

Experimental Report on Predicting Expression Datasets Using Convolutional Neural Network Models Defined in Keras Library

by Shuyao Jiang

Submission date: 04-May-2023 10:04PM (UTC+0100)

Submission ID: 204791827

File name:

41869_Shuyao_Jiang_Experimental_Report_on_Predicting_Expression_Datasets_Using_Convolutional_Neural_Network_Models_Def_1350348011.docx
(758.1K)

Word count: 3227

Character count: 17708

Experimental Report on Predicting Expression Datasets Using Convolutional Neural Network Models Defined in Keras Library

Tutor : Gordon Dickers

Student: Shuyao Jiang

Student number: 2203626

Abstract - We construct a CNN model by using the convolutional neural network model defined in the Keras library. And we define each neural network layer by ourselves. Classifying and validating the self-made facial expression dataset through our model, and complete the experiment. We constructed a neural network model with four convolutional layers and two fully connected layers to validate and classify a self-made grayscale image dataset with seven classes. The dataset was a self-made human expression dataset, with 7178 images in the test set and 28709 images in the training set. Through conducting experiments on a large amount of data, we ultimately obtained experimental results with a minimum accuracy of 31.1% and a maximum accuracy of 62.8%.

Keywords- CNN model, Keras library, neural network layer, facial expression dataset, grayscale image dataset, convolutional layers, fully connected layers, seven classes, test set, training set, accuracy.

I. INTRODUCTION

The CNN model was first proposed by Yann LeCun and his colleagues in 1989. At that time, LeCun and his team used a convolutional neural network model called LeNet for handwritten digit recognition tasks. LeNet is a neural network that includes multiple convolutional and pooling layers, which can automatically extract features from input images and map them to corresponding output categories. The innovation of this model lies in its successful application of convolutional operations in neural networks, which greatly reduces the complexity and number of parameters of the model by sharing weights and local connection characteristics, and achieves good recognition results.

CNN (Convolutional Neural Network) is a deep learning model commonly used for image classification tasks. The core principle of CNN is to extract and reduce features from the image through convolution and pooling operations, and then map the extracted features to the output category using fully connected layers. In CNN, the convolution layer convolves the input image with a set of learnable convolution kernels to extract local features of the image, such as edges, corners, and textures. The pooling layer reduces the dimensionality of the feature maps output by the convolution layer to reduce the number of model parameters and prevent overfitting. Finally, the fully connected layer maps the extracted feature vectors to the output category, typically consisting of one or more fully connected layers and a softmax layer, allowing the model to automatically learn the most discriminative features from the image and achieve high-precision image classification tasks.

II. THE STRUCTURE OF CONVOLUTIONAL NEURAL NETWORKS

A. Introduction to Convolutional Neural Network Structure

Convolutional Neural Network (CNN) is a kind of feedforward neural network, whose structure consists of convolution layer, pooling layer and full connection layer. Its basic structure is as follows:

1. Input Layer: Receives images or other multidimensional data to be processed.
2. Convolutional Layer: Use convolutional verification to perform convolution operations on input data and extract local features of the image.
3. Activation Layer: use the activation function to perform nonlinear transformation on the output of the convolution layer, so as to increase the expression ability of the model^[1].
4. Pooling Layer: conduct down sampling operation for the output of the convolution layer to reduce the amount of model parameters and reduce the risk of overfitting^[2].
5. Fully Connected Layer: Inputs the feature vectors output by the pooling layer into a fully connected neural network for classification or regression tasks.

6. Output Layer: Outputs classification or regression results.

We will focus on introducing the content of convolutional layer, activation layer, and fully connected layer. Then we will explain our model and introduce its structure.

