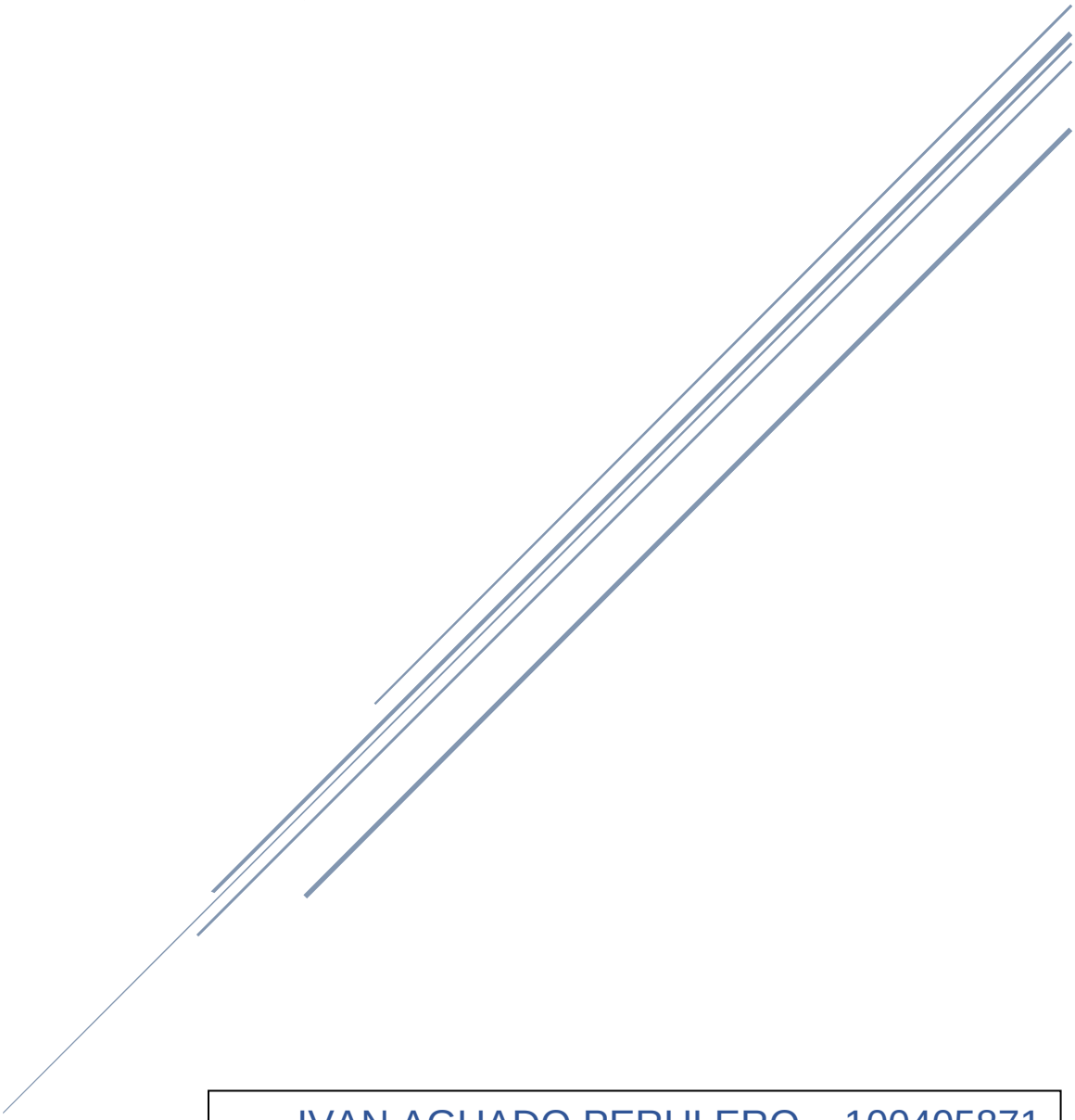
REDES DE NEURONAS ARTIFICIALES

PRACTICA 2-2

Universidad Carlos III Madrid, Ingeniería Informática, 2021



IVAN AGUADO PERULERO – 100405871
JORGE SERRANO PÉREZ – 100405987

Contenido

1.- INTRODUCCIÓN

El objetivo de esta segunda parte de la práctica es abordar un problema de clasificación utilizando el Perceptrón Multicapa (PM) y las redes de neuronas convolucionales (CNN). El problema en cuestión es el de clasificar distintas imágenes donde las entradas de la red serán directamente los píxeles de dichas imágenes. Para ello se nos ha proporcionado el conjunto de datos *CIFAR10* con 60000 imágenes en color de 32x32 pertenecientes a 10 clases diferentes, con 6000 imágenes por clase. Las clases posibles son "*airplane*", "*automobile*", "*bird*", "*cat*", "*deer*", "*dog*", "*frog*", "*horse*", "*ship*" y "*truck*". Además, se nos ha proporcionado un script en Python que deberemos modificar para experimentar correctamente.

El presente documento se divide en un capítulo para la experimentación realizada, los resultados obtenidos y un análisis de los resultados del PM, otro para lo mismo, pero de las CNN, y finalmente un capítulo con una comparación y las conclusiones obtenidas.

2.- PERCEPTRÓN MULTICAPA

La *loss* de test determina el número de ciclos óptimo, es decir, el ciclo en el que esta métrica que equivale al *sparse_categorical_crossentropy* es mínimo. Por lo tanto, los valores de los errores mostrados más abajo son del ciclo con la *loss* mínima.

