机器学习期末大作业之人脸关键点检测

1. 题目描述

使用（96，96）的灰度图进行15个人脸关键点的预测。

1. 运行环境及准备

Windows10，Anaconda-Spyder (Python 3.7)

1. 设计过程

为了查看效果，运行两个版本的程序。1号版本采用的数据集train.csv按照3:1的比例切分成train1.csv和test1.csv，以便减少部分运算量，节约时间，最终可查看结果，2号版本则直接采用数据集train.csv作为训练集，test.csv作为测试集，最终输出训练好的模型文件和结果数据集2018211348.csv。此部分主要描述1号版本，2号版本仅为稍作修改的1号版本，无需过多赘述，最终代码为2号版本。

首先要搞清楚数据集和题目的情况，数据集由6000个96×96的灰度图像组成，对于每个图象，我们要找到15个关键点的位置坐标，形如（x, y）。

可以使用baseline里的prepare\_data读入数据集

import numpy as np

import pandas as pd

Train\_Dir = './data/train1.csv'

Test\_Dir = './data/test1.csv'

train\_data = pd.read\_csv(Train\_Dir)

test\_data = pd.read\_csv(Test\_Dir)

# use the previous value to fill the missing value

train\_data.fillna(method='ffill', inplace=True)

# preparing training data

imga = []

for i in range(len(train\_data)):

img = train\_data['Image'][i].split(' ')

img = ['0' if x == '' else x for x in img]

imga.append(img)

image\_list = np.array(imga, dtype='float')

X\_train = image\_list.reshape(-1, 96, 96, 1)

# preparing training label

training = train\_data.drop('Image', axis=1)

y\_train = []

for i in range(len(train\_data)):

y = training.iloc[i, 1:]

y = (y - 48) / 48

y\_train.append(y)

y\_train = np.array(y\_train, dtype='float')

# preparing test data

timga = []

for i in range(len(test\_data)):

timg = test\_data['Image'][i].split(' ')

timg = ['0' if x == '' else x for x in timg]

timga.append(timg)

timage\_list = np.array(timga, dtype='float')

X\_test = timage\_list.reshape(-1, 96, 96, 1)

然后使用utils查看已经成功读入的图像

import matplotlib.pyplot as plt

import numpy as np

def show\_result(image, points):

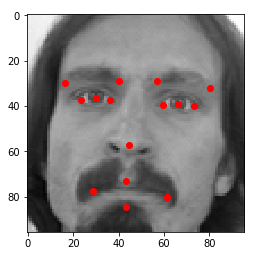
plt.imshow(image, cmap='gray')

for i in range(15):

plt.plot(points[2\*i], points[2\*i + 1], 'ro')

plt.show()

show\_result(X\_train[0].reshape(96,96), y\_train[0])



但是由于数据过于大，需要将数据进行归一化，需将其中的

timage\_list = np.array(timga, dtype='float')

改为

timage\_list = np.array(timga, dtype='float') /255

并新增加语句 y = (y - 48) / 48 将y的值控制在[-1, 1]之间

考虑到时间成本开销和PC的算力，采用CNN训练模型，其中包含卷积层，池化层和全连接层。

卷积层负责对图片做卷积运算，提取特征，并且使用relu激活函数来引入非线性因数。

池化层相当于压缩图片，例如max pool就是取设定矩阵中最大的值作为输出

通过卷积和池化，然后得到了众多特征，稠密层的每一个节点都与这些特征节点相连构成稠密层(全连接层)。稠密层的作用就是分类。简单的说就是每一个特征节点手里握着一定的权重来决定输入是属于那个分类，最终全部特征的权重共同决定了输入所属分类的权重或概率。

构建由三个卷积层和两个全连接层组成的CNN，每次卷积层后有2×2的max pool，filter数量为32、64、128，总神经元数量为500.

