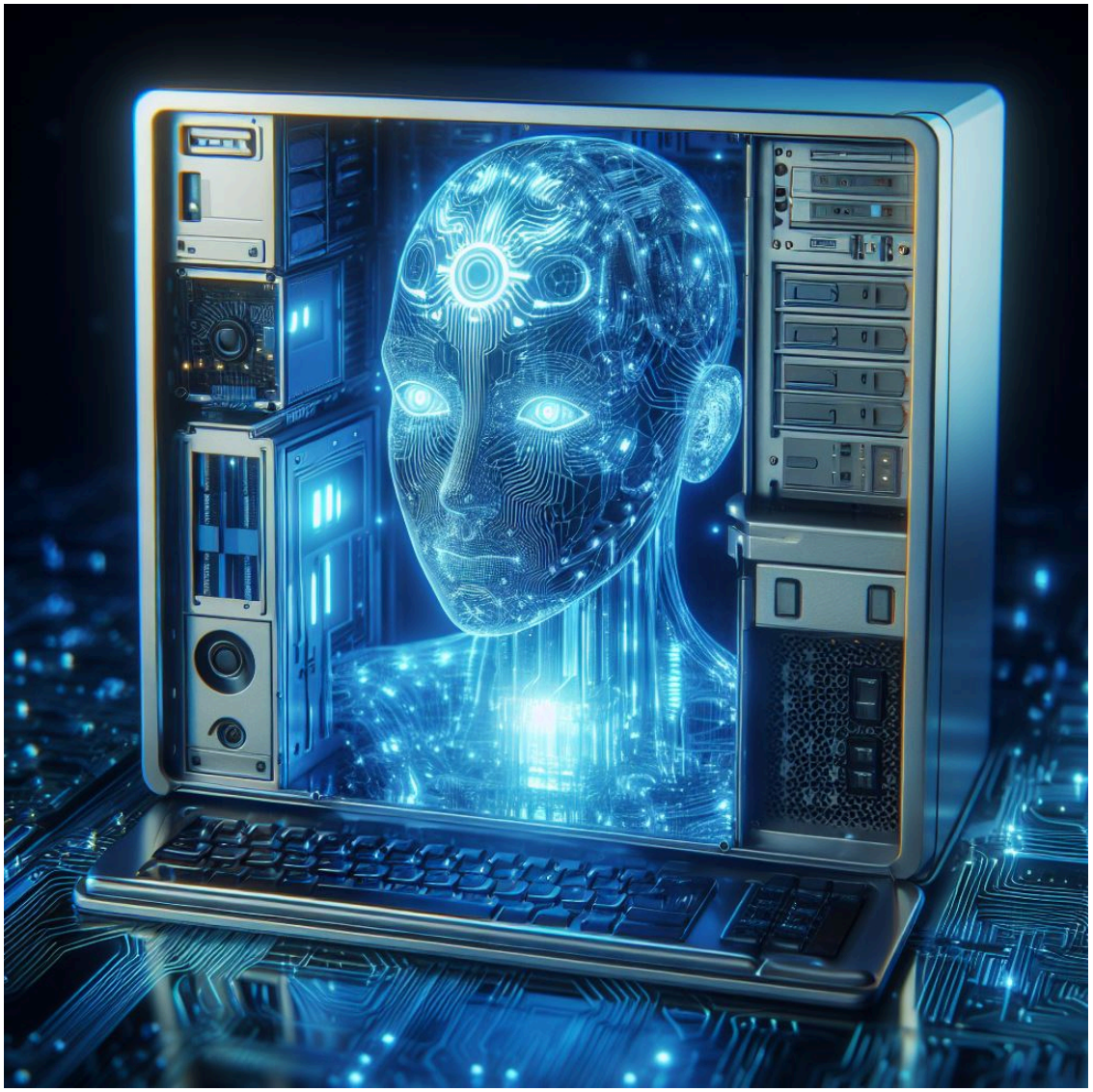
Red Neuronal Multimodal Numérica

Sergio Hidalgo



Índice

1- Propuesta inicial

Se pretende realizar una red neuronal que sea capaz de evaluar datos sobre números del 0 al 9, por lo que habrá 10 neuronas de salida. Recibirá tres tipos distintos de datos, que serán convertidos a entradas binarias (o a polares, dependiendo del avance del proyecto):

- Datos de texto: datos de entrada de texto plano, que tendrán naturaleza de tipo:
    - Binaria: podrá recibir un número en binario y predecir su clase. Necesita 4 entradas.
    - Hexadecimal, ASCII: podrá recibir un número en código ASCII y predecir su clase. Necesita 8 entradas.
    - Léxico Español: parecido al anterior, pero recibirá letras en ASCII, predecirá la clase según una palabra (uno, dos, tres…). Necesita 48 entradas (6 letras es el número máximo de entradas que tiene.

