

Práctica Redes Neuronales

Redes Neuronales Artificiales

Enric Bonet Cortés

MÁSTER UNIVERSITARIO IARFID Universitat Politècnica de Valencia

1.- Perceptrón Multicapa: corpus MNIST

Antes de empezar con el primero de los ejercicios, aclarar que de las dos posibilidades que se nos ofrecían para implementar las redes neuronales, se ha escogido la librería Keras de Python.

El primero de los ejercicios consistía en implementar el algoritmo Perceptrón Multicapa (MLP) para la clasificación del corpus MNIST, que consiste en un conjunto de imágenes de tamaño 28x28, a escala de grises, de dígitos escritos a mano. Esto equivale a un problema de clasificación con 10 clases diferentes, pertenecientes a los números del 0 al 9.



Ilustración 1: Corpus MNIST

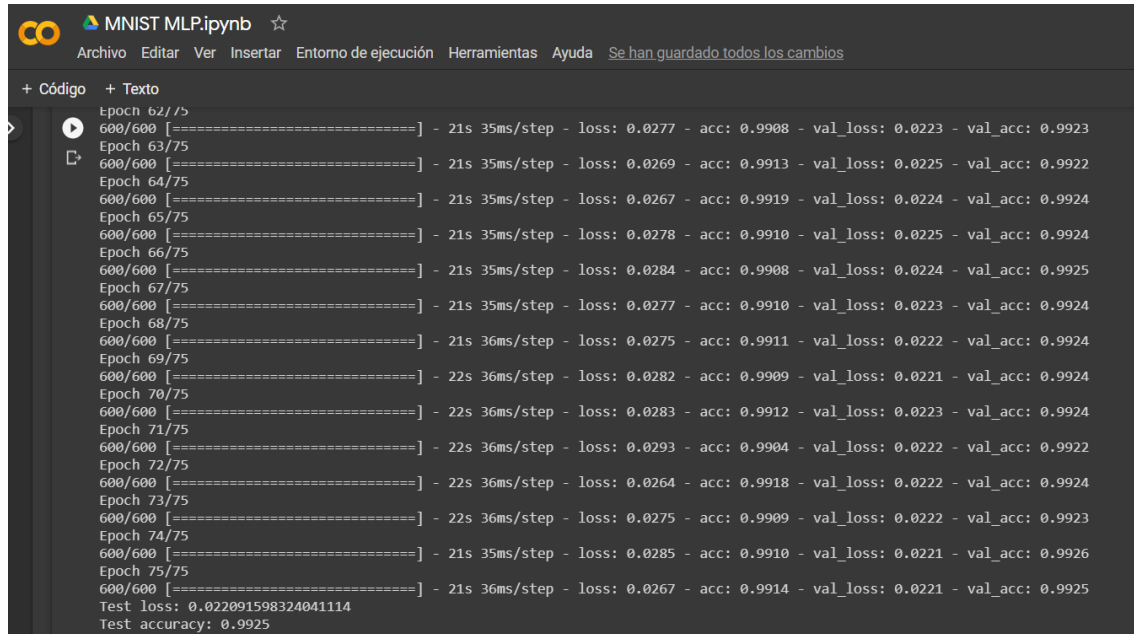
Para esta implementación, se ha partido de un Perceptrón Multicapa base con diferentes técnicas ya implementadas, vistas en clase, que ayudan a proporcionar un mejor porcentaje de acierto, así como previenen el "overfitting". Entre estas técnicas se encontraban:

- Data Augmentation, solo con rango de desplazamiento horizontal y vertical a 0.1
- Batch-norm
- Learning Rate Annealing
- Gaussian Noise, con el parámetro 0.3

Esta versión base, utilizaba como función de activación la función 'relu', cosa que se ha mantenido finalmente en el script final.

Para incrementar porcentaje de acierto, se ha ido modificando el algoritmo conforme se obtenían mejores resultados.