from keras.models import model\_from\_json

from keras.models import Sequential

from keras.layers import Dense, Activation

from keras.layers import Conv2D, MaxPool2D, Flatten

from keras.optimizers import SGD

from matplotlib import pyplot as plt

from pandas.io.parsers import read\_csv

from sklearn.utils import shuffle

import os

import random

def CNN(dim=9216):

"""

构建CNN，输入为整数/元组

返回为编译好的模型

"""

model = Sequential()

model.add(Conv2D(32, (3, 3), input\_shape=dim)) # CONV1

model.add(Activation('relu'))

model.add(MaxPool2D(pool\_size=(2, 2))) # FC1

model.add(Conv2D(64, (2, 2))) # Conv2

model.add(Activation('relu'))

model.add(MaxPool2D(pool\_size=(2, 2))) # FC2

model.add(Conv2D(128, (2, 2))) # Conv3

model.add(Activation('relu'))

model.add(MaxPool2D(pool\_size=(2, 2))) # FC3

model.add(Flatten())

model.add(Dense(500))

model.add(Activation('relu'))

model.add(Dense(500))

model.add(Activation('relu'))

model.add(Dense(30)) # 30 coords

sgd = SGD(lr=0.01, momentum=0.9, nesterov=True)

model.compile(loss='mean\_squared\_error', optimizer=sgd)

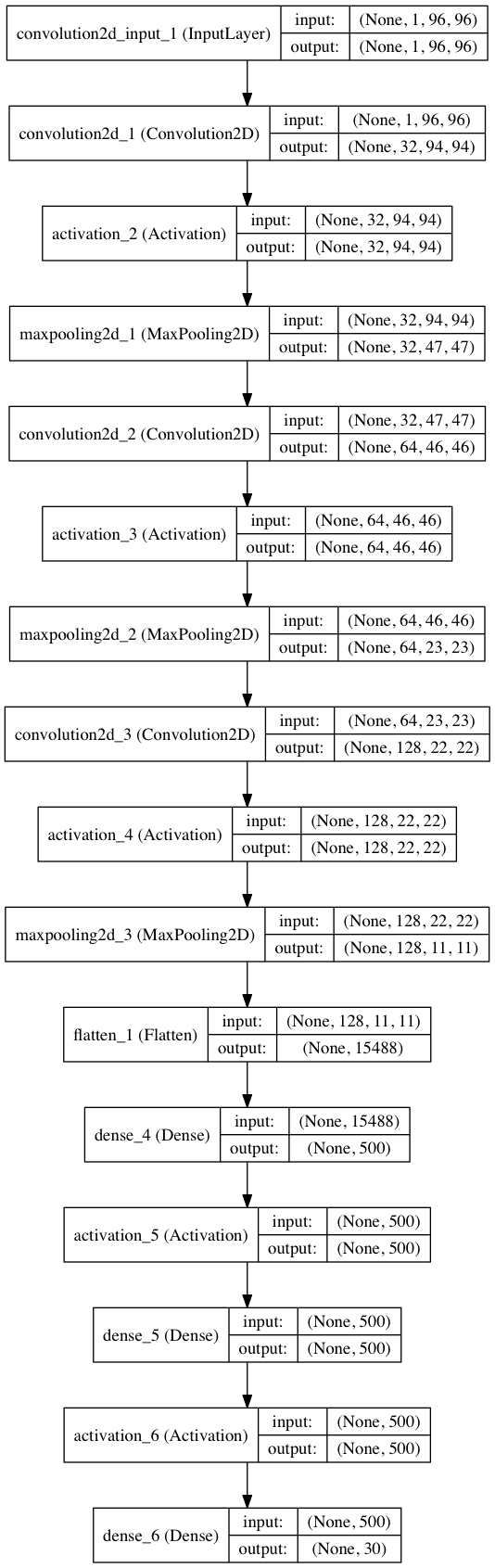
return model

modelCNN = CNN(dim=(96, 96, 1))

modelCNN.summary()

histCNN = modelCNN.fit(X\_train, y\_train, epochs=100, validation\_split=0.2)

