UAEM Universidad Autónoma del Estado de México

Materia:

Redes Neuronales

PROYECTO FINAL

CLASIFICADOR BINARIO (RECONOCIMIENTO DE PERSONAS)

Docente: Asdrúbal López Chau

Alumno:

Francisco Javier Miguel García

Gustavo Rodríguez Calzada

Fecha de Elaboración:

11 de diciembre de 2020

Breve explicación del programa:

Este programa funciona para el reconocimiento de personas, el usuario deberá ingresar una imagen que desee y el programa hará un procesamiento con esa imagen y dará como salida una respuesta si es o no un humano, todo es basándonos en clases ya vistas acerca del perceptrón, así como un poco de conocimiento de tratamiento de imágenes con open-cv ayudando al depuramiento de imágenes.

Para poder realizar este programa nos encontramos con un par d inconvenientes como el de que en un csv no podíamos meter muchos datos de entrenamiento, además de no saber los pasos adecuados para entrenar las neuronas con imágenes ya que solo habíamos visto ejemplos en clase los cuales solamente utilizamos un csv de entrenamiento ya hecho, pero en este caso nosotros teníamos que generar este, así que gracias a la basta documentación logramos terminar el programa de manera correcta

Explicación de código:

Cabe mencionar que para la realización de este proyecto lo dividimos en dos partes las cuales son:

- Entrenamiento.py
- Reconocimiento_de_Personas

Así haciendo que el código no fuese extenso y separáramos la parte del depuramiento de datos y finalmente otro en el reconocimiento, a continuación, aplicaremos la parte de Entrenamiento.py

```python
# -*- coding: utf-8 -*-
import cv2
import numpy as np


img = cv2.imread("/home/franck2407/Downloads/ProyectoFinal/ImagenesOriginalesO/unoH.jpg", 0)
primerF = cv2.resize(img, (100, 300))#(ancho->columnas)(largo->Renglones)
cv2.imwrite("/home/franck2407/Downloads/ProyectoFinal/ImagenesReducidasO/unoHReducida.jpg",primerF)
```

Bueno lo primero que hacemos es importar las librerías con las cuales trabajaremos

CV2 = Open-cv: Para el manejo de las imágenes

Numpy: para algunas opciones matemáticas que ocuparemos a continuación.

Una vez ya importadas las bibliotecas ahora con ayuda de open-cv abriremos las imágenes (a blanco y negro) que servirán como entrenamiento y para esto utilizaremos 12 imágenes en total, de las cuales TODAS son humanos (6 hombres y 6 mujeres) haciendo diversas posiciones (parado, corriendo, sentado, etc.)

Ya abiertas las imágenes las pasaremos a un tamaño para que todas estas imágenes de entrenamiento sean del mismo tamaño (100 de ancho y 300 de largo), esto para posteriormente ocuparlo

```
height, width = primerF.shape[0:2]
imR = cv2.getRotationMatrix2D((width/2, height/2),90,.30)
rot90 = cv2.warpAffine(primerF, imR, (width, height))
cv2.imwrite("/home/franck2407/Downloads/ProyectoFinal/ImagenesRotadas0/Imagen_90_G.jpg",rot90)
```

Antes de ocupar el tamaño ya dicho antes, primero haremos el primer filtro que es hacer que las imágenes tengan una rotación esto para que si la imagen que después se vaya a querer comprobar si es o no un humano no marque error si ve por ejemplo un humano sentado, pero de cabeza.

Repetimos esto para 90 grados, 180, 270 y finalmente 360 grados y todas estas imágenes las guardamos para después utilizarlas

```
img0 = im.open("/home/franck2407/Downloads/ProyectoFinal/ImagenesRotadas0/1/H/Imagen_90_G.jpg")
img1 = im.open("/home/franck2407/Downloads/ProyectoFinal/ImagenesRotadas0/1/H/Imagen_180_G.jpg")
img2 = im.open("/home/franck2407/Downloads/ProyectoFinal/ImagenesRotadas0/1/H/Imagen_270_G.jpg")
img3 = im.open("/home/franck2407/Downloads/ProyectoFinal/ImagenesRotadas0/1/H/Imagen_360_G.jpg")
```

Ahora ya trabajaremos con estas imágenes así que las abrimos

```
img00 = np.array(img0)
img11 = np.array(img1)
img22 = np.array(img2)
img33 = np.array(img3)
```

Las convertimos a un arreglo

```
img000 = img00.reshape([30000,1])
img111 = img11.reshape([30000,1])
img222 = img22.reshape([30000,1])
img333 = img33.reshape([30000,1])
```

Aplanamos este arreglo a un vector de [30000,1]

```
X0 = img000/255
X1 = img111/255
X2 = img222/255
X3 = img333/255
```

Dividimos el arreglo para mejor uso, y estas 4 variables las guardamos para después pasarlas al csv de entrenamiento de puras imágenes de humanos.

