|  |
| --- |
|  |
| # Part 1 - Building the CNN |
|  | #importing the Keras libraries and packages |
|  | from keras.models import Sequential |
|  | from keras.layers import Convolution2D |
|  | from keras.layers import MaxPooling2D |
|  | from keras.layers import Flatten |
|  | from keras.layers import Dense, Dropout |
|  | from keras import optimizers |
|  |  |
|  | # Initialing the CNN |
|  | classifier = Sequential() |
|  |  |
|  | # Step 1 - Convolution Layer |
|  | classifier.add(Convolution2D(32, 3, 3, input\_shape = (64, 64, 3), activation = 'relu')) |
|  |  |
|  | #step 2 - Pooling |
|  | classifier.add(MaxPooling2D(pool\_size =(2,2))) |
|  |  |
|  | # Adding second convolution layer |
|  | classifier.add(Convolution2D(32, 3, 3, activation = 'relu')) |
|  | classifier.add(MaxPooling2D(pool\_size =(2,2))) |
|  |  |
|  | #Adding 3rd Concolution Layer |
|  | classifier.add(Convolution2D(64, 3, 3, activation = 'relu')) |
|  | classifier.add(MaxPooling2D(pool\_size =(2,2))) |
|  |  |
|  |  |
|  | #Step 3 - Flattening |
|  | classifier.add(Flatten()) |
|  |  |
|  | #Step 4 - Full Connection |
|  | classifier.add(Dense(256, activation = 'relu')) |
|  | classifier.add(Dropout(0.5)) |
|  | classifier.add(Dense(10, activation = 'softmax')) |
|  |  |
|  | #Compiling The CNN |
|  | classifier.compile( |
|  | optimizer = 'adam', |
|  | loss = 'categorical\_crossentropy', |
|  | metrics = ['accuracy']) |
|  |  |
|  | #Part 2 Fittting the CNN to the image |
|  | from keras.preprocessing.image import ImageDataGenerator |
|  | train\_datagen = ImageDataGenerator( |
|  | rescale=1./255, |
|  | shear\_range=0.2, |
|  | zoom\_range=0.2, |
|  | horizontal\_flip=True) |
|  |  |
|  | test\_datagen = ImageDataGenerator(rescale=1./255) |
|  |  |
|  | training\_set = train\_datagen.flow\_from\_directory( |
|  | 'Data/train', |
|  | target\_size=(64, 64), |
|  | batch\_size=32, |
|  | class\_mode='categorical') |
|  |  |
|  | test\_set = test\_datagen.flow\_from\_directory( |
|  | 'Data/test', |
|  | target\_size=(64, 64), |
|  | batch\_size=32, |
|  | class\_mode='categorical') |
|  |  |
|  | model = classifier.fit\_generator( |
|  | training\_set, |
|  | steps\_per\_epoch=100, |
|  | epochs=100, |
|  | validation\_data = test\_set, |
|  | validation\_steps = 6500 |
|  | ) |
|  |  |
|  | #Saving the model |
|  | import h5py |
|  | classifier.save('Trained\_Model.h5') |
|  |  |
|  | print(model.history.keys()) |
|  | import matplotlib.pyplot as plt |
|  |  |
|  | # summarize history for accuracy |
|  | plt.plot(model.history['acc']) |
|  | plt.plot(model.history['val\_acc']) |
|  | plt.title('model accuracy') |
|  | plt.ylabel('accuracy') |
|  | plt.xlabel('epoch') |
|  | plt.legend(['train', 'test'], loc='upper left') |
|  | plt.show() |
|  |  |
|  | # summarize history for loss |
|  | plt.plot(model.history['loss']) |
|  | plt.plot(model.history['val\_loss']) |
|  | plt.title('model loss') |
|  | plt.ylabel('loss') |
|  | plt.xlabel('epoch') |
|  | plt.legend(['train', 'test'], loc='upper left') |
|  | plt.show() |
|  |  |
|  |  |