simpson_procesamiento

October 24, 2022

1 Procesamiento de la base de datos

Este código sirve para cargar y procesar las imágenes de n personajes de Los Simpsons a través del dataset encontrado en la siguiente dirección: https://www.kaggle.com/jfgm2018/the-simpsons-dataset-compilation-49-characters

1.1 Importar librerías

[1]:
```
# Importar librerías que se usarán
import numpy as np
import os
import cv2
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder
import random
import pickle

%matplotlib inline
```

1.2 Cargado de imágenes de n clases

Una vez cargadas las librerías, se procede a seleccionar qué personajes van a ser seleccionados para su procesamiento, las etiquetas deben ser las mismas que la base de datos arroja en cada una de sus carpetas

[2]:
```
# Se eligen 10 personajes de Los Simpsons
targets=["apu_nahasapeemapetilon",
         "bart_simpson",
         "homer_simpson",
         "krusty_the_clown",
         "lisa_simpson",
         "maggie_simpson",
         "marge_simpson",
         "ned_flanders",
         "patty_bouvier",
         "sideshow_mel"]
```

Posteriormente se declara una función la cuál permitirá almacenar todas las imágenes de la carpeta seleccionada, teniendo como parámetros a modificar la ruta, el listado de los personajes y el tamaño

de las imágenes. Obsérvese que al ser cargadas las imágenes inmediatamente se redimensionan a un formato cuadrado de img_size x img_size y son almcacenadas en arreglos de Numpy así como se crea otro arreglo de Numpy de la misma primera dimensión del primer arreglo pero con etiquetas numéricas (0 a 9) de su correspondiente clase del listado targets. Finalmente, la función al ser llamada imprime un listado de las clases de personajes que se cargaron y la cantidad de archivos de imagen encontradas por clase.

[3]:
```
def load(path,targets,img_size):
  print("Loading files...")
  lista_img = []
  lista_labels = []
  for ind,value in enumerate(targets):
      path2= path+value
      size = len(lista_img)
      for img in os.listdir(path2):
        img = cv2.imread(os.path.join(path2,img))
        img_resize = cv2.resize(img,(img_size,img_size))
        lista_img.append(img_resize)
        lista_labels.append(ind)
      print("Class:",value,"Files:",len(lista_img)-size)
  print("Loaded all files\n")
  return np.array(lista_img), np.array(lista_labels)
```

Se procede a cargar las imágenes de entrenamiento según la partición que arroja la base de datos de Kaggle. Únicamente se requiere definir la ruta donde se encuentra la carpeta de entrenamiento y el tamaño de las imágenes.

[4]:
```
train_path = "C:/Users/omar_/Documents/RNAA_prj/Tarea 2/simpsons_dataset/train/"
x_train, y_train = load(train_path,targets,img_size=80)
```

```
Loading files…
Class: apu_nahasapeemapetilon Files: 613
Class: bart_simpson Files: 2562
Class: homer_simpson Files: 4128
Class: krusty_the_clown Files: 1206
Class: lisa_simpson Files: 2383
Class: maggie_simpson Files: 1371
Class: marge_simpson Files: 1810
Class: ned_flanders Files: 1454
Class: patty_bouvier Files: 72
Class: sideshow_mel Files: 40
Loaded all files
```

Obsérvese que de esta forma logramos generar los arreglos: x_train y y_train. Se comprueba que la primera dimensión de x_train y y_train son iguales.

[5]:
```
x_train.shape
```

[5]: (15639, 80, 80, 3)

[6]:
```
y_train.shape
```

[6]: (15639,)

Se repite el proceso anterior pero con los datos de entrenamiento.