En la columna ciclos de la siguiente tabla se detalla el número de ciclos con el que hemos experimentado y entre paréntesis el ciclo óptimo en el que recogemos los errores y del que sacamos el modelo, en caso de que dicho experimento sea el que posee mejores resultados. Este será aquel que posea el valor *loss* más bajo en el conjunto de validación.

En la columna del número de neuronas entre paréntesis tiene el número de capas ocultas del Perceptrón Multicapa. Además, todos los experimentos se han realizado con un factor de aprendizaje de 0.001.

Destacar que la primera columna indica el número de parámetros del modelo, es decir, el número de pesos.

Experimentación realizada

En esta primera tabla, se han colocado los resultados siguiendo el orden de experimentación que hemos seguido, es decir, primero modificando el número de neuronas y luego el número de capas.

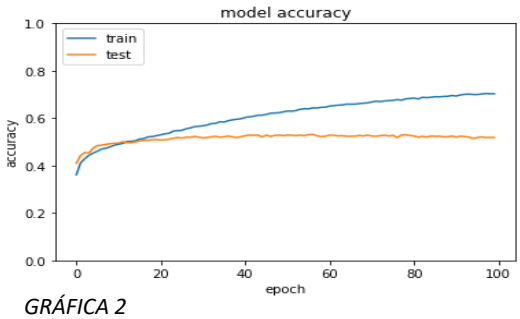
N. parámetros	N. neuronas (capas)	Ciclos	Accuracy (Entrenamiento)	Loss (Entrenamiento)	Accuracy (Test)	Loss (Test)
154172	50(1)	100(25)	0.494439	1.417504	0.476099	1.480001
770772	250(1)	100(33)	0.576960	1.198524	0.521099	1.359553
156722	50(2)	100(68)	0.493059	1.421409	0.475800	1.446004

833522	250(2)	100(52)	0.621079	1.052529	0.549199	1.277729
896272	250(3)	100(57)	0.605319	1.102982	0.547900	1.283971
833522 (RMSProp)	250(2)	100(43)	0.607739	1.134989	0.542200	1.319232

Tabla 1

Resultados obtenidos

Comenzamos la experimentación con una sola capa oculta, los números de neuronas probados fueron 50 y 250. Obtuvimos una *loss* de test de 1.48 y 1.35 y una *accuracy* de 0.47 y 0.52 respectivamente. Con esto pudimos empezar a comparar los resultados