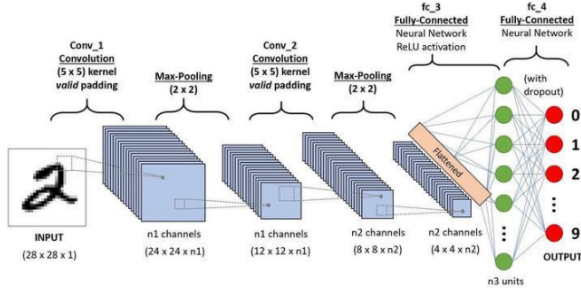


Figure.1. Schematic diagram of convolutional neural network

B. Convolutional Layer

Convolutional layer is a fundamental component of Convolutional Neural Network (CNN) used for feature extraction from input data such as images. In a convolutional layer, a set of learnable filters, also known as kernels or weights, are convolved with the input data to extract the relevant features. The convolution operation involves sliding the kernel over the input data^[3], computing the element-wise multiplication between the kernel and the input data at each location, and summing up the results to obtain the output feature map. The output feature map represents the activations of the learned features at each spatial location of the input data. The convolutional layer also includes non-linear activation functions such as ReLU to introduce non-linearity into the network and increase its expressive power. The use of convolutional layers allows the network to learn hierarchical representations of the input data, making it well-suited for tasks such as image classification, object detection, and segmentation^[3].

The number of convolutional layers in a neural network is usually influenced by factors such as the complexity of the input data and the size of the dataset^[4]. In theory, a neural network with more convolutional layers should be able to learn more complex features, but too many convolutional layers can lead to overfitting. Overfitting is a common problem in machine learning, where the model becomes overly complex and fits the training data too closely, resulting in poor generalization to new, unseen data. In other words, the model learns the noise and idiosyncrasies of the training data, rather than the underlying patterns and structures that generalize to new data^[5]. Therefore, it is important to strike a balance between the number of convolutional layers and the model's ability to generalize to new data.

Here, in order to prevent overfitting problems, we adopt a method of gradually increasing the number of convolutional layers to detect overfitting problems. Firstly, we set up a layer of convolutional layers and continuously add the number of convolutional layers to analyze the experimental results. This is a simple method, but it is troublesome. Of course, we also add a dropout layer behind each convolution layer to prevent overfitting. Due to the influence of experimental time and equipment, we ultimately constructed the convolutional layer of our neural network model into four layers.

C. Batch Normalization layer

Unlike normal CNN models, we have added a batch normalization layer between the convolutional layer and the activation layer. When using a shallow model, as the model training progresses, it is difficult for the output near the output layer to undergo significant changes as the parameters in each layer are updated. Batch normalization (BN) is a technology used for deep learning, which helps to improve the performance and stability of neural networks, and can prevent overfitting problems^[6].

The following is the process of training the BN algorithm:

For the output result of the previous layer, $X = x_1, x_2, \dots, x_m$. The learning parameters are γ, β . Firstly, the mean of the output data from the previous layer should be calculated:

$$\mu_\beta = \frac{1}{m} \sum_{i=1}^m (x_i)$$

(M is the size of the batch for this training sample)

Step 2: Calculate the standard deviation of the output data from the previous layer:

$$\sigma_\beta^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_\beta)^2$$

Step 3, normalization processing:

$$X_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}$$

D. Activation layer

The activation layer is a layer in convolutional neural networks and an important component of deep learning. Its main function is to introduce nonlinear factors by performing nonlinear transformations on the output of the convolutional layer after the convolutional layer, thereby improving the expression ability of the neural network. If there is no activation function, then the entire neural network is equivalent to a linear function, so it is unable to deal with complex nonlinear problems. The activation function is the core of the activation layer. The activation function can map the output of the neuron to a nonlinear interval, so that the neural network can fit more complex functions. Common activation function of the activation layer include ReLU, sigmoid, tanh, etc. Among them, ReLU is one of the most commonly used activation function, which has the advantages of fast calculation speed, fewer gradient disappearance problems, and can converge quickly in the training process. In addition, the activation layer can also help neural network models better adapt to data, improve the generalization ability of the model, and thus improve the prediction accuracy of the model^[6].

The activation function we use is ReLU function, and its mathematical formula is as follows:

$$f(u) = \max(0, u) \quad (4)$$

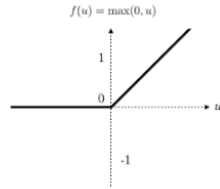


Figure 2. ReLU function

E. Fully Connected Layer

The fully connected layer is a basic layer of neural networks. In this layer, each neuron has connections to all neurons in the previous layer. The fully connected layer is usually used as the last layer of a neural network to classify or regress the features extracted from the previous layer. Unlike convolutional and pooling layers, fully connected layers do not share weights and require learning the weights and bias terms of each connection. The full connection layer can realize nonlinear mapping and increase the nonlinear expression ability of the model. However, in deep neural networks, the full connection layer often leads to overfitting of the model, so the full connection layer should be carefully used in the design of neural networks^[7].

