MELLERIO Antoine
GODOY Miguel Alexandro

12/2020

REDES DE NEURONAS

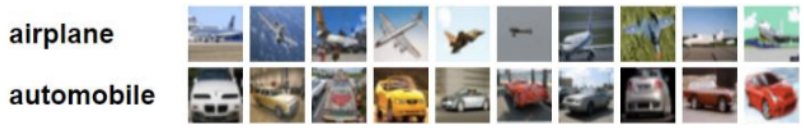
Practica 2 - Parte II

Clasificación de imágenes

SUMMARY

1. Introducción

El objetivo de esta parte es en encontrar a el mejor rede de neuronas para hacer una identificación del objeto en una photographia. Utilizamos el conjunto de datos CIFAR10 que se compone de 60 000 imágenes de color repartidas en 10 clases. Aquí esta un extracto de esta base :



Las fotografías ya están repartidos entre entrenamiento y test.

Las imágenes están compasados de 32x32 pixels y 3 canales (RGB). Para tratar los datos, aplanamos la estructura marcial en un vector gracias a la función *tensorflow.keras.Layers.Flatten*

Vamos a comparar dos typos de redes de neuronas : el perceptron multi-capa (PM) y el red de neurona convolucional (CNN), cada vez optimizando algunos de sus parametros.

2. Perceptron Multi-capa (PM)

Vamos a empezar en optimizar un perceptron multi-capa con 3 etapas :

A. En primer vamos a presentar nuestra método para elegir el bueno numero de ciclos de entrenamiento.

B. Después, vamos a probar diferentes conjuntos de parámetros, comparando sus errores de validación y sus exactitudes.

C. Finalmente, vamos a elegir la mejor configuración.

A. Numero de ciclos

Para entrenar nuestro red, tenemos que elegir un numero de ciclos que permite de entrenar suficiente sin sobre-entrenamiento. Para hacerlo, creamos un 'early stopping' callback :

```
from tensorflow.keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(
    min_delta=0.001, # minimium amount of change to count as an improvement
    patience=10, # how many epochs to wait before stopping
    restore_best_weights=True,
)
```

El entrenamiento del perceptron va a terminarse cuando no hay una diminución de la mejor perdida de validación de mas de 0,001 por mas de 10 ciclos siguiente. El modelo mantiene los pesos del fin del mejor ciclo.

Establecemos el parámetro del numero de ciclos de entrenamiento a 100 para estar seguro de no tomar demasiado tiempo. Debería ser suficiente para llegar a numero de ciclos optimo.

B. Comparación de conjuntos de parámetros

Comparamos diferentes configuraciones, modificando dos parámetros :

- el numero de capas ocultas
- el numero de neuronas en cada de estas capas.

Identificamos los redes por sus dos parámetros (N_capas_ocultas, N_neuronas). La primera configuración probada es (1,50) :

	Numero de capas ocultas	Numero de neuronas por capa oculta	Numero de ciclos	Errore de entrenamiento	Exactitud de entrenamiento	Errore de validación	Exactitud de validación
1	1	50	15	1,381	0,527	1,494	0,417
2	1	400	11	1,269	0,591	1,418	0,491
3	1	1000	8	1,319	0,577	1,461	0,484
4	2	400	14	1,479	0,608	1,360	0,530
5	3	400	14	1,477	0,609	1,368	0,534
6	5	400	18	1,233	0,697	1,378	0,519

Empezamos cambiando el numero de neuronas de la única capa oculta : {50, 400, 1000}. 400 neuronas es la major configuración con un error de validación de 1,4182.

Entonces seguimos con 400 neuronas en cada capa oculta, y aumentamos el numero de capas ocultas : 2, 3 y finalmente 5.

—> 3 capas ocultas de 400 neuronas parece ser el mejor numero.

Probamos añadir dropouts de probabilidad 0,3 entre cada capa oculta :

	Numero de ciclos	Errore de entrenamiento	Exactitud de entrenamiento	Errore de validación	Exactitud de validación
sin dropout	14	1,477	0,609	1,368	0,534
con dropout	79	1,393	0,651	1,240	0,572

El dropout mejora la exactitud de validación de 0.04, pero toma mas ciclos y tiempo.

