MEMORIA DE PRÁCTICAS

Visión por Computador

Enric Bonet Cortés
Máster en IARFID - Universidad Politécnica de Valencia

Basic Implementations

En este primer ejercicio, se ha realizado la implementación de una red convolucional, conocida como VGG, sobre el dataset CIFAR10. Este dataset, se encuentra dentro de la librería de ***Keras*** en Python, y nos ofrece un corpus de imágenes de 10 tipos diferentes de clases, entre las que se encuentran perros, gatos o aviones.

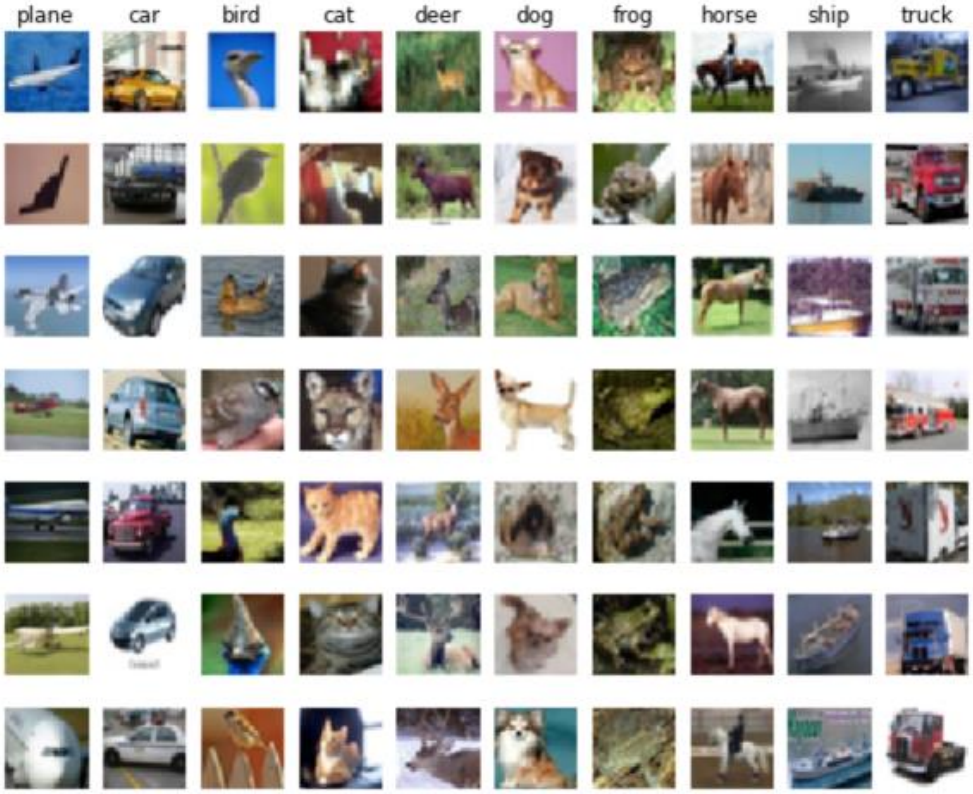

Figura 1: Dataset CIFAR10

En cuanto a la red convolucional que se ha comentado al principio, VGG, se trata de una conocida topología que se creó en el año 2014 en el Instituto de Robótica de Oxford. La VGG tiene diferentes nomenclaturas que derivan del número de capas que se decida utilizar en cada bloque, puesto que la topología está compuesta de 5 bloques de idéntico número de filtros convolucionales. Tras cada uno de estos bloques, la red añade una capa de Pooling que reduce el tamaño de la imagen que le entra a la mitad. Finalmente, tras el último bloque de convoluciones y pool, se encuentran 3 capas de lo que se denominan Fully Connected, que son las que se encargan de recoger la información obtenida de los bloques convolucionales para clasificar una imagen en una clase u otra.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figura 2 : Diferentes configuraciones de la VGG

Para realizar esta implementación en Python, primero se ha creado una función **def_vgg**, la cual toma como parámetros el número de capas convolucionales para cada bloque que se va a definir en el modelo. Con el fin de que se entienda mejor, si en los 5 primeros parámetros de **def_vgg** se pasan los valores **(1,1,2,2,2)**, la VGG resultante será una VGG-11, cuya configuración se corresponde con la configuración A de la Figura 2.

Más en detalle, cada una de estas capas de convoluciones, además de llevar la propia capa de convolución con tamaño de filtro de 3x3, aplican al modelo las técnicas Batch Normalization y Gaussian Noise (con factor 0.3) vista en clase de teoría.

```
datagen = ImageDataGenerator(
    width_shift_range=0.3,
    height_shift_range=0.3,
    rotation_range=20,
    zoom_range=[1.0,1.2],
    horizontal_flip=True)
```

Figura 3 : Data Augmentation utilizado en la VGG

Por último, con el fin de mejorar la precisión de la red, se han probado diferentes tipos de Data Augmentation, de los cuales el que mejor resultado ha dado es el que se muestra en la Figura 3.

Junto a él, la configuración que ha obtenido mayor rendimiento para dicho corpus ha sido la **D**, aunque en el propio Notebook de Google Collab que se adjunta llamado “VPC-CIFAR10 con VGG”, se encuentran comentadas el resto de configuraciones con el fin de probarse si es necesario.

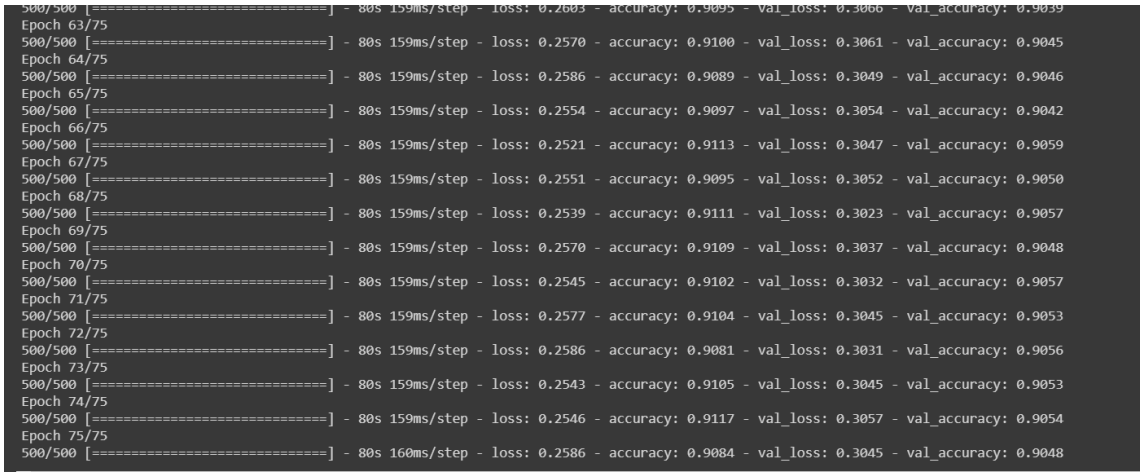

Figura 4 : Resultado final de la VGG-16

A = 0.8698, parámetros = 28,155,018

B = 0.8944, parámetros = 28,340,298

C = 0.8978, parámetros = 28,936,522

D = 0.9048, parámetros = 33,655,114

E = 0.9008, parámetros = 38,969,930