- Datos de audio: datos de entrada en archivo .mp3, de los números de una voz en español, para ello se reducirá el bitrate a 192 kbps, teniendo 196608 bits (y por tanto entradas), la idea es que conste de las siguientes partes:
    - Reconocimiento de letra: Reconocer la letra que se está diciendo en un punto en cuestión
    - Reconocimiento de palabra: Las letras forman una palabra, será identificada con la red de texto léxico.

- Datos de imagen: datos de entrada de imágenes, se reconocerá el número (escrito de forma numérica) en una imagen que se bajará a 32 px de resolución. Por lo que teniendo 4 bits por píxel tendrá 4096 bits de entrada.

2 - Contrapropuesta

Tras investigar y trastear con lo anteriormente mencionado (con imágenes y texto), se llegó a la conclusión de que el proceso directo y masivo a través de capas densas convencionales era inviable por dos motivos principales:

1. **Coste computacional**: Tener que gestionar de forma directa todos los datos de una imagen, tenía un coste muy elevado y unos resultados nefastos, en las primeras pruebas solo se llegaba hasta el 0.203 de precisión de predicción de imágenes y se tardaba en entrenar la red para alcanzarlo (por este motivo el límite de épocas está en 5000).

2. **Ruido entre los patrones**: Al introducir un patrón a una red neuronal densa tradicional, no se tiene en cuenta un hecho importante, y es que el patrón que se quiere predecir puede haber sido desplazado por otra información un par de entradas.

La solución a estos problemas viene dada por el uso de redes neuronales convolucionales (sobre todo en las imágenes), pues permiten “filtrar” patrones (según el tamaño del kernel) De manera que aunque ahora tengan ruido a su alrededor se podrá determinar de qué clase es un conjunto de datos, además de reducir el coste computacional, pues habrá partes de la información que no serán relevantes y serán omitidos por el filtro.

3 - Procesado y obtención de datos

3.1 - Datos de texto

Tras plantear el problema en cuanto al procesado, se decidió que finalmente el modelo respecto a los datos sólo contará con la predicción de una palabra en español asociada a la clase de dicho número (ejemplo: cero es la clase x0, o lo que es lo mismo, cuando sea 0 la clase 0 se activará). El modelo se entrenará con datos generados a través de un script de python **[Anexo 1]**. El formato en el que se guardan los datos es txt o parquet, generando el txt los datos en texto plano y el parquet en binario.

3.2 - Datos de audio

La idea principal era grabar unos cuantos audios recitando los números del 0 al 9, luego replicarlos y modificar sus parámetros de forma pseudoaleatoria, para así poder aumentar la cantidad de datos sin necesitar muchas voces y/o grabaciones.

3.3 - Datos de imagen

Se ha utilizado la base de datos de mnist **[Anexo 2]**, entrenando con dos capas convolucionales una red para que pueda reconocer los números escritos

4 - Arquitectura de la aplicación

La aplicación se compone de tres grandes módulos:

1. **Procesado de datos**: En este módulo se encuentra la lógica que procesa los datos antes de mandarlos a la red.

2. **Aplicación principal**: Aquí se recoge tanto la parte lógica de la aplicación como la parte gráfica

3. **Módulo del modelo:** Conforman los ficheros que describen el comportamiento del modelo

[Anexo 3]