We perform classification tasks through two fully connected layers. In the first full connection layer, 256 neurons are used, followed by a batch normalization layer, a activation function layer, and a Dropout layer. In the second full connection layer, 512 neurons are used, followed by a batch normalization layer, a activation function layer, and a Dropout layer. The final output layer uses the Softmax activation function to transform the output of the model into the probability distribution of seven categories, representing the probability of each category.

III. DATASETS

A. Dataset Introduction

Below, we will introduce the dataset we used in this experiment. The dataset used in this experiment is our self-made dataset, which includes 35887 grayscale images. The dataset has been divided into a test set and a training set. The number of images in the test set is 7178, and the number of training sets is 28709. All images are divided into seven categories, and the content of the images is about human expressions. What we need to do is to use our model to classify and predict the image dataset to determine whether our model is excellent enough to classify human expressions. For grayscale images, we know that they are a single channel image, and the storage of the image is in the form of a matrix. A grayscale image is composed of many points, with values ranging from 0 to 255. 0 represents a pure white dot, while 255 represents a pure black dot. The values of other gray points range from 0 to 255, which allows us to store images in the form of data^[8].

Secondly, the vast majority of the image formats used in this experiment are in JPG format, which is a widely used image compression standard that can compress images to a small fraction of their original size without affecting image quality. The JPG format is commonly used for photos taken by digital cameras, images on the internet, and other images that require compression. In JPG format, image data is divided into blocks of 8x8 pixels, and each block is compressed and stored.

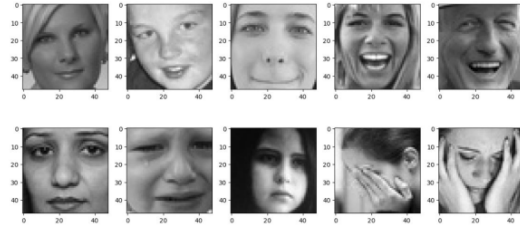


Figure 3. Image display in the dataset

B. Unified data format

Before the experiment, we observed the datasets and found that the vast majority of images were in JPG format. However, we cannot guarantee that there are no other formats in the dataset, so the first step is to remove other formats from the dataset. If the number of images removed is not too much, it is considered that it has no impact on the experimental results. After our verification, the number of JPG format images in the training set is 28709, which is consistent with the number of images in the original training set. We believe that different image formats have no impact on the experimental results.

7215	happy images
4830	sad images
4097	fear images
3171	surprise images
4965	neutral images
3995	angry images
436	disgust images

Table.1. Number of jpg format images in the training set

Setting an appropriate image size can ensure the computational efficiency of our model. For our human facial expression images, we set all images in the dataset to 48 pixels. This is obtained by consulting relevant literature, as the use of small images can accelerate the training speed of the model, reduce the computational cost of the model, and the size of 48 pixels is sufficient to distinguish different expression categories. This is why we set the image size.

What we need to do next is to enhance the training dataset to increase its diversity and quantity, thereby improving the model's generalization ability. Data augmentation can be achieved through various methods, such as random rotation, random scaling, horizontal or vertical flipping, random cropping, and so on. These transformations can make images appear different, thereby increasing the diversity of training data, enabling the model to learn more features, and improving its generalization ability. Here we use ImageDataGenerator. ImageDataGenerator is a Keras class used for data augmentation and batch processing of images. ImageDataGenerator can perform random transformations on images and train the processed images into the model through batch processing.

V. EXPERIMENTAL RESULTS

A. Create CNN Model

In this model, we employ four convolutional layers and four pooling layers, with each convolutional layer followed by a batch normalization layer and an activation function layer. These layers are designed to capture various image features, including texture, shape, and color. Moreover, to prevent overfitting, we add a dropout layer after each convolutional layer. Dropout is a regularization technique that randomly omits a portion of the output of neurons with a certain probability to reduce the network's dependence on specific features. For convolution operations, we use 64 kernels with a size of 3x3, maintain the input shape (padding='same'), and set the input image shape as (48, 48, 1). Batch normalization is applied to standardize the input, which helps to enhance the training effectiveness of the model. We choose ReLU as the activation function, but other activation functions can also be used. ReLU is used to perform nonlinear transformations on the standardized inputs. We use a 2x2 window size for maximum pooling operations to reduce the feature map's size. The choice of 2x2 pooling is based on its translation and local invariance properties, which can reduce the image's spatial size.

computational complexity, and parameter count of the model. Additionally, the 2x2 pooling operation can prevent overfitting to some extent without losing essential information, making it a commonly used pooling operation size^[10].