[7]:
```
test_path = "C:/Users/omar_/Documents/RNAA_prj/Tarea 2/simpsons_dataset/test/"
x_test, y_test = load(test_path,targets,img_size=80)
```

```
Loading files…
Class: apu_nahasapeemapetilon Files: 50
Class: bart_simpson Files: 261
Class: homer_simpson Files: 306
Class: krusty_the_clown Files: 50
Class: lisa_simpson Files: 222
Class: maggie_simpson Files: 162
Class: marge_simpson Files: 307
Class: ned_flanders Files: 49
Class: patty_bouvier Files: 29
Class: sideshow_mel Files: 23
Loaded all files
```

[8]:
```
x_test.shape
```

[8]: (1459, 80, 80, 3)

[9]:
```
y_test.shape
```

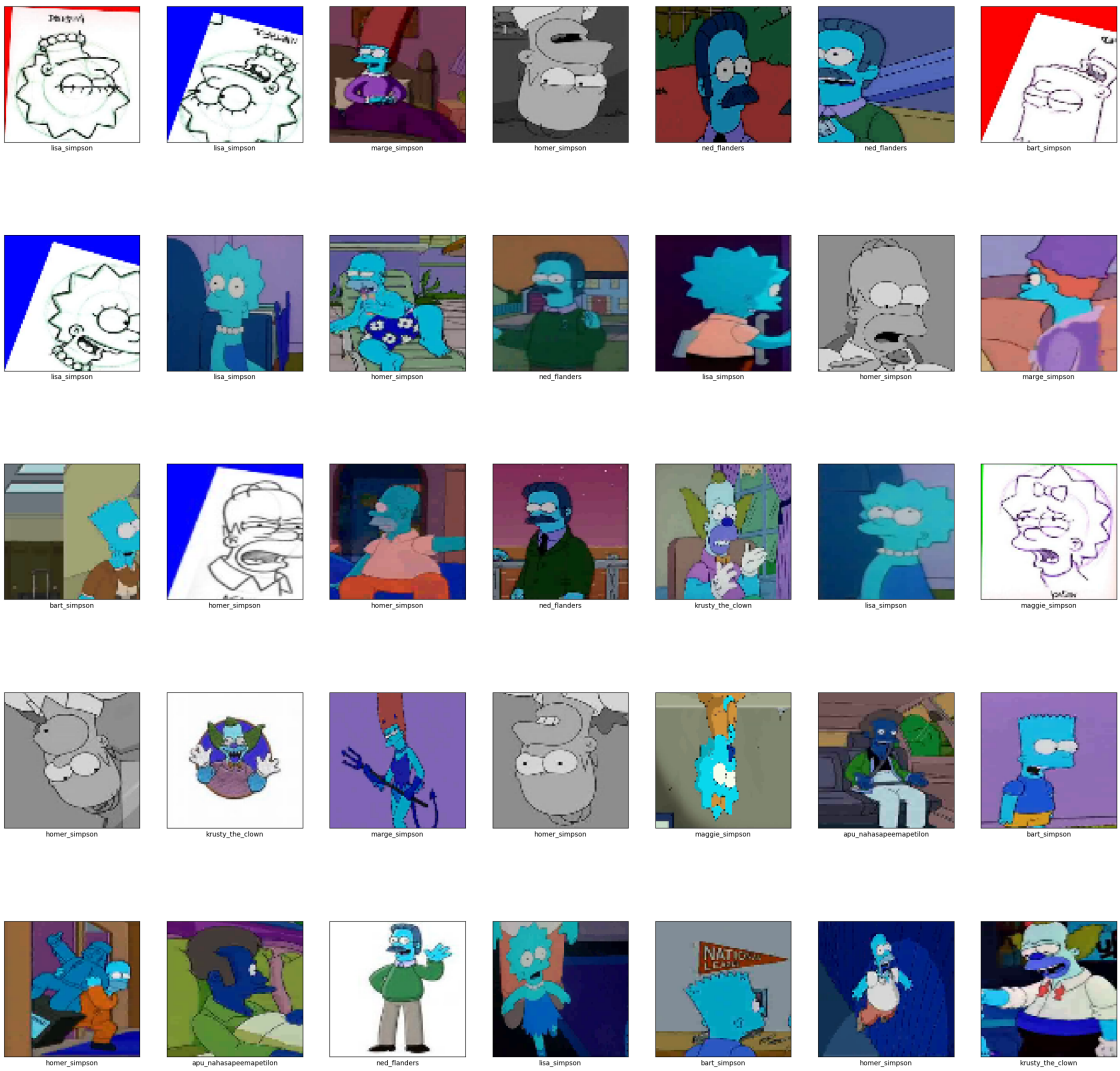
[9]: (1459,)