C. Selección final del PM

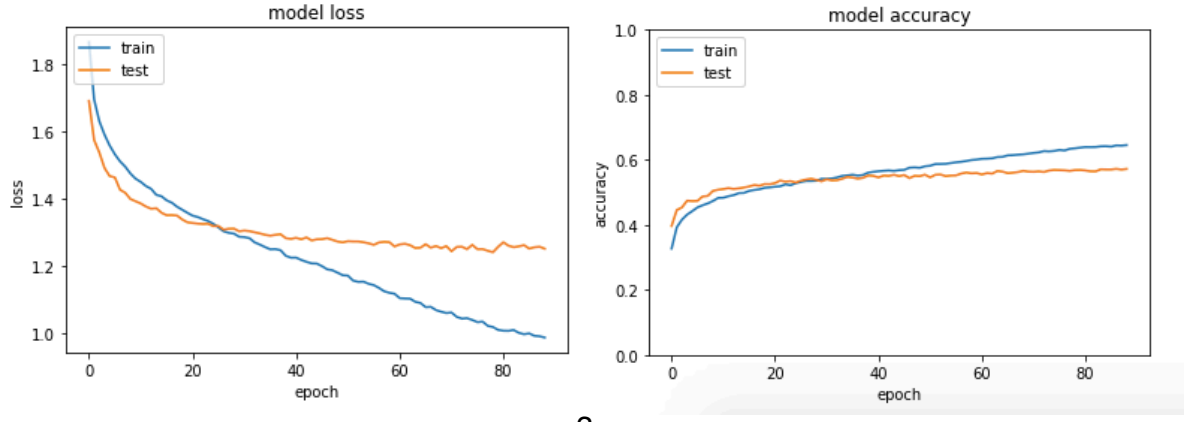
Para concluir, la mejor configuración del perceptron multi-capa es la siguiente :

- **numero de capas ocultas = 3**
- **numero de neuronas en cada capa oculta = 400**
- **dropout(0,3) entre cada capa oculta**

—> exactitud de 0,572

Aquí están las evoluciones del error de validación y de la exactitud del modelo optimizado :

```
Minimum validation loss: 1.2399516105651855
Maximum accuracy: 0.5723000168800354
```



3. Red de Neuronas de Convolución (CNN)

Ahora vamos a optimizar un CNN. Este estudio se realizará utilizando los siguientes parámetros :

- tamaño de los Kernel = 2x2 (pooling)
- optimizador = Adam
- función “loss” = sparse_categorical_crossentropy
- EarlyStopping para elegir el numero de ciclos

Comparamos diferentes configuraciones, modificando a parámetros : número de capas convolucionales, número de filtros, tamaño de los filtros, dropouts y optimizador.

A. Experimentos

- **NUMERO DE FILTROS EN LAS CAPAS DE CONVOLUCIÓN**

En primer, probamos con una única capa de convolución, cambiando el numero de filtros que todos son de tamaño (3,3):

	Numero de filtro en las capas convolucionales	Numero de ciclos	Errore de entrenamiento	Exactitud de entrenamiento	Errore de validación	Exactitud de validación
1	8	10	0,947	0,660	1,116	0,620
2	16	6	0,883	0,695	1,068	0,640
3	32	6	0,889	0,702	1,044	0,642
4	64	8	1,108	0,619	1,154	0,593

Aumentar el numero de filtros a 32 en cada capa optimiza la exactitud de algunos centésimos. Seguimos con esta valor.

- **TAMAÑO DE LOS FILTROS**

Ahora, utilizamos una única capa de convolución de 32 filtros, y cambiamos el tamaño de estos filtros :

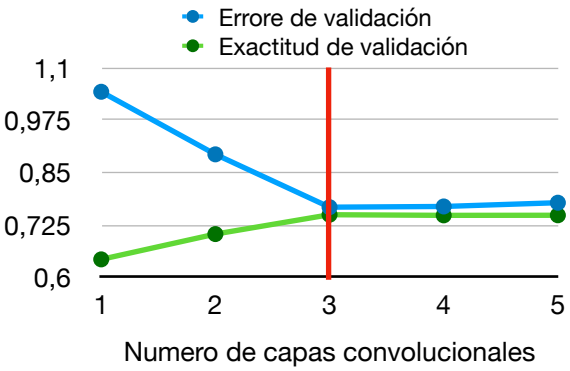
	Tamaño de kernel	Numero de ciclos	Errore de entrenamiento	Exactitud de entrenamiento	Errore de validación	Exactitud de validación
1	(2,2)	9	0,850	0,701	1,082	0,634
2	(3,3)	6	0,858	0,697	1,044	0,642
3	(6,6)	10	0,980	0,639	1,111	0,622

La filtros de tamaño (3,3) son los más eficientes, y también la configuración la mas rápida. Seguimos con ellos.

