|  |
| --- |
| #!/usr/bin/env python3 |
|  | # -\*- coding: utf-8 -\*- | | | | | |
|  | """ | | | | | |
|  | Created on Thu Jun 12 13:08:27 2018 | | | | | |
|  |  | | | | | |
|  | @author: keyur-r | | | | | |
|  | """ | | | | | |
|  |  | | | | | |
|  | # CNN classifier | | | | | |
|  |  | | | | | |
|  | # Building architecture of our CNN classifier | | | | | |
|  | import keras | | | | | |
|  | 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 | | | | | |
|  |  | | | | | |
|  | # Initialising the CNN | | | | | |
|  | classifier = Sequential() | | | | | |
|  |  | | | | | |
|  | # Step - 1 Convolution | | | | | |
|  | classifier.add(Convolution2D( | | | | | |
|  | 16, 3, 3, input\_shape=(28, 28, 3), activation='relu')) | | | | | |
|  |  | | | | | |
|  | # Step - 2 Pooling | | | | | |
|  | classifier.add(MaxPooling2D(pool\_size=(2, 2))) | | | | | |
|  |  | | | | | |
|  | classifier.add(Convolution2D(32, 3, 3, activation='relu')) | | | | | |
|  | classifier.add(MaxPooling2D(pool\_size=(2, 2))) | | | | | |
|  |  | | | | | |
|  | # Step - 3 Flattening | | | | | |
|  | classifier.add(Flatten()) | | | | | |
|  |  | | | | | |
|  | # Step - 4 Full connection -> First layer input layer then hidden layer | | | | | |
|  | # and last softmax layer | | | | | |
|  | classifier.add(Dense(56, activation='relu', kernel\_initializer='uniform')) | | | | | |
|  | classifier.add(Dense(3, activation='softmax', kernel\_initializer='uniform')) | | | | | |
|  |  | | | | | |
|  | # Compiling the CNN | | | | | |
|  | classifier.compile( | | | | | |
|  | optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy']) | | | | | |
|  |  | | | | | |
|  |  | | | | | |
|  | # Image Preprocessing | | | | | |
|  | 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( | | | | | |
|  | 'shapes/train', target\_size=(28, 28), batch\_size=1, class\_mode='categorical') | | | | | |
|  | #X\_images, y\_labels = training\_set.filenames, training\_set.classes | | | | | |
|  | test\_set = test\_datagen.flow\_from\_directory(   |  | | --- | | Logging the training of models | |  | from keras.callbacks import CSVLogger, EarlyStopping | | |  | csv\_logger = CSVLogger('log.csv', append=True, separator=';') | | |  | early\_stopping\_monitor = EarlyStopping(patience=5) | | |  |  | | |  | steps\_per\_epoch = len(training\_set.filenames) # 300 | | |  | validation\_steps = len(test\_set.filenames) # 90 | | |  |  | | |  | model\_info = classifier.fit\_generator(training\_set, steps\_per\_epoch=steps\_per\_epoch, epochs=25, validation\_data=test\_set, | | |  | validation\_steps=validation\_steps, callbacks=[csv\_logger, early\_stopping\_monitor]) | | |  |  | | |  | classifier.save("drawing\_classification.h5") | | |  |  | | |  | # plot model history after each epoch | | |  | from visulization import plot\_model\_history | | |  | plot\_model\_history(model\_info) | | | #!/usr/bin/env python3 | | | |  | | | # -\*- coding: utf-8 -\*- | | |  | | | """ | | |  | | | Created on Thu Jun 12 12:01:09 2018 | | |  | | |  | | |  | | | @author: keyur-r | | |  | | | """ | | |  | | |  | | |  | | | # Image Preprocessing for train, test and validation sets | | |  | | |  | | |  | | | import os | | |  | | | import random | | |  | | | import glob | | |  | | |  | | |  | | |  | | |  | | | def prepare\_test\_data(n): | | |  | | | base\_path = "shapes" | | |  | | | f1 = random.sample(glob.glob(os.path.join(base\_path, "test/circles") + "/\*"), n) | | |  | | | f2 = random.sample(glob.glob(os.path.join(base\_path, "test/squares") + "/\*"), n) | | |  | | | f3 = random.sample(glob.glob(os.path.join(base\_path, "test/triangles") + "/\*"), n) | | |  | | | for c in f1: | | |  | | | os.remove(c) | | |  | | | for s in f2: | | |  | | | os.remove(s) | | |  | | | for t in f3: | | |  | | | os.remove(t) | | |  | | |  | | |  | | | def prepare\_validation\_data(n): | | |  | | | base\_path = "shapes" | | |  | | | f1 = random.sample(glob.glob(os.path.join(base\_path, "validation/circles") + "/\*"), n) | | |  | | | f2 = random.sample(glob.glob(os.path.join(base\_path, "validation/squares") + "/\*"), n) | | |  | | | f3 = random.sample(glob.glob(os.path.join(base\_path, "validation/triangles") + "/\*"), n) | | |  | | | for c in f1: | | |  | | | os.remove(c) | | |  | | | for s in f2: | | |  | | | os.remove(s) | | |  | | | for t in f3:   |  | | --- | | os.remove(t) | |  |  | |  | # n = number of samples to remove from each categorical folder | |  | prepare\_test\_data(70) | |  | prepare\_validation\_data(40) | | | | | | | | |
| !/usr/bin/env python | | | | |
|  | | | | |  | |
|  | | | | | # Giving class label from probabilities | |
|  | | | | | from keras.utils import np\_utils | |
|  | | | | | import numpy as np | |
|  | | | | |  | |
|  | | | | |  | |
|  | | | | | def probas\_to\_classes(y\_pred): | |
|  | | | | | if len(y\_pred.shape) > 1 and y\_pred.shape[1] > 1: | |
|  | | | | | return categorical\_probas\_to\_classes(y\_pred) | |
|  | | | | | return np.array([1 if p > 0.5 else 0 for p in y\_pred]) | |
|  | | | | |  | |
|  | | | | |  | |
|  | | | | | def categorical\_probas\_to\_classes(p): | |
|  | | | | | return np.argmax(p, axis=1) | |
| #!/usr/bin/env python3 | | | |
|  | | | | # -\*- coding: utf-8 -\*- | | |
|  | | | | """ | | |
|  | | | | Created on Thu Jun 12 14:01:56 2018 | | |
|  | | | |  | | |
|  | | | | @author: keyur-r | | |
|  | | | | """ | | |
|  | | | |  | | |
|  | | | | import pandas as pd | | |
|  | | | | import numpy as np | | |
|  | | | | import cv2 | | |
|  | | | | import argparse | | |
|  | | | |  | | |
|  | | | | from keras.models import load\_model | | |
|  | | | | from keras.preprocessing.image import ImageDataGenerator | | |
|  | | | | from sklearn.metrics import confusion\_matrix, accuracy\_score | | |
|  | | | |  | | |
|  | | | | from myutil import probas\_to\_classes | | |
|  | | | |  | | |
|  | | | |  | | |
|  | | | | # Loading and compiling presaved trained CNN | | |
|  | | | | model = load\_model('drawing\_classification.h5') | | |
|  | | | |  | | |
|  | | | | label = {0: "Circle", 1: "Square", 2: "Triangle"} | | |
|  | | | |  | | |
|  | | | |  | | |
|  | | | | def predict\_one(file\_name): | | |
|  | | | |  | | |
|  | | | | img = cv2.imread(file\_name) | | |
|  | | | | img = cv2.resize(img, (28, 28)) | | |
|  | | | | img = np.reshape(img, [1, 28, 28, 3]) | | |
|  | | | | classes = model.predict\_classes(img)[0] | | |
|  | | | | category = label[classes] | | |
|  | | | | print("\nAnd {1} is the {0}".format(category, file\_name)) | | |
|  | | | | # return category | | |
|  | | | |  | | |
|  | | | |  | | |
|  | | | | def predict\_dataset(input\_dir): | | |
|  | | | | test\_datagen = ImageDataGenerator(rescale=1. / 255) | | |
|  | | | | test\_generator = test\_datagen.flow\_from\_directory("shapes/test", | | |
|  | | | | target\_size=(28, 28), | | |
|  | | | | color\_mode="rgb", | | |
|  | | | | shuffle=False, | | |
|  | | | | class\_mode='categorical', | | |
|  | | | | batch\_size=1) | | |