La versión final de esta red neuronal consiste en una versión similar del MLP base, el cual el Batch-norm se ha incorporado tras la función de activación. Además de esto, el Gaussian Noise se ha reducido de 0.3 a 0.1. Esta implementación del MLP ha obtenido mejores resultados, como se puede observar en la siguiente imagen:



```

+ Código + Texto
Epoch 62/75
600/600 [=====] - 21s 35ms/step - loss: 0.0277 - acc: 0.9908 - val_loss: 0.0223 - val_acc: 0.9923
Epoch 63/75
600/600 [=====] - 21s 35ms/step - loss: 0.0269 - acc: 0.9913 - val_loss: 0.0225 - val_acc: 0.9922
Epoch 64/75
600/600 [=====] - 21s 35ms/step - loss: 0.0267 - acc: 0.9919 - val_loss: 0.0224 - val_acc: 0.9924
Epoch 65/75
600/600 [=====] - 21s 35ms/step - loss: 0.0278 - acc: 0.9910 - val_loss: 0.0225 - val_acc: 0.9924
Epoch 66/75
600/600 [=====] - 21s 35ms/step - loss: 0.0284 - acc: 0.9908 - val_loss: 0.0224 - val_acc: 0.9925
Epoch 67/75
600/600 [=====] - 21s 35ms/step - loss: 0.0277 - acc: 0.9910 - val_loss: 0.0223 - val_acc: 0.9924
Epoch 68/75
600/600 [=====] - 21s 36ms/step - loss: 0.0275 - acc: 0.9911 - val_loss: 0.0222 - val_acc: 0.9924
Epoch 69/75
600/600 [=====] - 22s 36ms/step - loss: 0.0282 - acc: 0.9909 - val_loss: 0.0221 - val_acc: 0.9924
Epoch 70/75
600/600 [=====] - 22s 36ms/step - loss: 0.0283 - acc: 0.9912 - val_loss: 0.0223 - val_acc: 0.9924
Epoch 71/75
600/600 [=====] - 22s 36ms/step - loss: 0.0293 - acc: 0.9904 - val_loss: 0.0222 - val_acc: 0.9922
Epoch 72/75
600/600 [=====] - 22s 36ms/step - loss: 0.0264 - acc: 0.9918 - val_loss: 0.0222 - val_acc: 0.9924
Epoch 73/75
600/600 [=====] - 22s 36ms/step - loss: 0.0275 - acc: 0.9909 - val_loss: 0.0222 - val_acc: 0.9923
Epoch 74/75
600/600 [=====] - 21s 35ms/step - loss: 0.0285 - acc: 0.9910 - val_loss: 0.0221 - val_acc: 0.9926
Epoch 75/75
600/600 [=====] - 21s 36ms/step - loss: 0.0267 - acc: 0.9914 - val_loss: 0.0221 - val_acc: 0.9925
Test loss: 0.022091598324041114
Test accuracy: 0.9925

```

Ilustración 2: Resultado obtenido del MLP

Destacar también, que aparte de estas modificaciones, se probó a implementar una versión más compleja del Learning Rate Annealing, en la que no solo se decrementara el factor de aprendizaje, si no que este factor se reiniciase a la mitad del algoritmo, algo que también se ha dado en clase. No obstante, esta modificación del Learning Rate Annealing, no ofrecía mejores resultados, por lo que se descartó, dejándose finalmente su versión base.

2.- Redes Convolucionales: corpus CIFAR

En este caso, se ha implementado en Keras otro tipo de redes que hemos visto en teoría: una red Convolucional.

El corpus que se va a utilizar en este caso, CIFAR, consiste en un conjunto de imágenes pertenecientes a 10 clases diferentes, entre las cuales había algunas como perros o aviones.