- **NUMÉRO DE CAPAS DE CONVOLUCIÓN**

Después, cambiamos el numero de capas de convolución de 32 filtros de tamaño (3,3) :

	Numero de capas convolucionales	Numero de ciclos	Errore de entrenamiento	Exactitud de entrenamiento	Errore de validación	Exactitud de validación
1	1	6	0,858	0,697	1,044	0,642
2	2	6	0,768	0,725	0,894	0,702
3	3	8	0,601	0,790	0,767	0,749
4	4	7	0,603	0,793	0,769	0,747
5	5	7	0,604	0,788	0,778	0,748



Si utilizamos cada vez el mismo typo de capa, 3 capas es la mejor configuración.

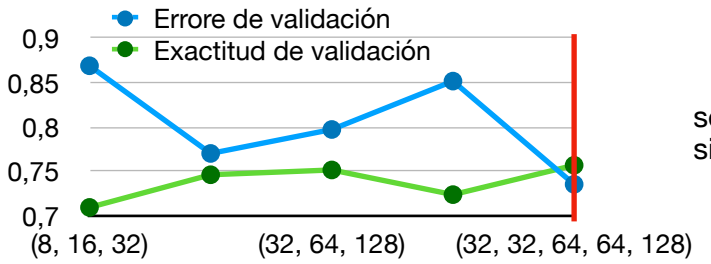
Aumentar este numero no diminuye la cualidad pero no la mejora. El tiempo de ejecución es el mismo también.

Es el mejor resultado que hemos encontrado hasta ahora.

- **COMBINACIÓN DE DIFERENTES CAPAS DE CONVOLUCIÓN**

Ahora, probamos a combinar capas de diferente números de filtro, cada una habiendo tamaños de filtro de (3,3) :

	Numero de filtro en las capas convolucionales	Numero de ciclos	Errore de entrenamiento	Exactitud de entrenamiento	Errore de validación	Exactitud de validación
1	**(8, 16, 32)**	7	0,799	0,725	0,869	0,709
2	**(16, 32, 64)**	6	0,613	0,788	0,770	0,746
3	**(32, 64, 128)**	5	0,524	0,820	0,797	0,752
4	**(8, 16, 32, 64)**	6	0,689	0,781	0,852	0,724
5	**(32, 32, 64, 64, 128)**	5	0,433	0,829	0,735	0,757



Entonces las dos configuraciones que se distinguen son la 3 y la 5 que tienen similares exactitudes (0,752 y 0,757)

- **DROPOUT**

Probamos ahora en añadir *dropout* de diferentes probabilidades después de cada typo de capa, por las dos configuraciones elegidas antes, cada vez con filtros de tamaño (3,3) :

	Dropouts	Numero de filtro en las capas convolucionales	Numero de ciclos	Errore de entrenamiento	Exactitud de entrenamiento	Errore de validación	Exactitud de validación
1	**sin**	**(32, 32, 64, 64, 128)**	5	0,433	0,829	0,735	0,757
2	**(0.3, 0.3, 0.3)**	**(32, 64, 128)**	53	0,495	0,826	0,625	0,819
3	**(0.3, 0.3, 0.3)**	**(32, 32, 64, 64, 128)**	94	0,518	0,829	0,528	0,828
4	**(0.2, 0.3, 0.4)**	**(32, 32, 64, 64, 128)**	64	0,439	0,841	0,531	0,821

Los dropouts de probabilidad 0,3 son eficientes : mejoran la exactitud de 0,0659 es decir de 8,7%. Habiendo 5 capas de convolución en vez de 3 diminuye un poco el tiempo total y aumenta la exactitud de de 0,03.

Por otro lado cambiar las probabilidades de los dropouts no mejoran nuestro red.

Notamos que el tiempo de ejecución aumentó mucho falta a los dropouts : de algunos diez minutos a mas de 1h45.

- **OPTIMIZADOR**

Hasta aquí, hemos utilizado el optimizado Adam. Probamos a los siguientes mintiendo los mejor parámetros de Adam :

- Adagrad (learning_rate=0.01)
- RMSprop (learning_rate=0.001, rho=0.9)

	Optmizador	Numero de ciclos	Errore de entrenamiento	Exactitud de entrenamiento	Errore de validación	Exactitud de validación
1	**Adam**	94	0,518	0,829	0,528	0,828
2	**RMSprop**	33	0,725	0,770	0,626	0,796
3	**Adagrad**	51	0,523	0,815	0,549	0,814

Entonces podemos decir que el optimizador Adam es probablemente el mejor para este problema.

B. Análisis globales

El numero de capas convolucionales, el numero de filtros en cada capa y el tamaño de estos filtros deben ser adaptados a nuestro conjunto de dato.

Aumentar el numero de capa de convolución mejora el aprendizaje hasta un punto donde es inútil aumentar mas porque solo aumenta el tiempo de cada ciclo.

