenseNet breaks away from the fixed thinking of deepening the number of network layers (ResNet) and widening the network structure (Inception) to improve network performance. From the perspective of features, through feature reuse and bypass (Bypass) settings, it not only greatly reduces the amount of network parameters, but also alleviates the generation of gradient vanishing problems to a certain extent. Combining the assumptions of information flow and feature reuse

3. Choice of parameters/hyperparameters

In this report, we will use CNN to recognize facial expressions. CNN is mainly used to identify displacement, scaling and other forms of distortion invariant two-dimensional graphics. Since the feature detection layer of CNN learns through training data, when using CNN, it avoids the displayed feature extraction, and implicitly learns from the training data; in addition,

because the neuron weights on the same feature map are the same, So the network can learn in parallel, which is also a major advantage of convolutional networks over networks that connect neurons to each other. Convolutional neural network has unique advantages in speech recognition and image processing with its special structure of local weight sharing. Its layout is closer to the actual biological neural network. Weight sharing reduces the complexity of the network, especially multi-dimensional The feature that the image of the input vector can be directly input to the network avoids the complexity of data reconstruction in the process of feature extraction and classification. Therefore, during the training process of CNN, it is very important to select and adjust its parameters/hyperparameters. Considering comprehensively, in this report I have selected and adjusted the following parameters.

3.1 Batch_size

batch_size is the number of samples selected for one training, which affects the optimization and speed of the model. At the same time, it directly affects the usage of GPU memory

In general, when the batch_size's number is too small and there are more categories, it may really cause the loss function to oscillate and not converge, especially when your network is more complex. As the batch_size increases, the faster the processing of the same amount of data, and the number of epochs required to achieve the same accuracy increases. Due to the contradiction between the above two factors, batch_size is increased to a certain time to reach the optimal time. Since the final convergence accuracy will fall into different local extrema, the batch_size is increased to a certain time to reach the optimal final convergence accuracy. The

selection of the specific batch size is related to the number of samples in the training set. According to the data set that the assignment offered, it is a better choice that batch_size equal to 64.

3.2 Epoch

Epoch's definition is that Use all the data in the training set to perform a complete training of the model, which is called "first generation training".

More iterations may lead to over-fitting of the model. Few iterations will result in insufficient fit of the model. Therefore, the number of iterations needs to obtain a suitable value in the model. In the construction of this model, it is a relatively better choice that epoch is equal to 20

3.3 Optimizer

Adam is a fast optimizer that is often used. Although SGD converges slowly, adding Momentum can speed up the convergence. At the same time, the SGD algorithm that drives momentum has a better optimal solution, that is, the model will have higher accuracy after convergence. After using the two optimizers to operate, it is found that the model accuracy of SGD is relatively high, so the optimizer chooses SGD.

3.4 Learning_rate

Learning_rate indicates the magnitude of each parameter update. If the learning_rate is too large, the parameters to be optimized will fluctuate near the minimum value and will not converge; If the learning_rate is too small, the parameters to be optimized will converge slowly.

So we should choose a suitable value.

3.5 Activation function

The activation function is not really going to activate something. In the neural network, the function of the activation function is to add some nonlinear factors to the neural network, so that the neural network can better solve more complex problems. After experimenting and comparing multiple activation functions, ReLu is a relatively good activation function

3.Description of experiment

The file given in the task is in the form of a picture file. Since the data set is open source, I found the corresponding data set in the form of a csv file, so I directly used the data set in the form of csv for training. We first checked the storage format of the data in the csv file, and then preprocessed the data, first standardized the data, and finally processed the data into a data format that can be recognized by CNN. Then build the CNN network, first perform convolution operation, convolution to obtain local features, the convolution layer is to constantly change the convolution kernel to initially determine the convolution kernel of the picture, and obtain the corresponding convolution kernel multiplication output matrix. Then perform a pooling operation to retain the most important features of the picture. Then two identical convolutional layers are being constructed. The convolutional layer transitions to the dense layer. The dense layer is set to have 1024 neurons, the features extracted by the convolutional layer are assembled into a picture, and the dense layer realizes the role of classification. Then set the loss

function and optimizer. Finally, save the model.

