Inteligencia Artificial II

LAB 03
Implementación de modelos neuronales profundos

1. Objetivo del laboratorio

*Desarrollar de forma autónoma distintas implementaciones de **redes neuronales profundas de aprendizaje supervisado** que permitan resolver distintos casos de uso.*

2. Elementos a utilizar:

- Lenguaje *Python*
- Librerías: numérica *NumPy*, estructuras de datos *pandas*, gráfica *Matplotlib* (opcional si se quieren implementar gráficas) y *pyTorch*.
- Editor Jupyter, VSCode, Colab.
- Archivos .csv proporcionados

3. Práctica 1 (CNN para clasificar imágenes de frutas)

Objetivo

Utiliza lo aprendido en clase respecto a Deep Learning y redes convolucionales para construir un clasificador de imágenes. El dataset usado será *Fruits and Vegetables,* un dataset de imágenes a color RGB de 4GB que consta de imágenes separadas en un conjunto de entrenamiento y otro de test y clasificadas por categorías, según se describe en la página de Kaggle de donde se puede descargar el dataset con el enlace proporcionado más abajo.

Crea una red convolucional de la forma que se indica en el apartado de "**Implementación**". Responde a las preguntas que se plantean en "**Cuestiones**". El objetivo es **reconocer la fruta representada en cada imagen**.

Implementación

Crea el notebook *L3P1-FandV.ipynb*.

Descarga el dataset a través del enlace https://www.kaggle.com/datasets/moltean/fruits

- En el directorio *Training* Las imágenes en sí no están etiquetadas, la etiqueta vendrá proporcionada por la misma carpeta que las contiene:
- Debes de preprocesar los datos: leer la imagen JPEG, convertirla en una matriz y crear el conjunto de entrenamiento añadiendo la correspondiente etiqueta (para poder realizar el entrenamiento)
- Opcionalmente se puede hacer data augmentation (si los resultados con el dataset tal cual no fueran satisfactorios).

Crea una red convolucional secuencial. Juega con los tamaños de los filtros y decide el tamaño de las capas de '*pooling*'. Utiliza al menos 3 capas de convolución. En el caso de las funciones de activación, lo normal en este tipo de redes es usar *ReLU* en todas las capas, menos en la de salida que se debe de usar *softmax* para clasificaciones no binarias. Para actualizar los pesos, usaremos el optimizer '*RMSprop*' y la función de error '*categorical_crossentropy*'. Prueba con distintos '*learning rates*'.

Desarrolla las distintas fases de un predictor: entrenamiento + validación y predicción. Esta última se hará con las imágenes de test que vienen en el dataset. Tienes que dar como salida:

- Las tres categorías en las que con mayor probabilidad la imagen es clasificada por la red, en orden descendente de probabilidad
- La propia imagen pintada

Cuestiones

Continúa en la memoria respondiendo a las siguientes cuestiones

1. Explica las funciones y el algoritmo para convertirlas a matrices. ¿Qué es cada matriz? ¿Qué tamaños tienen las imágenes? ¿Has normalizado? ¿Cómo y por qué?
2. Entrena la red usando como criterio de parada un **loss ≤ 0,2**. Recoge en la memoria los valores de *loss* y *accuracy* tanto del conjunto de entrenamiento como del de validación de los distintos experimentos que hayas llevado a cabo indicando la arquitectura de cada una de las redes empleadas. Dibuja la arquitectura de red que mejor clasifica y pinta, para esa red, la gráfica de variación del error (loss) en función de las epochs, para los conjuntos de entrenamiento y validación. Explica cómo es la capa de entrada a la red y la capa de salida.
3. Clasifica el conjunto de imágenes de prueba y recoge en la memoria la salida obtenida para cada imagen tal y como se indica en el apartado **implementación**.
4. Teniendo en cuenta las imágenes predichas cuyas '*accuracies*' están bastante repartidas entre las tres categorías obtenidas, indica cuales son estas imágenes, entre que categorías se confunden y por qué crees que ocurre esto.

4. Práctica 2 (CNN para clasificar imágenes de Rayos X de neumonía)

Objetivo

Analiza el dataset de radiografías torácicas **Chest X-Chest X-Ray Images (Pneumonia)**. El dataset contiene 3 carpetas (train,test,val) y por cada carpeta hay dos etiquetas (Pneumonia/Normal). Dentro de la carpeta Pneumonia se distingue entre bacteriana y vírica.