Hacemos el mismo procedimiento con las 48 imágenes (12 imágenes originales x 4 de imágenes rotadas = 48)

Ahora explicaremos el script de Reconocimiento_de_Personas.py

1.- En nuestro script “Reconocimiento_de_Personas.py” nos encontraremos con las siguientes líneas de código

```
# Importamos las bibliotecas
import cv2
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron
```

importamos las bibliotecas de

- cv2: Ocupamos esta biblioteca para acceder a la función de abrir y re dimensionar una imagen
- numpy: Para pasar una imagen a un arreglo
- pandas: Para abrir un conjunto de datos (csv)
- sklearn: para acceder a la función de train_test_split y a Perceptrón

2.- Ahora lo que se hizo después de importar las bibliotecas es lo siguiente:

- Abrir una imagen y convertirla a blanco y negro

```
# Leemos la imagen a predecir
img = cv2.imread("/home/gustavo/Descargas/ProyectoFinal/Imagnes_A_Predecir/SIHumanos/p7.jpg",0
```

- Para tener un tamaño “estándar”, todas las imágenes se re dimensionaron a 100x300

```
# Reeducimos la imagen a 100x300 pixeles para pasar a un arreglo de numpy
primerF = cv2.resize(img, (100, 300))#(ancho->columnas)(largo->Renglones)
```

- Ahora lo que se hizo es pasar dicha imagen a un arreglo de numpy

```
# Pasamos esta imagen ya redimencionada a un arreglo
img44 = np.array(primerF)
```

3.- Una vez teniendo la imagen en un arreglo, lo que se hizo con el método reshape es aplanarlo para tener un arreglo de 30,000(Filas) x 1(Columna). El número 30,000 es resultado de la operación 100x300 que es el tamaño “estándar” de nuestras imágenes

```
# Aplanamos el arreglo y lo hacemos de 30,000(Filas) x 1(Columna)
img444 = img44.reshape([30000,1]) #Tamaño dinámico
```

4.- Finalmente dividiremos cada fila de nuestra imagen ya aplanada y redimensionada con el número 255, esto con el fin de optimizar nuestro algoritmo. Ya que le resultará trabajar más rápido con números del 0 al 1, que con números de 0 a 255

```
# Lo dividimos para optimizar el funcionamiento
xf = img444/255
```

5.- Una vez teniendo nuestra imagen a predecir, abriremos el conjunto de datos (csv), con algunas imágenes ya registradas para el entrenamiento

```
# Leemos los datos de entrenamiento previamente hechos en "Entrenamiento.py"
datos = pd.read_csv("/home/gustavo/Descargas/ProyectoFinal/data/oficial.csv")
```

6.- En esta parte transponemos los datos del csv y el arreglo de la imagen que queremos predecir, esto es porque debemos tener los 30,000 datos como columnas, para que cada renglón o imagen, tenga su propia etiqueta, pero como el csv lo hicimos en Excel, solo nos permite un número máximo de columnas, por tanto, se colocaron los datos al revés, para que, al momento de manipularlos, solo cambiemos filas por columnas (Transponer)

```
# Transponemos los datos del csv y los datos de la imagen a predecir
datosFull = datos.T
imT = xf.T
```

7- Ahora vamos a separar las clases (X) que son cada una de las imágenes, con las etiquetas (Y) que son las salidas 1 o -1.

```
#Separamos las clase de las etiquetas
X = datosFull.iloc[:, 0:-1]
# Clase
Y = datosFull.iloc[:, -1]
```

8.- La función train_test_split, nos permite dividir el dataset (datos del csv) en dos datos, ya sean para datos de entrenamiento(75%) y validación del modelo(25%), para ello tomaremos como argumentos las variables X y Y (ya definidas anteriormente)
-En este caso con el parámetro test_size modificamos los datos y en vez de ocupar el 25% de los datos para pruebas, ocuparemos el 30% (test_size=0.3)
-Con random_state permite generar números aleatorios para la función

```
# Divide matrices en sub conjuntos de pruebas y trenes aleatorio
X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size=0.3, random_state=0)
```

9.- Lo que se hizo es usar el perceptrón, ya que solo lee los valores de entrada, los suma y de acuerdo a unos pesos sinópticos entra a una función de activación y genera un resultado

```
perceptron = Perceptron()
```

10.- Antes de finalizar llamaremos la función FIT. Que ajusta los pesos de acuerdo con los valores que le daremos, para que se logre una mayor precisión, y así entrenarlo.

```
perceptron.fit(X_train, y_train)
```

```
# Eficiencia del algoritmo
print(perceptron.score(X_test, y_test))

# Imprimimos si el valor es -1 0 1 dependiendo la salida
print("La predicción es: ", int(perceptron.predict(imT)))

# Para que el usuario entienda mas imprimimos la respuesta si es o no humano
if((int(perceptron.predict(imT))) == 1):
    print("Es un humano =D")
else:
    print("No es humano")
```

11.- FINALMENTE imprime en pantalla la eficiencia de nuestro algoritmo y hace la predicción de la imagen que anteriormente trabajamos