4. Experiment evaluation and comment of final model

Considering a variety of situations, the final model is obtained by adjusting certain parameters. The model is saved in the corresponding file. Although the result of the final experiment on the test set is not very satisfactory, it is still barely acceptable. It is also relatively satisfactory to the overall experimental process

Conclusion

The CNN network is a relatively efficient machine learning framework. After this experiment of CNN facial expression recognition, I have a deeper understanding of CNN network. Although the accuracy of the final model is not very high, in general the purpose of the experiment is to understand the knowledge. Achieving this goal is also the purpose of the experiment.

Reference

[1] TIAN Y-I, KANADE T, COHN J F. Recognizing action units for facial expression analysis

[J]. IEEE

Transactions on pattern analysis and machine intelligence, 2001, 23(2): 97-115.

[2]https://blog.csdn.net/qq_50460949/article/details/100038344?ops_request_misc=&request_id=&biz_id=102&utm_term=CNN%E5%92%8CDNN%E5%8F%91%E5%B1%95&utm_m

[edium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-3-100038344.first_rank_v2_pc_rank_v29&spm=1018.2226.3001.4187](https://my.oschina.net/u/4345418/blog/4019130)

(Access:January.16,2021)

[3] <https://my.oschina.net/u/4345418/blog/4019130>

[4] <https://my.oschina.net/u/4345418/blog/4019135>(Access:January.16,2021)

[5]<https://my.oschina.net/freeshow/blog/4428869>(Access:January.16,2021)

[6] <https://my.oschina.net/u/4345418/blog/4019131>(Access:January.16,2021)

Appendix

```
1. import pandas as pd
2. import numpy as np
3. from keras.models import Sequential
4. from keras.layers import Conv2D, Dense, Dropout, Flatten, MaxPooling2D
5. from keras.losses import categorical_crossentropy
6. from keras.optimizers import Adam
7. from keras.utils import np_utils
8. from keras import optimizers
9.
10.
11. Catch_bat = pd.read_csv('C:/Users/73194/Desktop/fer2013/fer2013.csv')
12. print(Catch_bat.info())
13. print(Catch_bat['Usage'].value_counts())
14. print(Catch_bat.head())
15.
16.
17. X_train, Y_train, X_test, Y_test = [], [], [], []
18.
19.
20. for index, row in Catch_bat.iterrows():
21.     #print(index)
22.     #print(row)
23.     val = row['pixels'].split(' ')
24.     #print(val)
25.     try:
26.         if 'Training' in row['Usage'] :
27.             X_train.append(np.array(val, 'float32'))
```

```

28.         Y_train.append(row['emotion'])
29.     elif 'PublicTest' in row['Usage'] :
30.         X_test.append(np.array(val, 'float32'))
31.         Y_test.append(row['emotion'])
32.     except:
33.         pass
34. print(f'X_train:{X_train[0:2]}')
35. print(f'Y_train:{Y_train[0:2]}')
36. print(f'X_test:{X_test[0:2]}')
37. print(f'Y_test:{Y_test[0:2]}')
38.
39. # 转化数据集载入 keras
40. X_train = np.array(X_train, 'float32')
41. Y_train = np.array(Y_train, 'float32')
42. X_test = np.array(X_test, 'float32')
43. Y_test = np.array(Y_test, 'float32')
44. #print(X_train)
45. #print(Y_train)
46.
47.
48.
49. #进行数据的标准化处理（进行均值处理和标准化的校正）
50. X_train -= np.mean(X_train, axis=0)    #axis =0 计算每列求均值
51. X_train /= np.std(X_train, axis=0)     #axis = 0 计算每一列的标准差
52.
53. X_test -= np.mean(X_test, axis=0)
54. X_test /= np.mean(X_test,axis=0)
55.
56. labels = 7
57. Y_train = np_utils.to_categorical(Y_train, num_classes=labels)
58. Y_test = np_utils.to_categorical(Y_test, num_classes=labels)
59.
60.
61. features = 64    #卷积输出滤波器的数量
62. batch_size = 64
63. epochs = 20
64. width, height = 48, 48
65.
66. #生成新的列表，修改数据
67.
68. X_train = X_train.reshape(X_train.shape[0],width, height,1)
69. X_test = X_test.reshape(X_test.shape[0],width, height,1)
70.
71. #构建 CNN 网络