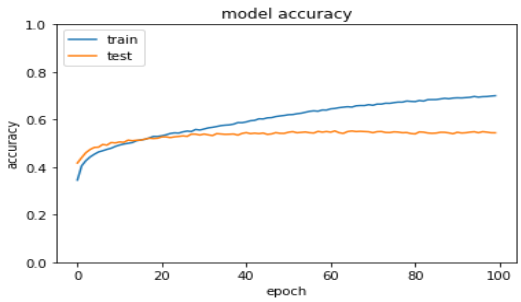


GRÁFICA 2

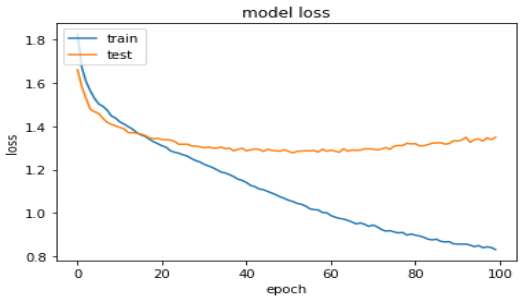
GRÁFICA 1

Tras esto nuestro siguiente paso fue realizar los mismos experimentos, pero con 2 capas ocultas. Aquí se mejoraron los resultados, pero de una manera muy sutil. Con una *loss* de 1.446 el de 50 neuronas seguía quedándose detrás del de 250 que tenía un valor de 1.277.

Aquí empezamos a observar que el Perceptrón Multicapa no era la mejor técnica para el problema al que nos enfrentábamos.



GRÁFICA 4



GRÁFICA 3

Nuestro siguiente paso fue pasar a las 3 capas, pues parecía que esto era beneficioso. Probamos con 250 neuronas que era el que mejor resultado nos había dado hasta ahora, sin embargo, no conseguimos mejorar el anterior. Tras esto, con el mejor, es decir, con el de 2 capas probamos a subir a 350 neuronas, pero esto no era beneficioso.

Como última prueba, ejecutamos con nuestra mejor arquitectura, pero con el optimizador *RMSprop*, pues era *Adam* el que veníamos utilizando. Sin embargo, observamos que este no mejoró los valores de las métricas del problema.

Análisis de los resultados del modelo final

El modelo final elegido es el de dos capas con 250 neuronas y el optimizador *Adam*. El número de ciclos óptimo se alcanzaba a los 52. Este cuenta con una *accuracy* y *loss* de test de 0.549 y 1.277 respectivamente. Resultados lejos de ser buenos, que se pueden corroborar con la matriz de confusión expuesta a continuación. La diagonal principal que son los aciertos cuenta con valores altos, sin embargo, fijándonos bien vemos que los fallos también son elevados. Como nos decía la *accuracy*, se clasifican las instancias un poco mejor que si fuera aleatoriamente.

```
[594  42  36  33  26  13  14  23 150  69]
[ 31 689   5  26   5   6   8  22  69 139]
[ 88  28 325 137 169  65  66  78  22  22]
[ 29  26  41 494  64 153  53  54  34  52]
[ 47  15  81  94 492  45  70 105  37  14]
[ 21   8  42 288  61 381  49  82  37  31]
[  8  26  46 134 151  52 523  18  22  20]
[ 35  14  20  86  63  71  15 607  25  64]
[ 85  63  10  25  24  11   2   7 722  51]
[ 28 192   6  49   7   8  11  32  74 593]
```

Matriz de confusión 1

3.- RED NEURONAL CONVOLUCIONAL

Experimentación realizada

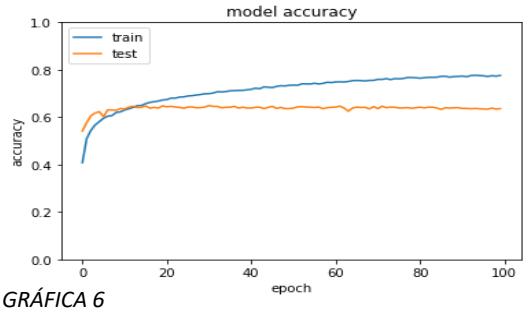
En esta siguiente tabla se muestran los resultados obtenidos en el orden de experimentación que hemos seguido. Primero hemos modificado el número de neuronas *fully connected* en una sola capa, al mismo tiempo que el número de filtros. El siguiente paso fue cambiar el tamaño o *kernel* de los filtros con las mismas combinaciones que antes. A continuación, pasamos a añadir más capas, primero de las convolucionales y luego de las *fully connected*. Por último, decidimos añadir muchas más capas convolucionales con filtros más pequeños.