Descripción: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

Link Descarga: https://drive.google.com/file/d/1Fy1BBM8fzS3Q5cREqd5gMwOrdsujmyj4/view?usp=sharing

Realiza dos clasificaciones:
Pneumonia(ambos tipos como si fuesen uno solo) vs Normal
Pneumonia bacteriana vs Pneumonía vírica vs Normal (3 opciones a clasificar)

Responde a las preguntas que se plantean en "**Cuestiones**".

Implementación

Crea el notebook *L3P2-Pneumonia.ipynb*. El programa en Python deberá responder a los siguientes puntos:

Prepara los datos de entrada (puede que no haya suficientes) y divídelos en los datasets necesarios.

Crea una red convolucional secuencial. Juega con los tamaños de los filtros y decide el tamaño de las capas de '*pooling*'. Utiliza al menos 3 capas de convolución. En el caso de las funciones de activación, lo normal en este tipo de redes es usar *ReLU* en todas las capas, menos en la de salida

que se debe de usar *softmax* para clasificaciones no binarias. Para actualizar los pesos, usaremos el optimizer '*Adam*' y la función de error '*categorical_crossentropy*'. Prueba con distintos 'learning rates'.
Desarrolla las distintas fases de un predictor: entrenamiento + validación y predicción.
Para cada imagen tienes que dar como salida:

- Las probabilidades de pertenencia a cada posible categoría en orden descendente de probabilidad, identificando claramente la categoría por su nombre
- La propia imagen pintada

Cuestiones

1. Explica las funciones y el algoritmo para convertir las imágenes a matrices. ¿Qué tamaños tienen las imágenes? ¿Has normalizado? ¿Cómo y por qué?
2. Entrena una red para cada clasificación usando como criterio de parada un **loss ≤ 0,2**. Recoge en la memoria los valores de *loss* y *accuracy* tanto del conjunto de entrenamiento como del de validación de los distintos experimentos que hayas llevado a cabo indicando la arquitectura de cada una de las redes empleadas. Dibuja la arquitectura de red que mejor clasifica y pinta, para esa red, la gráfica de variación del error (los) en función de las epochs, para los conjuntos de entrenamiento y validación.
3. ¿Qué es más fácil: distinguir entre dos tipos de neumonía o distinguir si el paciente está sano o no? ¿Por qué?
4. ¿En caso de no tener suficientes imágenes, que técnicas usarías? Explícalas con detalle
5. ¿Se te ocurre alguna manera de modificar las imágenes para que el modelo mejore (en velocidad de entrenamiento o precisión)?

5. Forma de entrega del laboratorio:

La entrega consistirá en un fichero comprimido RAR con nombre LAB03_GRUPOxx.RAR subido a la tarea **LAB3** que **contenga únicamente**

1. **Por cada práctica** un notebook de Jupyter (archivos con extensión **.ipynb**).
2. Una **memoria del laboratorio** en PDF.

Las entregas que no se ajusten exactamente a esta norma NO SERÁN EVALUADAS.

6. Rúbrica de la Práctica:

1. IMPLEMENTACIÓN: Multiplica la nota del trabajo por 0/1

Siendo una práctica de IA, todos los aspectos de programación se dan por supuesto. La implementación será:

- Original: Código fuente no copiado de internet. Grupos con igual código fuente serán suspendidos
- Correcta: Los algoritmos deben estar correctamente programados. El programa funciona y ejecuta correctamente todo lo planteado en el apartado "*Cuestiones*" de cada práctica.
- Comentada: Inclusión (**obligatoria**) de comentarios.

2. MEMORIA DEL LABORATORIO

Obligatorio redacción clara y correcta ortográfica/gramaticalmente con la estructura recogida en la plantilla.
Calificación de las cuestiones:

PRÁCTICA	CUESTIÓN	VALORACIÓN
Práctica 1	Cuestión 1	2
	Cuestión 2	1
	Cuestión 3	1
	Cuestión 4	1
Práctica 2	Cuestión 1	1
	Cuestión 2	1
	Cuestión 3	1
	Cuestión 4	1
	Cuestión 5	1