模型的体系结构如图所示：



得到训练好的模型后可进行对测试集的预测，同时可绘制loss的图像以及最终结果的图像。需要注意的是，需要将已经进行标准化的图像进行复原。

def normlabel(y, reverse=False):

if reverse:

return y\*48 + 48

else:

return (y - 48) / 48

def histplot(hist, save=None, show=True):

"""plotting loss function

Arguments:

hist {keras.callbacks.History} -- hist = model.fit

Keyword Arguments:

save {str} -- filename if to save (default: {None})

show {bool} -- show plot (default: {True})

"""

plt.figure()

plt.plot(hist.history['loss'], linewidth=2, label='train')

plt.plot(hist.history['val\_loss'], linewidth=2,label='valid set')

plt.xlabel('epoch')

plt.ylabel('loss')

# plt.ylim(1e-3, 1e-2)

plt.yscale('log')

plt.grid() # show metrics(0, 20, 40...)

plt.legend(loc='best')

figure = plt.gcf()

if save != None:

figure.savefig(save, dpi=300)

print(save, 'saved.')

if show:

plt.show()

def predplot(xtest, ypred, save=None, show=True):

"""plotting pred outcome

Arguments:

xtest {np.array} -- test img

ypred {np.array} -- pred label

Keyword Arguments:

save {str} -- filename if to save (default: {None})

show {bool} -- show plot (default: {True})

"""

total = xtest.shape[0]

start = random.randint(0, total-16)

print('image',start,'to',start+16)

fig = plt.figure(figsize=(12, 12))

# fig.subplots\_adjust(left=0, right=0.5, bottom=0, top=0.5, hspace=0.05, wspace=0.05)

for i in range(start, start+16):

x, y = xtest[i], ypred[i]

img = x.reshape(96, 96)

axis = fig.add\_subplot(4, 4, i-start+1, xticks=[], yticks=[])

axis.imshow(img, cmap='gray') # show image

axis.scatter(normlabel(y[0::2], reverse=True), # show lanmark

normlabel(y[1::2], reverse=True), marker='x', s=10)

figure = plt.gcf()

if save != None: # deprecated

figure.savefig(save, dpi=300)

print(save, 'saved.')

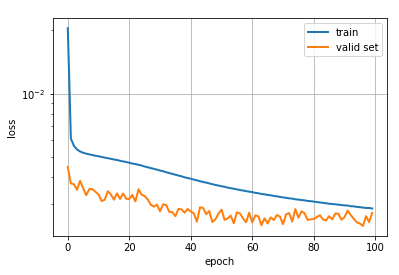
if show:

plt.show()

histplot(histCNN, save='CNN-test.png', show=True)

y\_pred = modelCNN.predict(X\_test)

predplot(X\_test, y\_pred, save='pred-CNN.png', show=True)





当然，还可以进一步优化相关参数，使得模型达到最好的效果。

最终保存预测结果以及模型，其中模型分为结构和参数两个部分保存。

def export(pred\_points, filename):

"""

:param pred\_points: result from your model use test.csv

:return:

"""

submission\_data = pd.DataFrame(pred\_points)

submission\_data.to\_csv(filename, index=False)

y\_pred1=normlabel(y\_pred, reverse=True)

export(y\_pred1, '2018211348.csv')

def savemodel(model, name, toprint=True):

"""save model to file

Arguments:

model {keras.models.Sequential} -- keras model

Keyword Arguments:

name {str} -- file name to save

"""

json\_string = model.to\_json()

open(name+'\_architecture.json', 'w').write(json\_string) # structure

model.save\_weights(name+'\_weights.h5') # weights

if toprint:

print('Structure:', name+'\_architecture.json')

print('Weights:', name+'\_weights.h5')

savemodel(modelCNN, name='model/CNN')