Reconocimiento de cactus con redes neuronales convolucionadas

Efren Lopez Jimenez

En este trabajo se hace uso de las redes neuronales convolucionadas para el reconocimiento de cactus, dicho trabajo está conformado por un dataset con fotografías que fueron tomadas en la reserva de la biósfera de Tehuacán-Teotitlán que comparten los estados Puebla y Oaxaca respecticvamente.

I. Introducción

Este trabajo abordan dos temas centrales sobre las redes neuronales convolucionada ((CNN) por sus siglas en inglés): La construcción del dataset y su implementación en una CNN, para este trabajo se realizaron capturas de imágenes en la reserva de la biósfera de Tehuacán-Teotitlán en los estados de Puebla y Oaxaca respectivamente, lo cual dada la cantidad de un total de 10,000 imágenes conforman el dataset para el entrenamiento de una red neuronal, dicha reserva está diversificada por una gran variedad de plantas silvestres de la región, este trabajo se enfoca en la familia de los cactus.
El objetivo de realizar este trabajo es obtener la estimación aproximada de la cantidad de plantas con la que cuenta la reserva, el cual ayudaría a contribuir con la información existente sobre estas plantas, por el cual la herramienta a usar son las redes neuronales convolucionadas que nos permite realizar el reconocimiento de dichas plantas dada sus características biológicas, textura, color entre otros, el cual permite su extracción y con ello facilitar el uso del aprendizaje profundo.

II. Trabajo relacionado

El trabajo que desarrolló Yann LeCun, Léon Bottou,. et al. ha sido el trabajo principal del cual sen han derivado diferentes métodos con redes neuronales para aprendizaje profundo, el algoritmo con el que funciona su trabajo es basado en gradiente aplicado a reconocimiento de caracteres escritas a mano en un documento.
[1] En el trabajo de Steve Lawrence et al, explican la

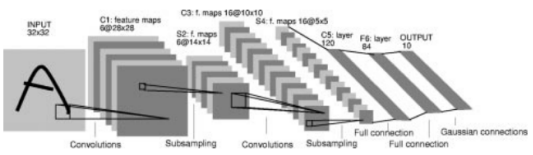

Figure 1. Arquitectura de una CNN LeNet-5

realización de reconocimiento facial con CNN. Las caras representan complejas características ya que el de desarrollo de modelos computacionales para reconocimiento facial es difícil. [2] Ellos proponen una solución con una red neuronal híbrida comparando favorablemente con otros métodos. Se pueden identificar al menos dos amplias categorías de sistemas de reconocimiento facial.

- Se quiere encontrar a una persona dentro de un extenso base de datos con rostros de personas. Esos sistemas generalmente retornan una base de datos con rostros mas parecidos.
- Se quiere identificar personas en tiempo real, permitir o denegar acceso a personas.

Por otro lado Christian Zsegedy et al, presentan una formulación para detección de objetos como un problema de regresión para encontrar las mascaras en un recuadro. [3]

III. Desarrollo

A. Descripción del Dataset

Para construir un dataset se requieren elementos importantes para conformarlo, por ejemplo:

- Un conjunto grande de datos
- Datos para entrenamiento, prueba y validación
- Etiquetado de cada una de las imágenes o datos
- Directorios identificados para cada una de las clases a utilizar.

El dataset está estructurado de la siguiente manera:
Se tienen dos clases: **Cactus** En donde en esta clase se tienen todas las imágenes que contiene los cactus como objeto a reconocer, el cual contiene un total de 5364 imágenes en donde cada imagen tiene como etiqueta **Cactus.1** a **Cactus.5364**. En la clase **No-cactus** se tienen 5364 imágenes en donde las imagenes contienen otros objetos que no contienen cactus, el cual tienen como etiqueta **SinPlanta.1** a **SinPlanta.5364**

Todas las imagenes son de 32x32 pixeles, formato ideal para Lenet-5.