```

```

72.
73. model = Sequential()
74.
75.
76. #第一部分
77.
78. #卷积取得局部特征，卷积层的作用是通过不断改变卷积核来初步确定图片的特征用的卷积核是哪些，得到再与相应的卷积核相乘输出矩阵
79. #进行卷积操作，kernel_size 指定两个维度高度宽度 input_shape 代表输入图片
80. model.add(Conv2D(features, kernel_size=(3, 3), activation='relu', input_shape=(X_train.shape[1:])))
81. model.add(Conv2D(features, kernel_size=(3, 3), activation='relu'))
82.
83. #进行池化操作，保留图片是最重要的特征
84. model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
85.
86. #让百分之五十神经元不工作
87.
88. model.add(Dropout(0.5))
89.
90. #第二部分
91. model.add(Conv2D(features, kernel_size=(3, 3), activation='relu'))
92. model.add(Conv2D(features, kernel_size=(3, 3), activation='relu'))
93. model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
94. model.add(Dropout(0.5))
95.
96. #第三部分
97.
98. model.add(Conv2D(2*features, kernel_size=(3, 3), activation='relu'))
99. model.add(Conv2D(2*features, kernel_size=(3, 3), activation='relu'))
100. model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
101.
102. #卷积层过渡到 全连接层
103. model.add(Flatten())
104.
105. #1024 个神经元，卷积提取的特征组装成一个图片，所以叫全连接层实现分类的作用
106. model.add(Dense(2*2*2*2*features, activation='relu'))
107.
108. #让百分之二十神经元不工作
109. model.add(Dropout(0.2))
110.
111.
112. #输出 7 个结果
113.

```

```

114. model.add(Dense(labels, activation='softmax'))
115.
116.
117. #loss funtion 交叉双损失函数 计算准确率, 制定优化器
118.
119.
120.
121. model.summary()
122. sgd=optimizers.SGD(lr=0.00001, decay=1e-6, momentum=0.9, nesterov=True)
123. model.compile(loss=categorical_crossentropy, optimizer=sgd, metrics=['accuracy'])
124.
125.
126. #训练模型 ,verbose=1 显示训练的进度条 validation_data 根据数据集上的效果进行调整 shuffle
    进行随机打乱
127.
128. model.fit(X_train, Y_train, batch_size=batch_size, epochs=epochs, verbose=1, valida
    tion_data=(X_test, Y_test), shuffle=True)
129.
130.
131. #保存模型
132.
133.
134. #保存网络结构
135. fer_json = model.to_json()
136. with open('fer.json', 'w') as json_file:
137.     json_file.write(fer_json)
138.
139. #保存模型的参数
140. model.save_weights('fer.h5')
141.
142.
143. test_score=model.evaluate(X_test, Y_test,verbose=0)
144. print('Test loss', test_score[0])
145. print('Test accuracy:', 100*test_score[1])

```

FINAL GRADE

GENERAL COMMENTS

Instructor

52/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11



Comment 1

We discussed other approaches in class such as early stopping.

PAGE 12



Comment 2

A little bit more detail here please!

PAGE 13



Comment 3

References.



Comment 4

This needs reformatting!

PAGE 14



Comment 5

Use quality refs! Avoid online refs if possible.

PAGE 15

PAGE 16

PAGE 17

GRADING FORM: SMU GENERIC_GJ

YINAN YAN

51.50

RESEARCH-INFORMED LITERATURE



Refs poor. Avoid online refs and format your refs correctly. Principles and related work, however is covered well.

6

KNOWLEDGE AND UNDERSTANDING



Key ideas covered here and terms defined. You sate you want to set your epochs to 20 early stopping? Could be more detailed in some places. Good understanding of code/CNN principles shown here. What about evaluation? Confusion matrix, etc.

7

ANALYSIS

 You could have tried several experiments to see if you could improve your architecture and hence the accuracy.

12

PRACTICAL APPLICATION AND DEP.

 DNN design coded and trained. Testing undertaken too. Some accuracy calculations undertaken but these weren't documented in your report.

24

SKILLS FOR PROFESSIONAL PRAC.

 Structured report. Refs could be better. Diagrams and plots would help here!

2.5