N. parámetros	N. capas convolucionales	N. filtros (tamaño kernel)	N. neuronas "fully connected" (capas)	Ciclos	Accuracy (Entrenamiento)	Loss (Entrenamiento)	Accuracy (Test)	Loss (Test)
263318	1	16(3x3)	64(1)	100(16)	0.657100	0.948008	0.647099	1.015182
526166	1	16(3x3)	128(1)	100(16)	0.696399	0.837967	0.655200	1.002098
525910	1	32(3x3)	64(1)	50(34)	0.653559	0.932398	0.639599	1.048368
1050902	1	32(3x3)	128(1)	50(11)	0.689840	0.852437	0.661099	0.982870
526934	1	16(5x5)	128(1)	50(13)	0.710739	0.804784	0.667299	0.962950
1052438	1	32(5x5)	128(1)	50(12)	0.705519	0.809214	0.665199	0.977012
533350	2	16(5x5)	128(1)	50(7)	0.720979	0.783462	0.669300	0.956613
543446	1	16(5x5)	128(2)	50(14)	0.720000	0.798055	0.675800	0.951731
549862	2	16(5x5)	128(2)	50(12)	0.741119	0.736703	0.676500	0.929501
2193846	2	32(5x5)	128(2)	20(8)	0.797699	0.578062	0.688000	0.914510
2336726	3	32(7x7)	256(3)	50(14)	0.764259	0.695696	0.670199	0.975847
551686	12	16(3x3)	128(1)	20(19)	0.624939	1.045117	0.641600	1.025175
537766	6	16(3x3)	128(1)	20(19)	0.728820	0.749812	0.704999	0.859580
544726	9	16(3x3)	128(1)	30(27)	0.682720	0.883056	0.682799	0.909589
554278	6	16(3x3)	128(2)	40(37)	0.750800	0.707790	0.716099	0.818857

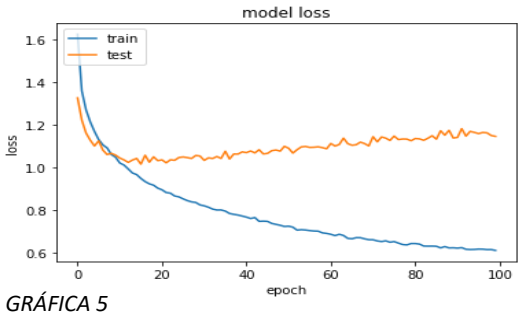
Tabla 2

Resultados obtenidos

Para comenzar con la experimentación en las redes convolucionales, inicialmente probamos con una sola capa convolucional de 16 filtros 3x3 y 64 neuronas en una sola capa *fully connected*. De esta manera obtuvimos unos resultados iniciales con los que poder comparar. Aquí ya nos dimos cuenta de que eran necesarios muchos menos ciclos que para el perceptrón multicapa para realizar correctamente el aprendizaje. A continuación, podemos ver las gráficas de la evolución de la *accuracy* y la *loss* de este primer experimento.

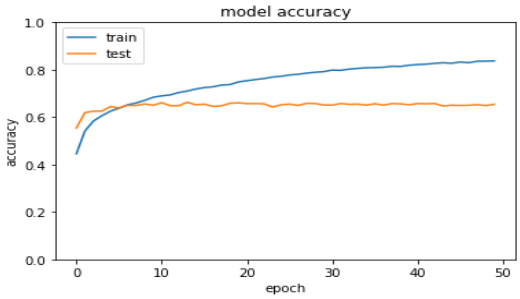


GRÁFICA 6

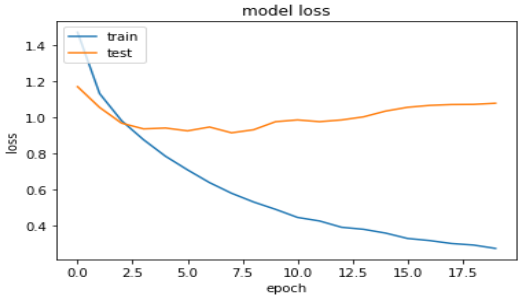


GRÁFICA 5

Continuamos con la experimentación aumentando primero el número de neuronas del PM y luego el número de filtros, y observamos que daba mejores resultados con mayor número de neuronas en vez de más filtros. Por ello, mostramos la siguiente gráfica del experimento con mejor *loss* hasta ese momento (0.98287), que era con 128 neuronas y 32 filtros.