Finally, the classification task is performed through two fully connected layers. In the first fully connected layer, 256 neurons are used, followed by a batch normalization layer, an activation function layer, and a Dropout layer^[9]. In the second fully connected layer, 512 neurons are used, followed by a batch normalization layer, an activation function layer, and a Dropout layer. The final output layer uses the Softmax activation function to convert the model's output into a probability distribution of 7 classes, representing the probability of each class.

Model: "sequential"			
Layer (type)	Output Shape	Param #	
=====			
conv2d (Conv2D)	(None, 48, 48, 64)	640	conv2d_3 (Conv2D) (None, 6, 6, 512) 2358008
batch_normalization (Batch Normalization)	(None, 48, 48, 64)	256	batch_normalization_3 (Batch Normalization) (None, 6, 6, 512) 2048
activation (Activation)	(None, 48, 48, 64)	0	activation_3 (Activation) (None, 6, 6, 512) 0
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0	max_pooling2d_3 (MaxPooling2D) (None, 3, 3, 512) 0
dropout (Dropout)	(None, 24, 24, 64)	0	dropout_3 (Dropout) (None, 3, 3, 512) 0
conv2d_1 (Conv2D)	(None, 24, 24, 128)	204928	flatten (Flatten) (None, 4608) 0
batch_normalization_1 (Batch Normalization)	(None, 24, 24, 128)	512	dense (Dense) (None, 256) 1179004
activation_1 (Activation)	(None, 24, 24, 128)	0	batch_normalization_4 (Batch Normalization) (None, 256) 1024
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 128)	0	activation_4 (Activation) (None, 256) 0
dropout_1 (Dropout)	(None, 12, 12, 128)	0	dropout_4 (Dropout) (None, 256) 0
conv2d_2 (Conv2D)	(None, 12, 12, 512)	590336	dense_1 (Dense) (None, 512) 121584
batch_normalization_2 (Batch Normalization)	(None, 12, 12, 512)	2048	batch_normalization_5 (Batch Normalization) (None, 512) 2048
activation_2 (Activation)	(None, 12, 12, 512)	0	activation_5 (Activation) (None, 512) 0
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 512)	0	dropout_5 (Dropout) (None, 512) 0
			dense_2 (Dense) (None, 7) 3591
=====			
Total params: 4,478,727			
Trainable params: 4,474,759			
Non-trainable params: 3,968			

Figure.4. Result of creating CNN model

B.Loss rate and accuracy rate

In machine learning, loss is an indicator used to measure the difference between the predicted results of a model and the actual results. The smaller the loss, the closer the predicted results of the model are to the true values, which means the better the performance of the model. Accuracy refers to the proportion of the correct number of samples in all predictions of a model to the total number of samples. In classification problems, accuracy refers to the classification accuracy of the model for all class samples, and is one of the most commonly used evaluation indicators^[11]. Generally speaking, the higher the accuracy, the better the model's performance.

experimental result:

accuracy

training	(min: 0.311, max: 0.628, cur: 0.628)
validation	(min: 0.374, max: 0.621, cur: 0.618)

Loss

training	(min: 0.980, max: 1.796, cur: 0.980)
validation	(min: 1.010, max: 1.564, cur: 1.017)