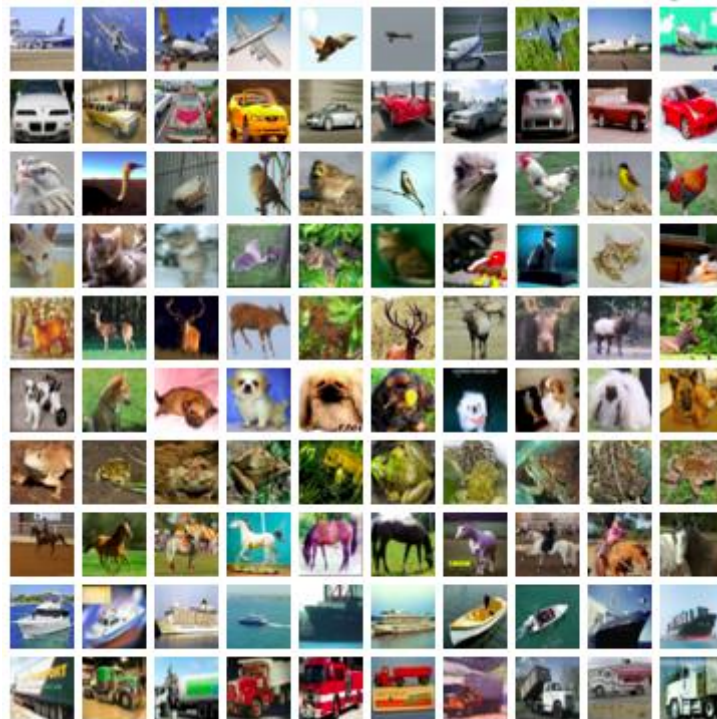


Ilustración 3: Corpus CIFAR

Al igual que para el algoritmo anterior, se ha partido de un algoritmo base el cual se ha ido modificando. Esta primera red Convolucional, cuenta con 5 capas de filtros convolucionales de tamaño 3x3, cuya profundidad, es decir, número de filtros, va desde 32 duplicándose hasta llegar a 512. Cada una de estas capas, cuenta, no solo con la propia capa de convolución y una función de activación 'relu', sino también con Batch-Norm, Gaussian Noise y MaxPooling, con una ventana de pool de 2x2.

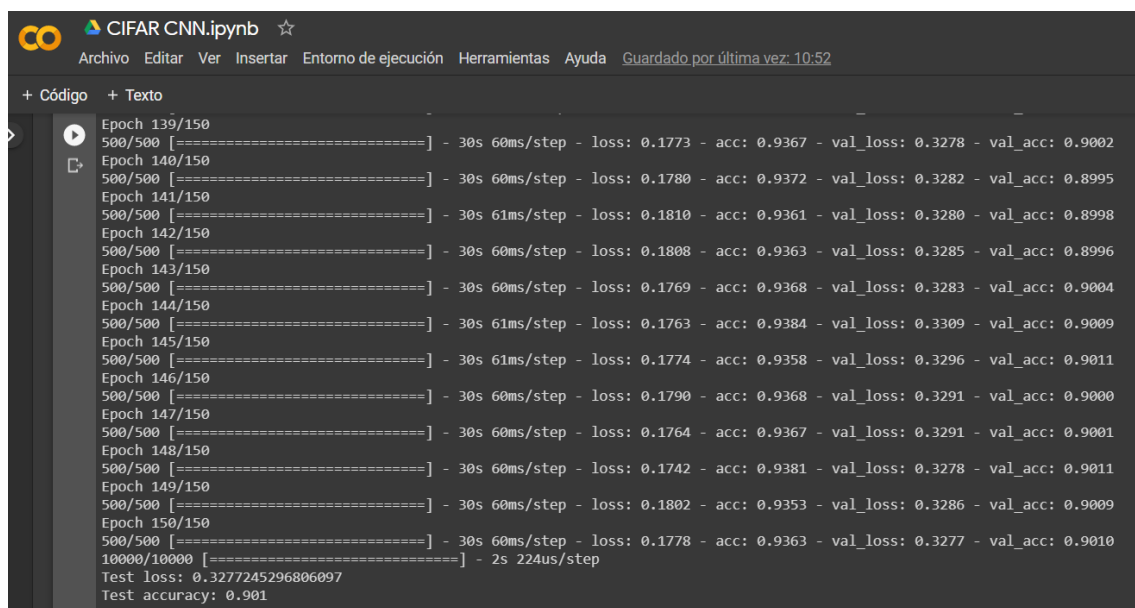
Respecto al Batch-Norm y al Gaussian Noise, se han realizado las mismas modificaciones que con el Perceptrón Multicapa que se aplicaba sobre MNIST. Además, se ha optado por utilizar el mismo tipo de Learning Rate Annealing que en el MLP de MNIST.