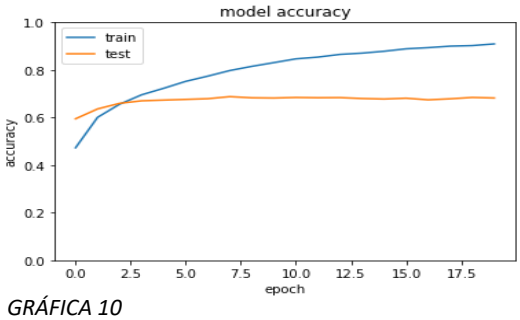


GRÁFICA 8

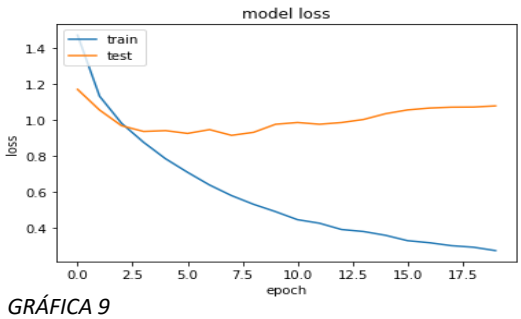


GRÁFICA 7

Visto lo mencionado anteriormente, pasamos a probar con filtros de mayor tamaño (5x5). Estos mejoraron nuestros resultados, quizá debido a que tiene en cuenta más pixeles de los alrededores. Como vimos que con filtros más grandes realizaba un mejor aprendizaje, el siguiente paso fue añadir más capas. De esta manera, con 2 capas convolucionales de 32 filtros 5x5 y 2 capas fully-connected de 128 neuronas, obtuvimos los mejores resultados que se muestran a continuación.



GRÁFICA 10

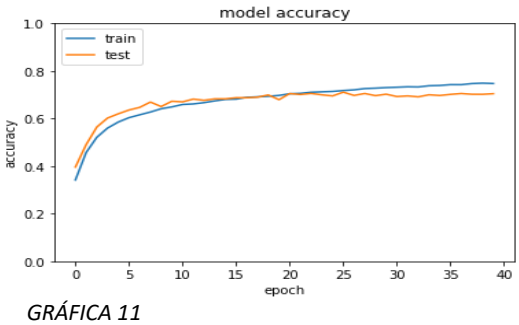


GRÁFICA 9

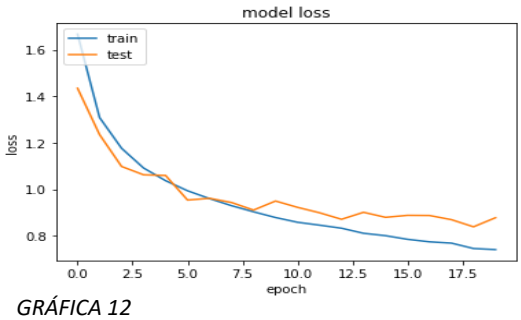
Como es lógico, la *accuracy* aumenta en el conjunto de entrenamiento, ya que según va entrenando va ajustándose mejor a los datos. Sin embargo, en el conjunto de test aumenta mucho al principio y luego se mantiene estable. Por otro lado, la *loss* en el entrenamiento sigue bajando, pero en el de test sube, lo que nos indica que se ha producido sobreaprendizaje.

Finalmente, siguiendo el razonamiento de antes, probamos un último caso con filtros 7x7 y además 3 capas convolucionales y 3 *fully connected*, pero no obtuvimos mejores resultados. Por lo tanto, visto que no encontrábamos un experimento que nos diera unos resultados especialmente buenos, cambiamos radicalmente de estrategia.

Pasamos a añadir muchas capas convolucionales de 16 filtros 3x3, ya que era donde habíamos obtenido los mejores valores, en concreto 12. Los resultados no mejoraron, así que bajamos el valor a la mitad. Y aquí obtuvimos un muy buen experimento con una *loss* de 0.85958 y una *accuracy* de 0.70499. Creemos que se debe a que debido a la gran cantidad de filtros que tenemos al añadir capas, estos captan mejor las características de la imagen original, es decir, el preprocesado de la imagen.



GRÁFICA 11



GRÁFICA 12

En las gráficas anteriores podemos ver la evolución de la *accuracy* y la *loss* para el experimento con 6 capas convolucionales de 16 filtros 3x3 y 128 neuronas en una sola capa *fully connected* del Perceptrón Multicapa.