Aumentar el numero de filtros puede mejorar el aprendizaje pero puede también provocar un sobre-aprendizaje si aumentamos demasiado.

El tamaño de los filtros debe ser adaptado al tamaño de la información la mas pequeña de nuestros imágenes. Si es demasiado pequeño la exactitud va a ser diminuyas por el ruido. Si es demasiado grande no traduise toda la información y entonces la exactitud diminuye.

Los dropout aumentan el numero de ciclos necesario entonces el tiempo global de ejecución. Sin embargo, permite al aprendizaje de generalizarse mucho mejor, y entonces permite de alcanzar a una exactitud de validación más grande.

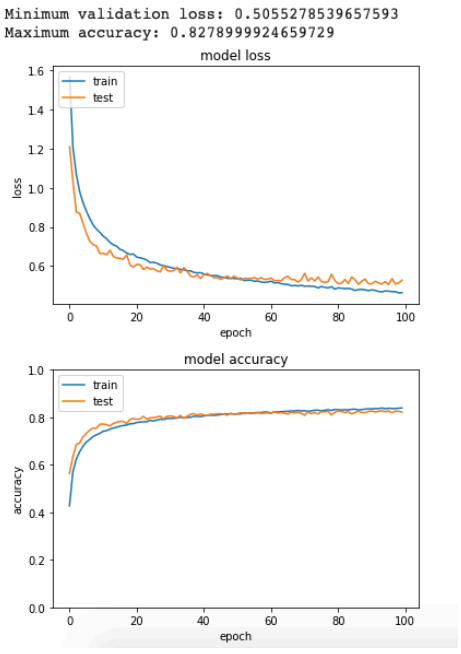
Los exactitudes y errores de validación varian mucho más con dropout. Se explica porque el red no se entrena con las mismas conexiones cada ciclo, y entonces las entradas de las neuronas cambian, y entonces las salidas del test también.

C. Selección final del CNN

La mejor configuración de CNN que hemos encontrado es :

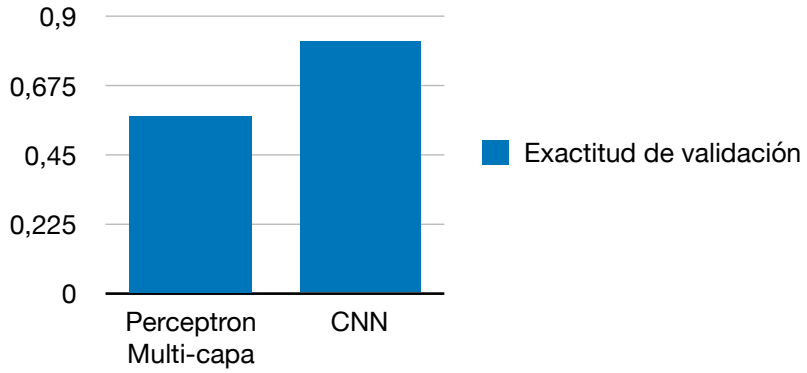
- tamaño de los Kernel : 2x2 (pooling)
- optimizador : Adam
- función “loss” : sparse_categorical_crossentropy
- capas de convolución : 5 de diferentes numero de filtros (32, 32, 64, 64, 128)
- dropouts : 3 de probabilidad 0,3 (después de cada typo de capa de convolución)
- tamaño de kernel de filtro : (3,3)

—> Exactitud de validación = 0,828



4. Comparación final

Para concluir, hemos optimizamos un perceptron multi-capa y un red de neuronas de convolución y hemos obtenido las exactitudes de validación de cada modelos :



Con el primer, alcanzamos a una exactitud de 0,572. Utilizar capas de convolución nos permitió de aumentar la exactitud de 44,8% para alcanzar a 0,828.

En los dos casos, añadir a dropout aumentó nuestro resultados de manera significativa.

Aquí esta la matriz de confusión para el conjunto de test para la mejor experimentación :

```
[[766  11  36  27  35   2  13  12  70  28]
 [  6 918   1   4   2   2   7   0  21  39]
 [ 34   0 703  40  77  40  72  23   8   3]
 [  4   1  44 658  57 131  56  34  10   5]
 [  2   1  23  32 848  13  41  30  10   0]
 [  6   2  21 121  54 738  19  36   2   1]
 [  3   2  16  39  16  12 908   2   2   0]
 [  8   0   9  28  51  21   6 868   4   5]
 [ 18   9   4  10   5   1   5   3 935  10]
 [ 14  51   3   9   3   1   7   6  30 876]]
```

Podemos ver que el error el mas común es cuando el classifier predice que un perro es un gato (131 vez) y el contrario (121 vez).