

Clasificador de nubes

A continuación haremos un modelo predictivo de imágenes, en concreto, para reconocer cualquier tipo de nube que fotografiemos en el cielo.

```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
import seaborn as sns

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import TensorBoard
import datetime

datadir = '/Users/alexandremartinez/Desktop/Proyectos_py/Nubes'
categorias = ['Ac', 'As', 'Cb',
              'Cc', 'Ci', 'Cs', 'Ct',
              'Cu', 'Ns', 'Sc', 'St' ]

IMG_SIZE = 55
```

Buscamos en nuestro escritorio la carpeta donde se encuentren las fotos ordenadas por tipos y las exportamos y re-dimensionamos.

```
training_data = []

def create_training_data():
    for category in categorias:
        path = os.path.join(datadir, category) #Nos metemos en la carpeta correspondiente dentro de carpeta
        class_num = categorias.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,img))
                new_array = cv2.resize(img_array,(IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])
            except Exception as e:
                pass

create_training_data()
```

Tras pasar las imágenes a un formato numérico, las normalizamos y las dividimos en un set de entrenamiento y otro de testeo.

```
import random
random.shuffle(training_data) #randomiza

X = []
```

```

y = []

for features, label in training_data:
    X.append(features)
    y.append(label)

X = np.array(X)
y = np.array(y)
X = X/255
#X = X.reshape(-1, IMG_SIZE, IMG_SIZE, 1)

import sklearn
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=5, test_size=0.1)

#X = X.astype('float32') # dtype of x changed back to numpy float32
#y = y.astype('float32') # dtype of x changed back to numpy float32

```

Realizamos un modelo neuronal convolucional

```

#Modelo

model = Sequential()
#Primera capa
model.add(Conv2D(64, (3,3), 1, activation = 'relu', input_shape = (IMG_SIZE, IMG_SIZE, 3)))
model.add(MaxPooling2D())

#Segunda capa
model.add(Conv2D(128, (3,3), 1, activation = 'relu'))
model.add(MaxPooling2D())

#Tercera capa
model.add(Conv2D(64, (3,3), 1, activation = 'relu'))
model.add(MaxPooling2D())

#Tercera capa
model.add(Flatten())

#Última capa con 10 neruonas de salida
model.add(Dense(100, activation = 'relu'))
model.add(Dense(len(categorias), activation = 'softmax'))

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

```

Evaluamos el modelo

```
# evaluate the model

cvscores = []

scores = model.evaluate(X_test, y_test, verbose=0)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))

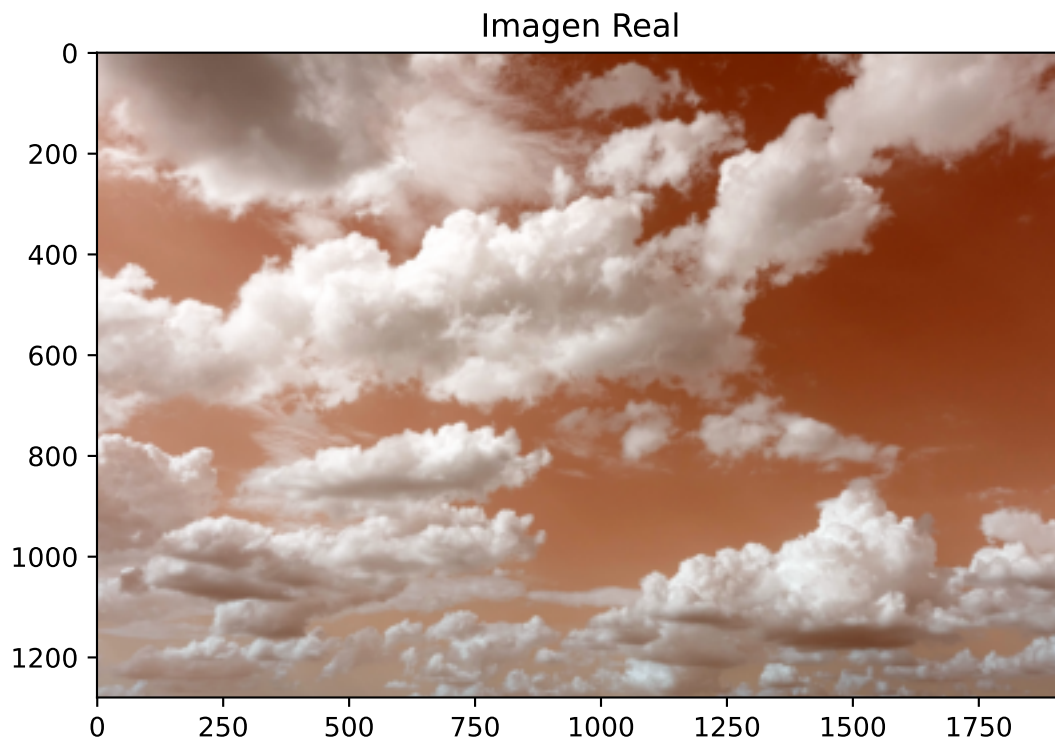
## accuracy: 29.02%
```

Lo ponemos a prueba con una foto real

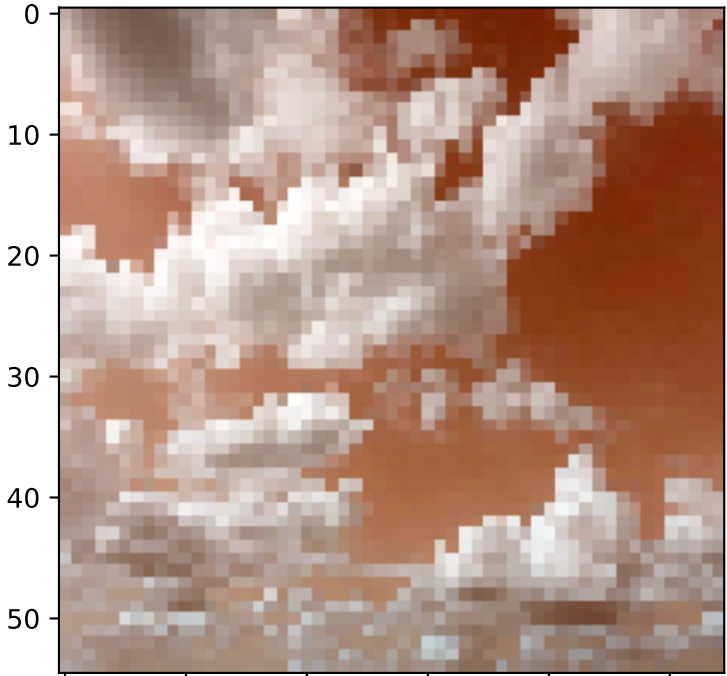
```
categorias = ['Ac', 'As', 'Cb',
              'Cc', 'Ci', 'Cs', 'Ct',
              'Cu', 'Ns', 'Sc', 'St' ]

img_array = cv2.imread('/Users/alexandremartinez/Downloads/cumulus_tipos-de-nubes.jpg')

plt.imshow(img_array)
plt.title('Imagen Real')
```



```


#img_array = img_array.astype('float32')
new_array = cv2.resize(img_array,(IMG_SIZE,IMG_SIZE) )

plt.imshow(new_array)
plt.show()
#new_array = new_array.astype('float32')

```

```

new_array = new_array.reshape(-1,IMG_SIZE,IMG_SIZE,3)
new_array = new_array/255

print('La foto presenta una nube de tipo:',categorias[y_classes[0]])

## La foto presenta una nube de tipo: St

```