Por último, pensamos que con dos capas en el PM en vez de una quizá daría mejor, ya que eso es lo que habíamos adquirido de experiencia al realizar los experimentos anteriores. Y estábamos en lo cierto, siendo este experimento final el mejor.

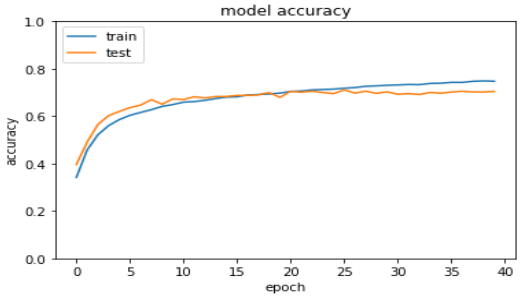
Análisis de los resultados del modelo final

El modelo final elegido es el de 6 capas con 16 filtros 3x3 en la parte convolucional. El número de ciclos óptimo se alcanzaba a los 37. Este cuenta con una *accuracy* y *loss* de test de 0.716 y 0.818 respectivamente. Resultados bastante aceptables, que se pueden corroborar con la matriz de confusión expuesta a continuación. La diagonal principal que son los aciertos cuenta con valores altos. Como nos decía la *accuracy* se clasifican la mayoría de las imágenes bien.

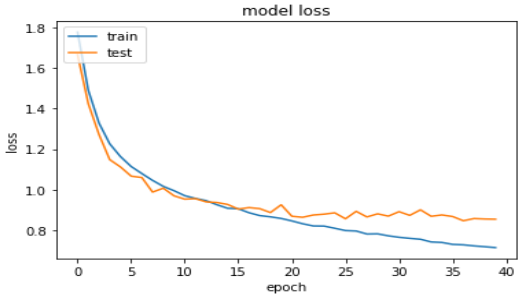
```
[696  17  37  31  13   4   5   9 155  33]
[ 15 816   6  23   1   4  10   5  54  66]
[ 91   3 581  87  86  60  40  30  17   5]
[ 23   7  62 597  61 128  49  41  20  12]
[ 25   2  85  99 650  21  45  61  10   2]
[ 10   3  63 263  36 536  16  58  13   2]
[ 10   7  75  75  41  16 764   5   7   0]
[ 18   0  24  78  54  48   3 766   8   1]
[ 41  20  19  25   3   3   6   4 869  10]
[ 31  75   8  39   4   3   0  22  47 771]
```

Matriz de confusión 2

A continuación, se puede observar la evolución de la *accuracy* y la *loss*.



GRÁFICA 14



GRÁFICA 13

Como trabajos futuros se podrían probar modelos con más número de capas, en los que ir alternando en cada capa el número de neuronas, filtros y tamaño de los mismos. Esto al investigar por internet era algo muy utilizado, sin embargo, probando nos dimos cuenta de que era un proceso muy lento, por lo que para realizar esta práctica lo descartamos.

4.- COMPARACIÓN Y CONCLUSIONES

Como se ha podido observar con las redes de neuronas convolucionales se obtiene un resultado mucho mejor que con el perceptrón multicapa. Mientras que con este último solo llegamos a clasificar con una precisión un poco superior a la aleatoriedad, con las CNN conseguimos clasificar bien en un 71% de las veces. Al final es lógico, pues en la parte de las CNN utilizamos una serie de filtros y el pooling antes de pasar el perceptrón multicapa. Este preproceso ayuda a que se consiga finalmente un mejor modelo que solo utilizando el PM. Estos resultados también se ven reflejados en la *loss* de test, que, al fin y al cabo, ha sido la métrica que utilizábamos.

Como conclusiones, consideramos que las CNN son una buena opción para clasificar imágenes, sin embargo, para conseguir un muy buen resultado, fuera de lo académico de esta práctica, se necesitaría bastante tiempo. No por dificultad, sino por lo que puede tardar en entrenar un buen modelo. Al final esto también depende de los recursos con los que cuente el usuario que vaya a hacerlo.

En definitiva, una práctica que ayuda a fijar los conocimientos teóricos sobre la materia, ayudando a ver cómo funciona y qué resultados se obtienen con una técnica como puede ser CNN para clasificar imágenes.