SARA EL BAKKALI EL BAKKALI

ILAND KATIANA CHOO YALE

DETECTOR DE GATOS

1. Funciones programadas
   create_label → Parámetro de entrada: nombre de la imagen
   Salida: Array dependiendo de si el nombre de la imagen es "cat" o no.
   Este método se encarga de quedarse con el nombre de cada imagen para poder clasificar las imágenes como gato o no gato.

   create_train_data→ Parámetro de entrada: Ninguno
   Salida:Array con los datos de entrenamiento.
   Este método se sirve del directorio de imágenes de entrenamiento (TRAIN_DIR). Crea un array con los datos de cada imagen que se va a usar para entrenar nuestra red neuronal. Guarda también, este array en train_data.npy.
   create_test_data→ Parámetro de entrada: Ninguno
   Salida:Array con los datos de *testeo*.
   Este método se sirve del directorio de imágenes de entrenamiento (TEST_DIR). Crea un array con los datos de cada imagen que se va a usar para *testear* nuestra red neuronal. Guarda también, este array en test_data.npy.

2. Programa paso a paso
   Lo primero se descargan todos los módulos necesarios para poder usar las funciones que necesitamos para poder programar. Para esto se usa el comando en Anaconda Prompt "pip install *package*".

   Tuvimos un problema con *tensorflow* por lo que se creó un *enviroment* en Anaconda con Python = 3.6.4
   Una vez tenemos nuestro entorno con todas las dependencias solucionadas. Pasamos a codificar de la siguiente manera:

   Se configura los directorios y las variables de entorno. Las variables de entorno son IMG_SIZE(tamaño de la imagen),LR y MODEL_NAME(nombre del modelo que se va a utilizar para pasar los datos de los gatos).
   Los directorios son TRAIN_DIR que es el directorio de entrenamiento del modelo de los gatos y TEST_DIR que es el directorio de testeo y pruebas de los modelos de los gatos.
   Estas variables se configuran a nivel global para poder utilizarlas en cualquier momento de la aplicación,ya que son variables de entorno básicas con las que se rige la aplicación en todo su contexto. Estas variables deben ser accesibles en cualquier momento del código principal para poder ser invocadas por la aplicación.

Se define la primera función créate _label que toma como argumento imagen_name que es un argumento que es el nombre de la imagen que se pasa a la función create_label para generar un vector codificado de esa imagen con ese nombre. Con esta función logramos convertir la imagen en una función matricial o vectorial que define al gato cuyos valores corresponden con las variables de entorno booleanas 1,0 o 0,1 dependiendo de lo que caracterice a la imagen en una división de la imagen para poder determinar los valores matriciales de la matriz de la imagen del gato.

Una división de la imagen, una división de la etiqueta que sale de la imagen y por último un if que dependiendo del valor que tome la etiqueta dará un valor de array. Si es igual la etiqueta a cat se devuelve [1,0] y si no se devuelve [0,1]. Con esto se construye el array codificado que sirve como etiqueta para trasladar la imagen a un vector final.
La segunda función se llama create_train_data que no tiene argumentos. Esta función se utiliza para crear los datos de entrenamiento, los valores que se utilizaran para comprobar si el gato coincide con el modelo de las imágenes de los gatos.
Esta función primero crea un array vacio para ir introduciendo los datos de entrenamiento que se recogen del directorio de entrenamiento.
A continuación se hace un recorrido en una variable con un for del directorio de entrenamiento se coge la imagen y se pasa mediante la librería cv2 y se redimensiona para prepararla al tamaño que tenemos por defecto en la variable de entormo.Esta acción nos permite recorrer con un for cada una de las imágenes de entrenamiento del directorio de entrenamiento y así se carga cada imagen de ensayo de este directorio de entrenamiento.
A continuación en la función se añade al array vacio el array de la imagen de entrenamiento y el modelo de la imagen de datos para conseguir tener en ese array vacio las imágenes de entrenamiento con su par del gato del modelo.Así sabremos si la imagen de entrenamiento se asemeja a un gato o no.
Después se utiliza la función random shuffle para crear los datos de entrenamiento se crea un peso aleatorio de movimiento de la imagen de ensayo.
A continuación se guarda en memoria la imagen de ensayo para mantenerla en memoria y finalmente se sale de la función devolviendo los datos de ensayo.
La tercera función es create_test_data sin argumentos,que se utiliza para generar los datos de la prueba o test.Empieza también con un array vacio que se inyecta a una variable testing_data que guarda los valores para los datos del test.
A su imagen y semejanza como la anterior función ,solo que esta vez se coge el directorio de test para crear los datos del testeo y guardarlos en la anterior variable.Se lee cada fichero imagen y se añade tras hacer el resize y la lectura de la imagen de cada una de las imágenes del directorio de test.
Después se utiliza un random shuffle de los datos de la imagen conseguida y se guarda en memoria para mantener en memoria estos datos.
Por último se retorna los datos del testeo ,el array creado anteriormente y que ha ido añadiendo cada uno de estos datos.
El siguiente paso realizamos el programa principal,las dos primeras acciones que se hacen en el programa principal son generar los datos de entrenamiento y testeo con las funciones anteriores y se guardan en dos variables llamadas train_data y test_data.