Posteriormente, se ha implementado también la técnica de Data Augmentation. En este caso, se han utilizado los mismos parámetros que en MNIST, y se han añadido, además,

un zoom y rotación a las imágenes del 20%. Tras varias pruebas, el desplazamiento tanto lateral como vertical, se ha fijado en 0.3, donde se obtenían mejores resultados.

Con esta versión, y un aumento de las iteraciones a 150, el algoritmo se aproximaba al 90% de acierto, aunque no lo llegaba a alcanzarlo del todo, por lo que se realizó un último experimento, añadiéndole por cada bloque de convoluciones, dos capas más de convolución, con la misma profundidad y tamaño, así como las mismas funciones de activaciones y diversas técnicas que se ha comentado. Gracias a esto, el algoritmo ha sido capaz de superar el 90% de acierto, sacrificando un aumento del número de parámetros a aprender considerablemente.

En la siguiente imagen se puede observar una ejecución donde se supera el 90% de acierto:



```

Epoch 139/150
500/500 [=====] - 30s 60ms/step - loss: 0.1773 - acc: 0.9367 - val_loss: 0.3278 - val_acc: 0.9002
Epoch 140/150
500/500 [=====] - 30s 60ms/step - loss: 0.1780 - acc: 0.9372 - val_loss: 0.3282 - val_acc: 0.8995
Epoch 141/150
500/500 [=====] - 30s 61ms/step - loss: 0.1810 - acc: 0.9361 - val_loss: 0.3280 - val_acc: 0.8998
Epoch 142/150
500/500 [=====] - 30s 60ms/step - loss: 0.1808 - acc: 0.9363 - val_loss: 0.3285 - val_acc: 0.8996
Epoch 143/150
500/500 [=====] - 30s 60ms/step - loss: 0.1769 - acc: 0.9368 - val_loss: 0.3283 - val_acc: 0.9004
Epoch 144/150
500/500 [=====] - 30s 61ms/step - loss: 0.1763 - acc: 0.9384 - val_loss: 0.3309 - val_acc: 0.9009
Epoch 145/150
500/500 [=====] - 30s 61ms/step - loss: 0.1774 - acc: 0.9358 - val_loss: 0.3296 - val_acc: 0.9011
Epoch 146/150
500/500 [=====] - 30s 60ms/step - loss: 0.1790 - acc: 0.9368 - val_loss: 0.3291 - val_acc: 0.9000
Epoch 147/150
500/500 [=====] - 30s 60ms/step - loss: 0.1764 - acc: 0.9367 - val_loss: 0.3291 - val_acc: 0.9001
Epoch 148/150
500/500 [=====] - 30s 60ms/step - loss: 0.1742 - acc: 0.9381 - val_loss: 0.3278 - val_acc: 0.9011
Epoch 149/150
500/500 [=====] - 30s 60ms/step - loss: 0.1802 - acc: 0.9353 - val_loss: 0.3286 - val_acc: 0.9009
Epoch 150/150
500/500 [=====] - 30s 60ms/step - loss: 0.1778 - acc: 0.9363 - val_loss: 0.3277 - val_acc: 0.9010
10000/10000 [=====] - 2s 224us/step
Test loss: 0.3277245296806097
Test accuracy: 0.901

```

Ilustración 4: Resultado obtenido de la CNN

No obstante, al igual que en el caso de MNIST, se han probado diferentes técnicas diferentes, las cuales se han desechado debido a que el algoritmo no sufría una mejoría en la tasa de acierto. Las otras técnicas que se han probado son:

- Learning Rate Annealing similar al que se probó con MNIST, con decremento y reinicio.
- Dropout

3.- Corpus externo: Blood-Cells

Presentación del Corpus

Por último, para este ejercicio final, se ha pedido que se realice un ejercicio relacionado con los dos anteriores y con lo visto en la asignatura durante el curso. En este caso, se ha utilizado un corpus totalmente diferente para entrenar una red convolucional parecida a la de CIFAR.