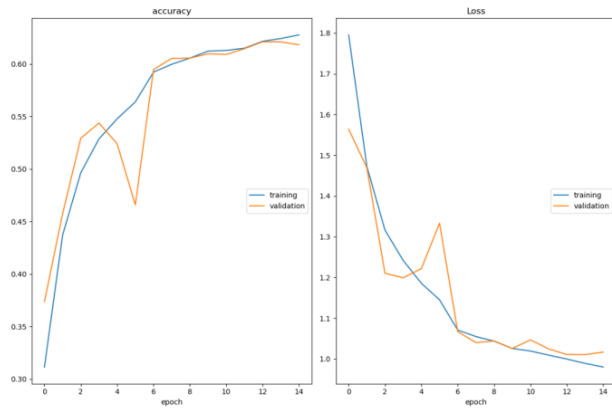


Figure.5. result of the loss and accuracy

We can see that the loss value is high and the accuracy is low, indicating that our model has some flaws. Here we set the value of epochs to 15 because a larger value of epochs would result in a longer experimental time. If we increase the value of epochs, theoretically, it can improve accuracy and reduce loss. Another reason for the poor results may be that our CNN model is not constructed perfectly. We can increase the number of convolutional layers to better fit the problem. In theory, increasing the number of convolutional layers can improve accuracy.

C.Verification results using the confusion matrix

The confusion matrix is a form of visual classification model prediction results. It compares each category predicted by the model with the actual category and categorizes the results as True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).

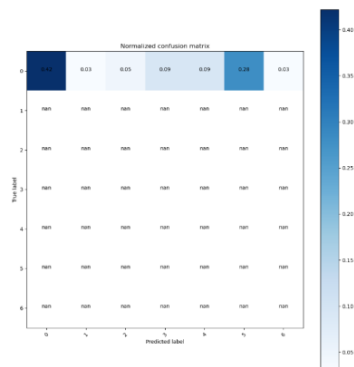


Figure.6.confusion matrix

The recall rate refers to the proportion of correctly predicted positive samples to all actual positive samples.. It represents the ability of the model to correctly recognize positive examples.

$$TP/(TP+FN)$$

Accuracy refers to the proportion of correctly predicted samples to all samples.. It represents the overall accuracy of the model.

$$(TP+TN)/(TP+TN+FP+FN)$$

Accuracy refers to the proportion of correctly predicted samples to all samples. It represents the overall accuracy of the model.

$$(TP+TN)/(TP+TN+FP+FN)$$

	<i>precision</i>	<i>recall</i>	<i>f1-score</i>	<i>support</i>
0	1.00	0.42	0.59	64
1	0.00	0.00	0.00	0
2	0.00	0.00	0.00	0
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
5	0.00	0.00	0.00	0
6	0.00	0.00	0.00	0
<i>accuracy</i>		0.42		64
<i>macro avg</i>	0.14	0.06	0.08	64
<i>weighted avg</i>	1.00	0.42	0.59	64

REFERENCE

- [1]. Shouwu Hou, Zheng Zhang, Shiyuan Feng, Haiping Zhu, Liangzhi Fan. "A diagnosis method of capsule surface damage based on convolutional neural network" , ITM Web of Conferences, 2022
- [2]. "Intelligence Science I" , Springer Science and Business Media LLC, 2017
- [3]. Babak Chehreh, Alexandra Moutinho, Carlos Viegas. "Latest Trends on Tree Classification and Segmentation Using UAV Data—A Review of Agroforestry Applications" , Remote Sensing, 2023
- [4]. Paul Kariuki, Dr. Patrick Kinyua Gikunda, Dr. John Mwangi Wandeto. "Deep Transfer Learning Optimization Techniques for Medical Image Classification - A Survey" , Institute of Electrical and Electronics Engineers (IEEE), 2023
- [5]. Weihua Zhang, Daniel C.W. Tsang. "Conceptual framework and mathematical model for the transport of metal–chelant complexes during in situ soil remediation" , Chemosphere, 2013
- [6]. Peibing Wang, Yuan Peng. "Research on Feature Extraction and Recognition Method of Underwater Acoustic Target Based on Deep Convolutional Network" , 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA), 2020
- [7]. Shashi Bhushan Jha, Radu F. Babiceanu. "Deep CNN-based visual defect detection: Survey of current literature" , Computers in Industry, 2023
- [8]. Yanyi Li, Jian Shi, Yuping Li. "Real-Time Semantic Understanding and Segmentation of Urban Scenes for Vehicle Visual Sensors by Optimized DCNN Algorithm" , Applied Sciences, 2022