Se generan otras dos variables train y test cargando estas variables con los 500 últimos elementos de train_data en train y los 500 primeros elementos de train_data en test para generar las variables que permitan organizar el train y el test.
A continuación se generan los lados x e y de train y test para obtener las filas y las columnas de estas variables,para ello se utiliza una redimensión al tamaño de la imagen variable global con un for y así se puede obtener cada una de estas variables.
En la siguiente instrucción se llama a rest_default_graph que llama a la librería tensorflow.
Con esta instrucción se resetea al gráfico por defecto,porque hay que comenzar con un grafico en cuestión inicial.
La siguiente instrucción crea una variable en la que se introduce la input_data para redimesionar estos datos de entrada en las variables convnet .
Con esta variable se aplica la función conv_2d de la librería tflearn para con una activación basada en el valor relu se genera el grafico 2d basada en los datos de entrada anteriores.
La siguiente función que se utiliza para la imagen que se esta generando en convnet es la función max_pool_2d de la librería tflearn para maximizar los datos de entrada.
Después se invoca la función fully_connected de la librería tflearn con la activación relu para garantizar todos los puntos conectados de los datos de entrada.
A continuación con la función regression de la librería tflearn se genera una regresión de estos datos de entrada y se categorizan los datos perdidos con un optimizador Adam.
El modelo se genera con la función DNN de tflearn y se pasan como argumentos la variable convnet ya trabajada previamentey unos valores de tablero de tensorboard que son el directorio en este caso log para el tratamiento de errores y verbosa para que de información.
Posteriormente el modelo se autoajusta con los valores de X_train,Y_train de valores de entrada y una validación cuyo input son los valores de x_test y y_test .Los valores de entrada son x o bien x_train o bien x_test y la salida las y o bien y_train o y_test.Caracterizando como valor del autoajuste el modelo con el nombre.
En este momento se utiliza para ejecutar y tener una salida el modelo con su nombre.
Con todos estos pasos hemos logrado saber si el gato es en los valores de prototipo de entrenamiento un gato en realidad u otra imagen,así podremos discernir si se trata de un prototipo si es gato o un prototipo que no coincide con ningún gato.
A continuación se relata un ejemplo de como utilizar el programa anterior.
Inicialmente se genera una variable que guarda los valores del testeo inicial,es decir el valor 0 de la matriz test_data.
Se guarda la imagen data y el número de imagen generada anteriormente ,los datos se redimensionan al tamaño de la imagen definida como variable global y se lanza una predicción sobre estos datos sobre el modelo.
Se genera una figura con los datos de la imagen y de entrada ,y se predice si es gato o no.Esta predicción se imprime con un prin sabiendo así si es un gato o noes un gato la imagen del prototipo de entrada .
Se genera un for para crear sobre los 16 primeros elementos de test_data una función pequeña que redimensiona cada una de estas enumaraciones de test_data y así poder saber redimensionando al tamaño de la imagen como variables global una predicción .
La predicción es la ya conocida de si el prototipo es gato o no ,para cada una de las ítems que son en total 16.

Este ejemplo de código final se podía hacer con más elementos, pero nosotras hemos cogido 16 en este caso.