El corpus en este último experimento, llamado Blood-Cells, cuenta con un conjunto de aproximadamente 12400 imágenes de células de 4 tipos diferentes. Para la parte del entrenamiento se utilizarán 10000 imágenes, y para el test, las otras 2400. Esta partición viene dada por el formato de la carpeta de descargar del corpus, en la que ya vienen estas imágenes separadas. El corpus además de tener un test de 2400, cuenta con otro test más simple de solo 70 imágenes, que se ha descartado para este experimento.

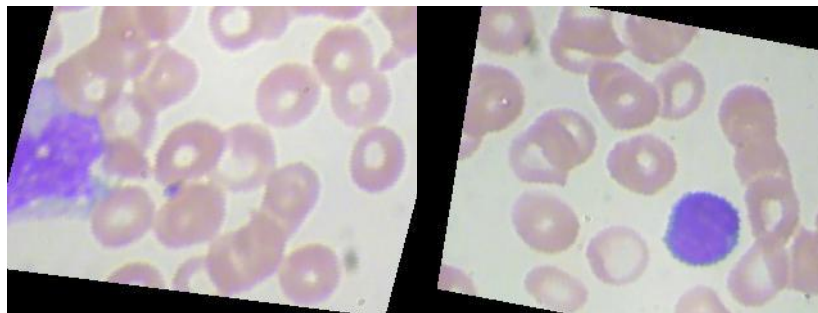


Ilustración 5: Ejemplo de imágenes del corpus

Las imágenes en sí, tienen una resolución inicial de 320x240, en formato RGB. Puesto que el tamaño original de las imágenes es demasiado grande para ser utilizado por nuestra red convolucional (el tiempo de ejecución se incrementaría demasiado), el rescalado de las imágenes es obligatorio, por lo que se comentará más adelante.

A diferencia de los ejercicios anteriores, en las que los corpus de MNIST y CIFAR se podían importar directamente de la librería de Keras, este nuevo corpus ha sido descargado de Kaggle, página web en la que se puede encontrar una gran variedad de datasets para redes neuronales.

Uso del corpus

Al ser un corpus externo a Keras, la importación del dataset en Google Colab se complica.

Para este experimento en concreto, se ha optado por utilizar Google Drive para la carga del dataset en el cuaderno de Colab. Una vez subido el dataset a Drive, se importa

“drive” de la librería “google.colab” que permite montar en la máquina de Colab, la ruta de archivos pasada como parámetro y que existe en la cuenta Drive vinculada.

Carga del dataset

Con la ruta de ficheros montada, se define el método “get_images(folder)”, con el que posteriormente, se podrá obtener las matrices de las imágenes y sus respectivas clases, tras un procesamiento de estas.

En este método, se recorre dentro de la ruta dada “folder” cada una de las carpetas de tipos de imágenes, para convertir estas imágenes a la representación tradicional de vectores con valores de 0 a 255 por pixel, utilizando distintas librerías de Python para ello. Simultáneamente, se guarda además la clase a la que pertenece cada imagen. Además, antes de la conversión imagen-vector, las imágenes son rescaladas tal y como se había mencionado anteriormente que era obligatorio.

Utilizando este método “get_images()” se carga el conjunto de imágenes tanto para la parte del training como la parte del testing, de forma similar a CIFAR. Por último, una vez cargadas estos conjuntos, se aplica la normalización de los dos conjuntos para que en vez de matrices con valores de 0 a 255, sea con valores de 0 a 1, y además del método “to_categorical()” de Keras, para convertir al formato one-hot, los vectores de clases del training y el test.