|  | | | | filenames = test\_generator.filenames | | |
|  | | | | nb\_samples = len(filenames) | | |
|  | | | | predict = model.predict\_generator(test\_generator, steps=nb\_samples) | | |
|  | | | | return predict, test\_generator | | |
|  | | | |  | | |
|  | | | |  | | |
|  | | | | def main(): | | |
|  | | | | # Instantiate the parser | | |
|  | | | | parser = argparse.ArgumentParser() | | |
|  | | | | parser.add\_argument( | | |
|  | | | | '--testdata', help='Classify images on test data', action='store\_true') | | |
|  | | | | parser.add\_argument( | | |
|  | | | | '--validationdata', help='Classify images on test data', action='store\_true') | | |
|  | | | |  | | |
|  | | | | parser.add\_argument('--image', help='Input your image file name') | | |
|  | | | | args = parser.parse\_args() | | |
|  | | | |  | | |
|  | | | | on\_dataset = False | | |
|  | | | | if args.testdata: | | |
|  | | | | print("Classify images on test dataset") | | |
|  | | | | on\_dataset = True | | |
|  | | | | input\_dir = "shapes/test" | | |
|  | | | | if args.validationdata: | | |
|  | | | | print("Classify images on validation dataset") | | |
|  | | | | on\_dataset = True | | |
|  | | | | input\_dir = "shapes/validation" | | |
| if on\_dataset: | | |
|  | | | predict, test\_generator = predict\_dataset(input\_dir) | | |
|  | | | y\_pred = probas\_to\_classes(predict) | | |
|  | | | y\_true = test\_generator.classes | | |
|  | | | X\_images = test\_generator.filenames | | |
|  | | | cm = confusion\_matrix(y\_true, y\_pred) | | |
|  | | | ac = accuracy\_score(y\_true, y\_pred) | | |
|  | | | for ele in list(zip(X\_images, y\_true, y\_pred)): | | |
|  | | | print(ele) | | |
|  | | | else: | | |
|  | | | file\_name = args.image | | |
|  | | | predict\_one(file\_name) | | |
|  | | |  | | |
|  | | | if \_\_name\_\_ == '\_\_main\_\_': | | |
|  | | | main() | | |
| import matplotlib.pyplot as plt | |
|  | | import numpy as np | | | | |
|  | |  | | | | |
|  | |  | | | | |
|  | | def plot\_model\_history(model\_history): | | | | |
|  | | fig, axs = plt.subplots(1, 2, figsize=(15, 5)) | | | | |
|  | | # summarize history for accuracy | | | | |
|  | | axs[0].plot(range(1, len(model\_history.history['acc']) + 1), | | | | |
|  | | model\_history.history['acc']) | | | | |
|  | | axs[0].plot(range(1, len(model\_history.history['val\_acc']) + 1), | | | | |
|  | | model\_history.history['val\_acc']) | | | | |
|  | | axs[0].set\_title('Model Accuracy') | | | | |
|  | | axs[0].set\_ylabel('Accuracy') | | | | |
|  | | axs[0].set\_xlabel('Epoch') | | | | |
|  | | axs[0].set\_xticks(np.arange(1, len(model\_history.history[ | | | | |
|  | | 'acc']) + 1), len(model\_history.history['acc']) / 10) | | | | |
|  | | axs[0].legend(['train', 'val'], loc='best') | | | | |
|  | | # summarize history for loss | | | | |
|  | | axs[1].plot(range(1, len(model\_history.history['loss']) + 1), | | | | |
|  | | model\_history.history['loss']) | | | | |
|  | | axs[1].plot(range(1, len(model\_history.history['val\_loss']) + 1), | | | | |
|  | | model\_history.history['val\_loss']) | | | | |
|  | | axs[1].set\_title('Model Loss') | | | | |
|  | | axs[1].set\_ylabel('Loss') | | | | |
|  | | axs[1].set\_xlabel('Epoch') | | | | |
|  | | axs[1].set\_xticks(np.arange(1, len(model\_history.history[ | | | | |
|  | | 'loss']) + 1), len(model\_history.history['loss']) / 10) | | | | |
|  | | axs[1].legend(['train', 'val'], loc='best') | | | | |
|  | | plt.show() | | | | |