3. Resultados obtenidos durante las experimentaciones

   Durante los primeros entrenamientos el acierto (*acc*) era muy bajo, entorno a 0,6 aproximadamente, por lo que se decidió aumentar el número de imágenes tanto de gatos como de otras cosas (coches, flores, personas, etc) para que el entrenamiento sea más óptimo. Además, se añadieron más capas convolucionales para mejorar el acierto (*accuracy*) y mejoró sustancialmente hasta un 0,88-0,9, estas capas se dotaron desde 32 filtros, aumentando progresivamente en las consiguientes capas hasta 256 filtros.
   Hemos usado *Adam* (https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/) como optimizador con un rango de aprendizaje de 0.001.
   Se ha entrenado la red neuronal con 10 epochs.

4. Forma de ejecutar el programa

   Para ejecutar el programa bastaría tener instaladas las dependencias necesarias y Python =3.6

```
(gatos2) C:\Users\iland>python --version
Python 3.6.8 :: Anaconda, Inc.
```

   Nosotras hemos lanzado la aplicación con el comando "Python *ruta_programa.py*"

```
(gatos2) C:\Users\iland>python C:\Users\iland\gatos-convolucional\gatos.py
curses is not supported on this machine (please install/reinstall curses for an opti
mal experience)
Scipy not supported!
100%|                                                | 7544/7544 [00:26<00:00, 288.49it/s]
100%|                                                | 16/16 [00:00<00:00, 116.28it/s]
```

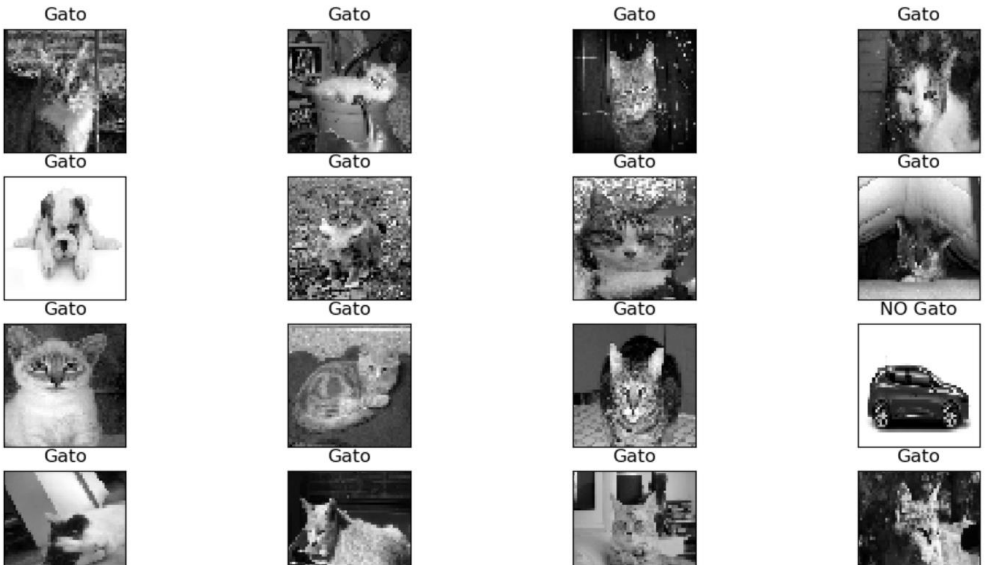
5. DataSet

   Consta de 7044 imágenes. Las imágenes de gato están nombradas como cat y las demás tienen nombres aleatorios que no nos importan en el aprendizaje ya que se evalúa si el nombre es *cat* u otro.

```
Run id: redConvolucional-gatos
Log directory: log/
---------------------------------
Training samples: 7044
```

6. Resultado final

   No hemos conseguido que aprenda al 100% pero como podemos ver en la siguiente imagen, el coche nos lo clasifica como NO Gato.




```
| Adam | epoch: 010 | loss: 0.31305 - acc: 0.8936 | val_loss: 0.30803 - val_acc: 0.8920 -- iter: 7044/7044
--
Gato: 0.858624279499054, noGato: 0.14137572050094604
```