1.3 Gráficar imágenes cargadas

La siguiente función sirve para graficar rows x columns imágenes de forma aleatoria del arreglo imgs, seleccionando valores de 0 a size(imgs).

[10]:
```
def graficar(imgs,rows,columns):
  plt.figure(figsize=(30,30))
  for i in range(rows*columns):
    img_i = random.randint(0,imgs.shape[0])
    plt.subplot(rows,columns,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(imgs[img_i])
    plt.xlabel(targets[y_train[img_i]])
  plt.show()
```

[11]:
```
# Revisamos cómo son las imágenes
graficar(x_train,5,7)
```



Las imágenes han sufrido un cambio de espacio de color por la forma en la que fueron cargadas; sin embargo, aunque no pude resolver éste problema para ésta tarea, no representa un problema para su posterior clasificación (sólo para la visualización) puesto que ésta transformación es realizada a todas las imágenes y por tanto, mantienen su identidad.

1.4 Preparación de los datos para su clasificación

Ahora continuamos modificando los arreglos de y_xxx de valores cuantitativos (0 a 9) por una codificación one-hot.

[12]:
```
#Cambiamos la etiqueta categórica/discreta por una one-hot
onehot_encoder = OneHotEncoder(sparse=False)
```

```
#Para las etiquetas de train
y_train = y_train.reshape(len(y_train), 1)
y_train_onehot = onehot_encoder.fit_transform(y_train)

#Para las etiquetas de test
y_test = y_test.reshape(len(y_test), 1)
y_test_onehot = onehot_encoder.fit_transform(y_test)
```

Para comprobar que las etiquetas ahora se encuentran en forma de vectores de size(x_xxx) x size(targets) se ejecuta la siguiente instrucción.

[13]:
```
#Revisamos en que dimensiones quedaron nuestros vectores one hot
y_train_onehot.shape
```

[13]:
```
(15639, 10)
```

1.5 Almacenamiento de los arreglos

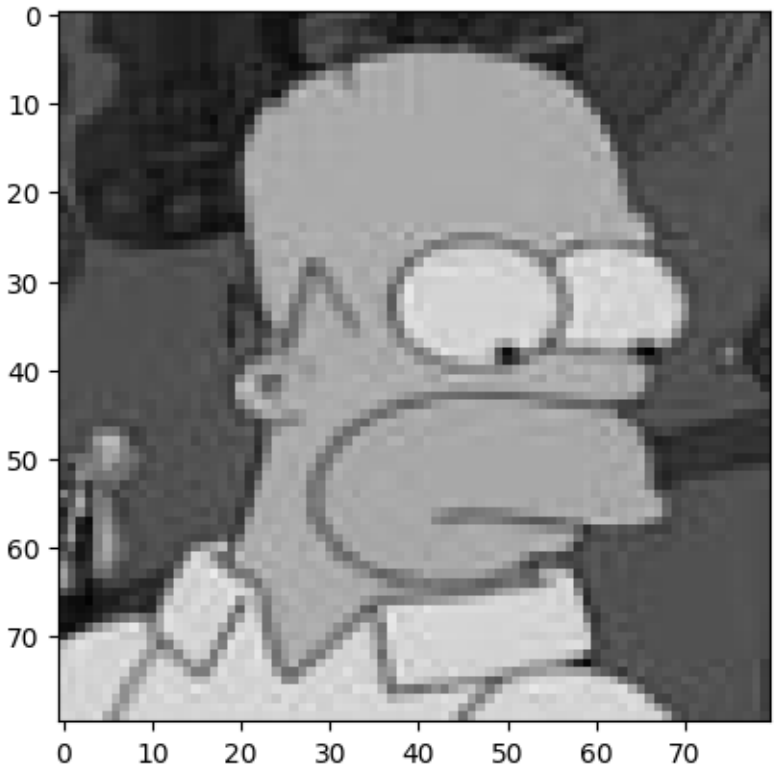
Honestamente la siguiente función fue desarrollada por Fernando Javier Aguilar Canto; sin embargo, creo entender el cómo funciona y lo explicaré a lo mayor detalle posible en los comentarios.