- [9]. Bing Cheng, Chao Zhang, Xiaojuan Ding, Xiaopei Wu. "Convolutional neural network implementation for eye movement recognition based on video", 2018 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC), 2018
- [10]. Min Gao, Lang Lin, Richard O. Sinnott. "A Mobile Application for Plant Recognition through Deep Learning", 2017 IEEE 13th International Conference on e-Science (eScience), 2017
- [11]. Jingjing Zhang, Shuangjiu Xiao, Huichao Zhang, Lan Jiang. "Isolated Word Recognition with Audio Derivation and CNN", 2017 IEEE 19th International Conference on Tools with Artificial Intelligence (ICTAI), 2017

APPENDIX

Part. A. Code

Task 1: Import Libraries

```
In [1]: import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import os
%matplotlib inline

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Input, Dropout, Flatten, Conv2D
from tensorflow.keras.layers import BatchNormalization, Activation, MaxPooling2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau

from IPython.display import SVG, Image
from livelossplot.tf_keras import PlotLossesCallback
import tensorflow as tf
print("Tensorflow version:", tf.__version__)

Tensorflow version: 2.8.0
```

Task 2: Plot Sample Images

```
In [2]: # Define the path to the directory containing the images
data_path = "train"

# Loop through all the subdirectories in the data path
for root, dirs, files in os.walk(data_path):
    # Loop through all the files in each subdirectory
    for file in files:
        # Check if the file is not a JPG file
        if not file.endswith(".jpg"):
            # If it's not a JPG file, delete it
            os.remove(os.path.join(root, file))

# Define the path to the directory containing the images
test_path = "test"

# Loop through all the subdirectories in the data path
for root, dirs, files in os.walk(test_path):
    # Loop through all the files in each subdirectory
    for file in files:
        # Check if the file is not a JPG file
        if not file.endswith(".jpg"):
            # If it's not a JPG file, delete it
            os.remove(os.path.join(root, file))
```

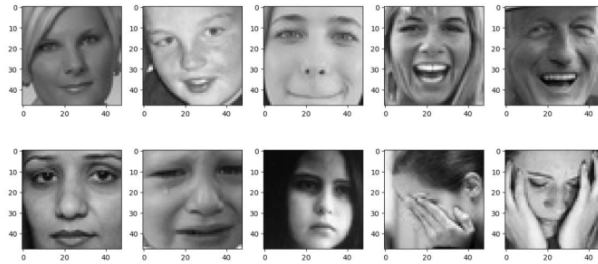
```
In [3]: import os
from tensorflow.keras.preprocessing.image import load_img, img_to_array

def plot_example_images(plt):
    img_size = 48
    plt.figure(0, figsize=(12,20))
    ctr = 0

    for expression in os.listdir("train/"):
        for i in range(1,6):
            ctr += 1
            plt.subplot(7,5,ctr)
            img = load_img("train/" + expression + "/" + os.listdir("train/" + expression)[i], target_size=(img_size, img_size))
            plt.imshow(img, cmap="gray")

    plt.tight_layout()
    return plt
```

```
In [4]: plot_example_images(plt).show()
```



```
In [5]: for expression in os.listdir("train/"):
        print(str(len(os.listdir("train/" + expression))) + " " + expression + " images")
```

```
7215 happy images
4830 sad images
4097 fear images
3171 surprise images
4965 neutral images
3995 angry images
436 disgust images
```

Task 3: Generate Training and Validation Batches

```
In [6]: img_size = 48
        batch_size = 64

        datagen_train = ImageDataGenerator(horizontal_flip=True)

        train_generator = datagen_train.flow_from_directory("train/",
                                                            target_size=(img_size,img_size),
                                                            color_mode='grayscale',
                                                            batch_size=batch_size,
                                                            class_mode='categorical',
                                                            shuffle=True)

        datagen_validation = ImageDataGenerator(horizontal_flip=True)
        validation_generator = datagen_validation.flow_from_directory("test/",
                                                                      target_size=(img_size,img_size),
                                                                      color_mode='grayscale',
                                                                      batch_size=batch_size,
                                                                      class_mode='categorical',
                                                                      shuffle=False)
```

```
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
```

Task 4: Create CNN Model

```
In [7]: # Initializing the CNN
        model = Sequential()

        # 1 - Convolution
        model.add(Conv2D(64, (3, 3), padding='same', input_shape=(48, 48, 1)))
        model.add(BatchNormalization())
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.25))

        # 2nd Convolution layer
        model.add(Conv2D(128, (5, 5), padding='same'))
        model.add(BatchNormalization())
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.25))

        # 3rd Convolution layer
        model.add(Conv2D(512, (3, 3), padding='same'))
        model.add(BatchNormalization())
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.25))
```

```

# 4th Convolution layer
model.add(Conv2D(512, (3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Flattening
model.add(Flatten())

# Fully connected layer 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(7, activation='softmax'))

opt = Adam(lr=0.0005)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

