Trabajo TP6-2: Reconocimiento de patrones con Redes Neuronales

1 - Objetivos

El objetivo general es **resolver mediante redes neuronales problemas reales de clasificación** utilizando Keras, una API sencilla para Redes Neuronales, que permite entrenar y ejecutar redes neuronales utilizando Tensorflow como *back-end*. Al igual que en la práctica 5, se utilizará Python dentro del entorno Google Colab.

El objetivo de la práctica es encontrar una red que tenga el menor error de clasificación posible, utilizando ***accuracy*** como métrica de comparación. Ve construyendo una tabla que refleje las diferentes redes que has probado, y los resultados que obtienen. Como parte de la documentación deberás presentar esta tabla, discutida y comentada adecuadamente.

Aunque el objetivo del trabajo sea obtener una *accuracy* lo más alta posible, de cara a la evaluación lo que más se va a tener en cuenta es el **proceso de diseño y la interpretación** de los diferentes resultados obtenidos, no el hecho de llegar a un valor determinado de accuracy.

2 - Dataset MNIST

El principal dataset que se va a utilizar es un dataset para el reconocimiento de dígitos manuscritos, MNIST, Figura 1. El dataset incluye 60.000 muestras de entrenamiento y validación, y 10.000 muestras de test, que sólo deben utilizarse para comprobar el funcionamiento de la red entrenada (nunca para entrenar!).

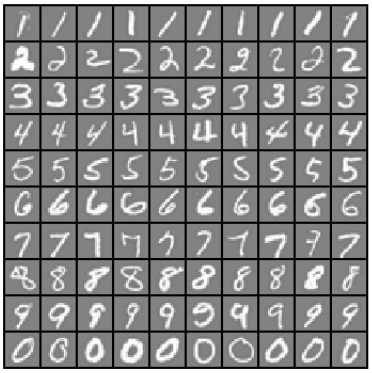

Figura 1. Ejemplos de datos de dígitos manuscritos

Como atributos para la clasificación se utilizarán directamente los niveles de intensidad de los 28x28 píxeles de cada muestra, es decir, los valores de los píxel, tal cual.

2.1 - Desarrollo del trabajo

Haz una copia en tu drive del Notebook de Ipython suministrado: (notebook: TP6_MLP.ipynb). Estudia y ejecuta cada sección para entender su funcionamiento.

Modifica la arquitectura y los hiper parámetros de la red para mejorar su funcionamiento. Como mínimo deberás probar redes de los siguientes tipos:

1. **Perceptrón**. El notebook que se suministra como punto de partida incluye el ejemplo para definir y entrenar este tipo de red. Compara los resultados que se obtienen para diferentes algoritmos de entrenamiento/optimización (prueba dos distintos) y funciones de activación (prueba dos distintas).

2. **Perceptrón multi-nivel con una capa oculta**. Define una red para reconocimiento de patrones con una capa oculta. El principal parámetro que debes explorar es el **número de neuronas de la capa oculta**. Estudia las transparencias de clase para saber qué tipos de funciones de activación deberías probar. Si ves que se produce sobreajuste, prueba alguna técnica para reducirlo.

3. **Perceptrón multi-nivel con dos capas ocultas**. Define una red para reconocimiento de patrones con dos capas ocultas, e intenta mejorar los resultados obtenidos en el apartado anterior. De nuevo, los principales parámetros a explorar son el **número de neuronas de las capas ocultas**.

Notas:

1. Al probar distintos algoritmos de optimización y distintas funciones de activación, puedes observar si mejora la convergencia (tiempo de entrenamiento, y error obtenido). Pero el **objetivo principal es encontrar un tamaño de red adecuado** para el problema, y **controlar el sobreajuste**.

2. Analiza si se produce **sobreajuste** y prueba diferentes estrategias para evitarlo. **Recuerda que añadir capas de dropout y términos de regularización** ayuda a reducir este problema, sobre todo en redes de mayor complejidad.

3. Utilizar *validation_split*=0.1 permite analizar si se está produciendo sobre-ajuste, pero desaprovecha un 10% de los datos de entrenamiento. En algunos casos puede ser interesante re-entrenar la red unas pocas épocas con todos los datos de entrenamiento (sin *validation_split*) para mejorar su tasa de aciertos.

4. Recuerda mostrar y comentar algún **ejemplo de imagen mal clasificada** por la red que mejor se comporte.

3 - Dataset CIFAR 10

Una vez que se haya entendido bien el funcionamiento de las redes neuronales con el dataset anterior, vamos a utilizar el dataset CIFAR 10[1], un dataset de mayor complejidad que incluye imágenes en color de 32x32 píxeles de 10 objetos (clases) diferentes, Figura 2.

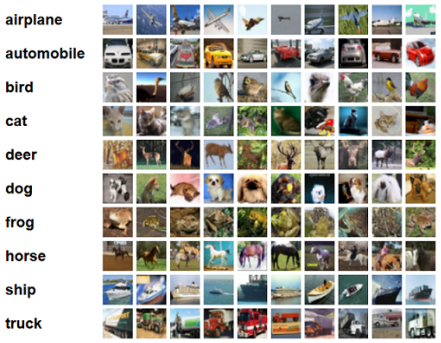

Figura 2. Clases consideradas en el dataset CIFAR 10

Este dataset se va a utilizar para **comparar** el comportamiento de un **perceptrón multicapa** y una **red convolucional**. Se incluye código de ejemplo para definir y entrenar una red convolucional para el dataset de MNIST (notebook: TP6_CNN.ipynb)

Para este dataset se pide:

1. Modifica la carga de datos, y la configuración inicial de la red si lo ves necesario, para que cargue/utilice el dataset CIFAR10 en lugar de MNIST. Puedes pasar a blanco y negro las imágenes, o hacer que la red trabaje con imagenes en color (n x m x 3), y siempre, normalizar los valores (atributos con valores entre 0 y 1).
2. Entrena un perceptrón multi-capa y evalúa los resultados que obtiene para CIFAR10. Compara estos resultados con los obtenidos en el dataset anterior (MNIST)
3. Entrena una red convolucional y evalúa los resultados que obtiene con CIFAR10. Compara estos resultados con los obtenidos para el perceptrón.
4. Opcional: Repite el último paso, entrenar la red convolucional, utilizando el dataset CIFAR 100[2], con 100 clases diferentes. En este caso, recuerda asegurarte que el Colab está utilizando una configuración con GPU[3].

3.4 Documentación a entregar

- Puedes copiar el notebook en tu Google Drive y editarlo, añadiendo las secciones de código y de texto que necesites para añadir tus comentarios, explicaciones y respuestas.
- Es **obligatorio** que en los **notebook se guarden los resultados de la evaluación** de cada celda de código (para no tener que volver a ejecutarlas). Por defecto los notebooks estan configurados asi (Edit→Notebook Settings → NUNCA actives: *Omit code cell output when saving this notebook*)
- Una vez hayas terminado, **descarga los dos notebook en formato ipynb** y súbelos a Moodle en un zip con el nombre NIP_TP62.zip a la tarea habilitada para el trabajo.

[1] https://keras.io/api/datasets/cifar10/
[2] https://keras.io/api/datasets/cifar100/
[3] https://colab.research.google.com/notebooks/gpu.ipynb