Convolutional Neuronal Network

Visto todo lo anterior, solo queda explicar lo importante del experimento: la red convolucional.

Como se ha adelantado al principio, la red convolucional es parecida a la utilizada en CIFAR, y también se ha utilizado Keras para su implementación. Esto se debe a que, para esta red convolucional, se ha partido de la utilizada en CIFAR dados sus buenos resultados. También se han utilizado diferentes tamaños para las particiones (batch_size), debido a que el número de datos es más reducido que en CIFAR, y diferente número de “epochs”, siendo fijados estos finalmente en 50 y 75 respectivamente.

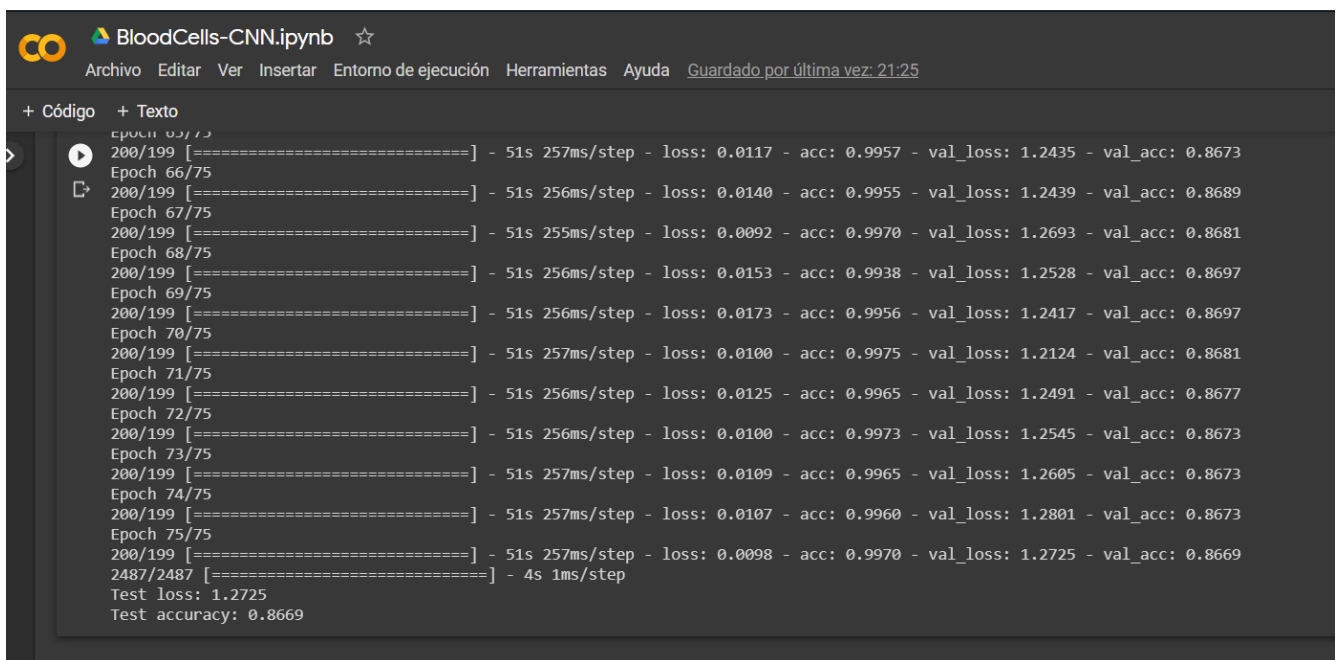
En cuanto a técnicas que evitan el sobreajuste de la red, se han empleado las mismas que en CIFAR, sobre las cuales se han probado diferentes parámetros como la “p” del Gaussian-Noise, los desplazamientos del Data Augmentation o las bajadas del Learning Rate Annealing. No obstante, ninguna diferencia en los parámetros ha resultado en una mejora de la clasificación, salvo la posición de la capa de Batch-Norm, que esta vez se ha optado por situarla antes de la función de activación, ya que esta modificación sí que ha aumentado el acierto en la clasificación.