```

2023-05-02 18:19:06.062960: I tensorflow/core/platforms/gpu/gpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

Model: "sequential"

Task 5: Train and Evaluate Model

```

In [23]: %time

epochs = 15
steps_per_epoch = train_generator.n//train_generator.batch_size
validation_steps = validation_generator.n//validation_generator.batch_size

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1,
                               patience=2, min_lr=0.00001, mode='auto')
checkpoint = ModelCheckpoint('model_weights.h5', monitor='val_accuracy',
                             save_weights_only=True, mode='max', verbose=1)
callbacks = [PlotLossesCallback(), checkpoint, reduce_lr]

history = model.fit(
    x=train_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=epochs,
    validation_data = validation_generator,
    validation_steps = validation_steps,
    callbacks=callbacks
)

```

```
In [12]: import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
import itertools
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

#test_data, test_acc = model.evaluate_generator(test_generator, steps=len(test_generator))
#print("test acc: %.2f%%" % test_acc)
labels = (validation_generator.class_indices)
labels = dict((v, k) for k, v in labels.items())
x_test, y_test = validation_generator.__getitem__(0)
test_true = np.argmax(y_test, axis=-1)
test_pred = np.argmax(model.predict(x_test), axis=-1)
print(test_pred)
print("CNN Model Accuracy on test set: {:.4f}".format(accuracy_score(test_true, test_pred)))#输出模型测试平均值

preds = model.predict(x_test)
print(classification_report(test_true, test_pred)) # Output classification report (accuracy rate, regression rate, F1_score)

def plot_confusion_matrix(cm, classes, title='Confusion matrix', cmap=plt.cm.Blues):
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    plt.figure(figsize=(10, 10))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    fmt = '.2f'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment='center',
                 color="white" if cm[i, j] > thresh else "black")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()

# compute confusion matrix
cmf_matrix = confusion_matrix(test_true, test_pred)
sp.set_printoptions(precision=45)
# plot normalized confusion matrix
plot_confusion_matrix(cmf_matrix, classes=labels, title='Normalized confusion matrix')
plt.show()

[[ 0.55 0.4 0.05 0.14 5.34 0.0055 3.000055 1.045 0.0555 0.04
  3.5 0.5 0.0 0.5 5.4 2.5 0.2 3.0 0.3 3.5 0.5 0.6 0]
CNN Model Accuracy on test set: 0.4219
```

Part. B. video link

https://uwot.sd-my.sharepoint.com/personal/2203626_student_uwot.sd_ac_uk/_layouts/15/stream.aspx?id=%2Fpersonal%2F2203626%5Fstudent%5Fuwt.sd%5F%5F%2FDocuments%2FAttachments%2FPython%20code%2Eavi&ga=1

Experimental Report on Predicting Expression Datasets Using Convolutional Neural Network Models Defined in Keras Library

GRADEMARK REPORT

FINAL GRADE

52/100

GENERAL COMMENTS

Instructor

Abstract provided. Good. Could be more tightly worded.

Overall the report is well structured following accepted format. Grammar/style may needs a final edit to improve the written English a little.

Regarding referencing, please use inline citation to support arguments and assert definitions, etc.

If using other's figures then please cite these in the figure title/label.

A brief review of CNNs provided in the intro. Related work mainly focuses on theory/concepts. Quite detailed here.

Please use refs to support your arguments.

You could have repeated your experiments after adjusting parameters after analysing results.

Your confusion matrix has problems and really needs to be resolved. This limits the quality of your report as you don't have a full set of per-class results to analyse.

Refs used are, in general, appropriate.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6



Comment 1

Why? Explain.

PAGE 7

GRADING FORM: SMU GENERIC_GJ

SHUYAO JIANG

0

RESEARCH-INFORMED LITERATURE

KNOWLEDGE AND UNDERSTANDING

ANALYSIS

PRACTICAL APPLICATION AND DEP.

SKILLS FOR PROFESSIONAL PRAC.