[16]:
```
# Función para almacenar los arreglos en archivos .pickle
def create_datsets_and_labels(x_train,y_train,x_test, y_test, nBatches,␣
 ↪test_size=0.2):
  '''
  Parámetros:
  x_train, y_train : Arreglos numpy de igual primera dimensión, contiene los␣
 ↪datos que servirán para entrenamiento.
  x_test, y_test : Arreglos numpy de igual primera dimensión, contiene los␣
 ↪datos que servirán para testeo.
  nBatches : Número de lotes que se van a trabajar.
  test_size : Tamaño de proporción del testeo vs el entrenamiento.
                (Es decir, el test_size permitirá tomar mayor o menos imágenes␣
 ↪que las que la base de datos declaró en su almacenamiento.)
  '''

  # Se declara la ruta en donde se encuentra una carpeta en la que se␣
 ↪almacenarán los datos.
  path= "C:/Users/omar_/Documents/RNAA_prj/Tarea 2/simpsons/"

  #Unimos los datasets, para acomodarlos según nuestra propia proporción.
  data = np.concatenate((x_train,x_test), axis=0)
  labels = np.concatenate((y_train, y_test), axis=0)


  # Se calcula la cantidad de datos que debe tener cada lote (Batche)
  lenBatches = int(len(data)/nBatches)
```

```
print("Se haran lotes con: ", lenBatches, " ejemplos")

# Inicializan listas en las que se agregarán las imágenes.
test_x = []
test_y = []
i=0
for n in range(nBatches):
  start = i*lenBatches
  end = start+lenBatches
  testing_size = int(test_size*lenBatches)
  f_data = data[start:end]
  f_labels = labels[start:end]
  features = []

  for d in range(len(f_data)):
    features.append([f_data[d], f_labels[d]])

  # Revuelve de forma aleatoria los datos para su almacenamiento no secuencial.
  random.shuffle(features)
  features = np.array(features,dtype=object)
  print("Se revuelven los datos...")

  # Comprobar que al terminar la mezcla la imagen corresponda a su clase.
  imagendemo = features[0][0]
  print("label:", features[0][1])
  plt.imshow(imagendemo,cmap='gray')
  plt.show()

  train_x = list(features[:,0][:-testing_size]) #tomamos el dato
  train_y = list(features[:,1][:-testing_size]) #tomamos las etiquetas
  test_x += list(features[:,0][-testing_size:])
  test_y += list(features[:,1][-testing_size:])

  datx= np.array(train_x).astype("float32")
  print("datx.shape", datx.shape)

  # Almacena de forma binaria el archivo
  with open(path+'simpsons_set'+str(n)+'.pickle','wb') as f:
    pickle.dump([datx,np.array(train_y).astype("float32")],f)
  i+=1

# Almacena de forma binaria el archivo
with open(path+'simpsons_setTest.pickle','wb') as f:
  pickle.dump([np.array(test_x).astype("float32"),np.array(test_y).astype("float32")],f)
```

[17]:
```
# Guardamos la base de datos procesada.
create_datsets_and_labels(x_train,y_train_onehot,x_test, y_test_onehot, 1,␣
 ↪test_size=0.2)
```

```
Se haran lotes con:  17098  ejemplos
Se revuelven los datos…
label: [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```




```
datx.shape (13679, 80, 80, 3)
```