Para realizar la respectiva clasificación se hizo el etiquetado para las dos clase, como se muestran en las siguientes imágenes:

En la siguiente figura se muestra el contenido de los directorios divididos en dos clases:

IV. Implementación del Dataset en Lenet-5

Una vez realizada la clasificación de las imágenes en sus respectivas clases, se implementó en una red convolucional propuesto por LeNet-5, dada la arquitectura y retomando

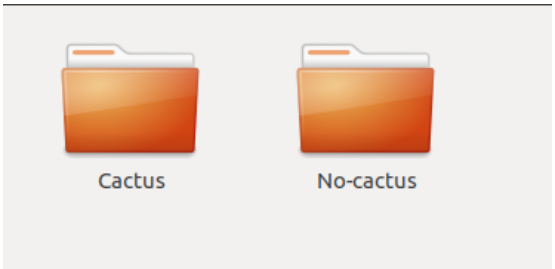

Figure 2. Directorio del dataset

Figure 3. Directorio del dataset

Figure 4. Directorio del dataset

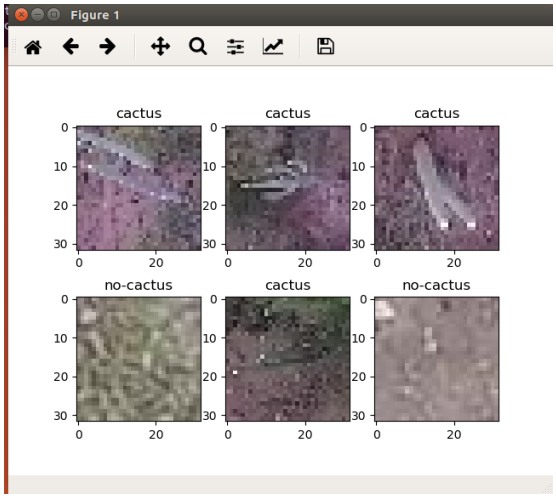

Figure 5. Clasificación de los datos

proyectos implementados, se usó Keras sobre tensorflow. En el siguiente código se muestra la forma de cargar las imagenes en Keras. https://github.com/anujshah1003/own_data_cnn_implementation_keras, se hicieron modificaciones quedando de la siguiente manera: https://github.com/jmzelectronica/Cactus_NN/blob/master/own_dataset_2.py

```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout,
    Activation, Flatten
from keras.layers.convolutional import Convolution2D
    , MaxPooling2D
from keras.optimizers import SGD,RMSprop,adam
from keras.utils import np_utils

import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import os
import theano
from PIL import Image
from numpy import *
# SKLEARN
from sklearn.utils import shuffle
from sklearn.cross_validation import
    train_test_split

# input image dimensions
img_rows, img_cols = 200, 200

# number of channels
img_channels = 1

#%%
#  data

path1 = 'C:\Users\Ripul\Documents\Python Scripts\
    keeras-cnn-tutorial\input_data'    #path of
    folder of images
path2 = 'C:\Users\Ripul\Documents\Python Scripts\
    keeras-cnn-tutorial\input_data_resized'  #path
    of folder to save images
```

En en siguiente código se muestra la arquitectura que se usa:

```
# convert class vectors to binary class matrices
Y_train = np_utils.to_categorical(y_train,
    nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)

i = 100
plt.imshow(X_train[i, 0], interpolation='nearest')
print("label : ", Y_train[i,:])

#%%

model = Sequential()

model.add(Convolution2D(nb_filters, nb_conv, nb_conv
    ,
border_mode='valid',
input_shape=(1, img_rows, img_cols)))
convout1 = Activation('relu')
model.add(convout1)
model.add(Convolution2D(nb_filters, nb_conv, nb_conv
    ))
convout2 = Activation('relu')
model.add(convout2)
model.add(MaxPooling2D(pool_size=(nb_pool, nb_pool))
    )
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(128))
```

```
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adadelta')
```

V. Resultados

El resultado que se obtuvo para este trabajo, fue de manera exitosa el cual se hizo el entrenamiento con 10 epocas, con 8000 datos de entrenamiento.

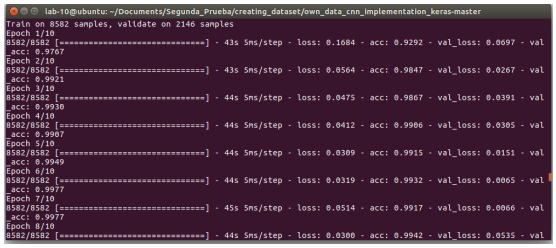

Figure 6. Proceso de entrenamiento

VI. Conclusiones

La realización el dataset para una red neuronal resulta ser una etapa importante para obtener un buen resultado en la salida de una red neuronal, para el caso de las redes neuronales convolucionadas se requieren un pre-procesamiento de dichas imágenes dado que si existe algún error es muy probable que el resultado no sea correcto.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[3] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in neural information processing systems*, 2013, pp. 2553–2561.