Por último, por lo que respecta a la estructura de la CNN, se ha modificado respecto a la de la CIFAR. Se siguen aplicando 3 capas de convoluciones iguales por cada bloque, aunque el número de bloques ha sido modificado.

Esto se debe a que, en este corpus, se ha experimentado con la resolución de las imágenes. Inicialmente, el rescalado de las imágenes se ha aplicado para que dé como resultado imágenes de 32x32, con lo que se ha llegado a obtener una buena tasa de acierto (sobre 80%) sin tener que modificar la estructura de la CNN base. No obstante, se han realizado pruebas posteriores con resoluciones de 64x64 y, finalmente, 128x128, que han resultado en un aumento de la precisión de la CNN resultante. Para cada aumento de la resolución, se ha tenido que incorporar un nuevo bloque de capas convolucionales, con un número de filtros el doble que el bloque inmediatamente anterior. Esto quiere decir que si para una resolución de 32x32, requeríamos de una capa final con 512 filtros, para 64x64 necesitábamos un bloque más de 1024 filtros, y para 128x128, otro bloque más de 2048 filtros.

Como el último bloque, está conectada directamente con la red que se encarga de la clasificación, el número de neuronas (parámetro de la función Dense() de Keras) de esta red neuronal tiene que ser igual al número de filtros que tiene el último bloque (2048 tras fijación del rescalado a 128x128).

La tasa de acierto obtenida por esta nueva CNN sobre el corpus Blood-Cells, se muestra en la siguiente imagen:



```

+ Código + Texto
epoch 0/75
200/199 [=====] - 51s 257ms/step - loss: 0.0117 - acc: 0.9957 - val_loss: 1.2435 - val_acc: 0.8673
Epoch 66/75
200/199 [=====] - 51s 256ms/step - loss: 0.0140 - acc: 0.9955 - val_loss: 1.2439 - val_acc: 0.8689
Epoch 67/75
200/199 [=====] - 51s 255ms/step - loss: 0.0092 - acc: 0.9970 - val_loss: 1.2693 - val_acc: 0.8681
Epoch 68/75
200/199 [=====] - 51s 256ms/step - loss: 0.0153 - acc: 0.9938 - val_loss: 1.2528 - val_acc: 0.8697
Epoch 69/75
200/199 [=====] - 51s 256ms/step - loss: 0.0173 - acc: 0.9956 - val_loss: 1.2417 - val_acc: 0.8697
Epoch 70/75
200/199 [=====] - 51s 257ms/step - loss: 0.0100 - acc: 0.9975 - val_loss: 1.2124 - val_acc: 0.8681
Epoch 71/75
200/199 [=====] - 51s 256ms/step - loss: 0.0125 - acc: 0.9965 - val_loss: 1.2491 - val_acc: 0.8677
Epoch 72/75
200/199 [=====] - 51s 256ms/step - loss: 0.0100 - acc: 0.9973 - val_loss: 1.2545 - val_acc: 0.8673
Epoch 73/75
200/199 [=====] - 51s 257ms/step - loss: 0.0109 - acc: 0.9965 - val_loss: 1.2605 - val_acc: 0.8673
Epoch 74/75
200/199 [=====] - 51s 257ms/step - loss: 0.0107 - acc: 0.9960 - val_loss: 1.2801 - val_acc: 0.8673
Epoch 75/75
200/199 [=====] - 51s 257ms/step - loss: 0.0098 - acc: 0.9970 - val_loss: 1.2725 - val_acc: 0.8669
2487/2487 [=====] - 4s 1ms/step
Test loss: 1.2725
Test accuracy: 0.8669

```

Ilustración 6: Resultado obtenido del corpus Blood-Cells

Esta tasa de acierto se puede considerar eficiente, ya que estamos tratando esta vez con un corpus más reducido que el de CIFAR, por lo que procesa menos imágenes para su entrenamiento y aun así proporciona una tasa de acierto por encima del 86%.