5 - Arquitectura del modelo

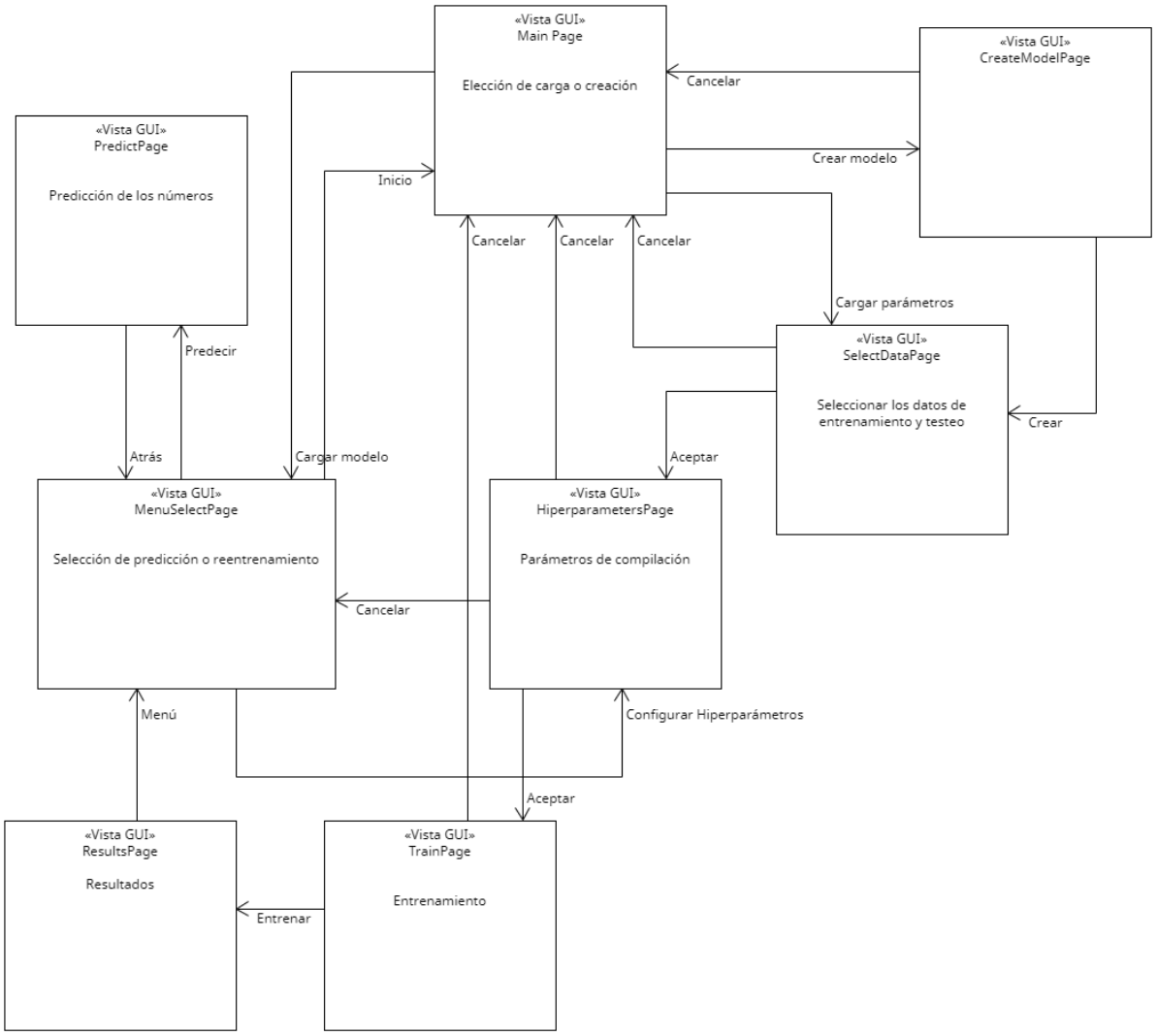
La idea principal era generar los modelos por separado, y luego juntarlos para entrenarlos, no obstante hubo una serie de problemas con ello

1. **Desconocimiento ante la herencia de los modelos**: Keras permite extender las clases de modelo que lo conforman y tienen una guía sobre ello en su documentación, pero al descubrirlo ya era demasiado tarde
.
2. **Incompatibilidad de la longitud de entrenamiento:** En caso de que un modelo tuviese más vectores de entrenamiento que el resto (aunque este problema se podía solucionar con redundancia de datos en el modelo con menos patrones y/o eliminando datos del conjunto largo)

3. **Desconocimiento general del funcionamiento de los datasets de keras**: Todos los ejemplos encontrados para conformar una red de predicción de texto o bien hacían uso de los datasets incluidos en keras o bien extraían el texto con la función "*text_dataset_from_directory"* (función que se ha utilizado para extraer los datos) y esto repercutió en que no se pudiese acoplar al modelo general como se pensó inicialmente, pues este estaba programado pensando en que los datos tendrían forma: (x_train, y_train) , no en un solo dataset con ambas.

6 - Interfaz gráfica

La interfaz gráfica sigue el siguiente diagrama:



[Anexo 4]

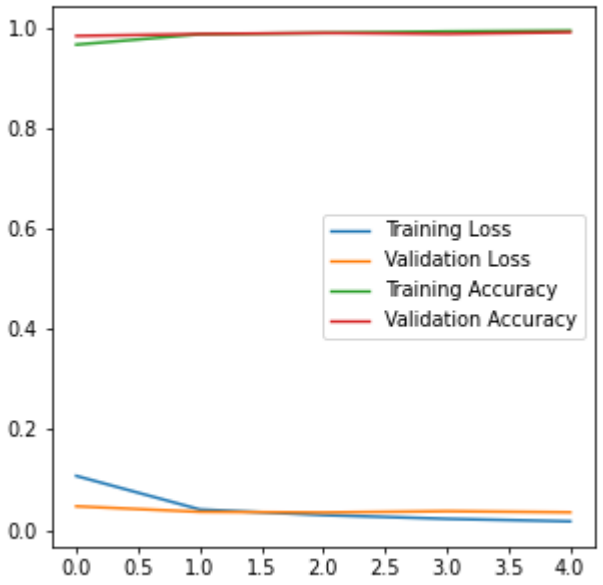
7 - Resultados

La predicción de imágenes se ha llevado a cabo satisfactoriamente y el texto en última instancia también. Por otro lado el audio no ha habido tiempo de mirarlo, dado que se estaba implementando el texto dentro de la red

7.1 - Ejemplo de ejecución de imágenes

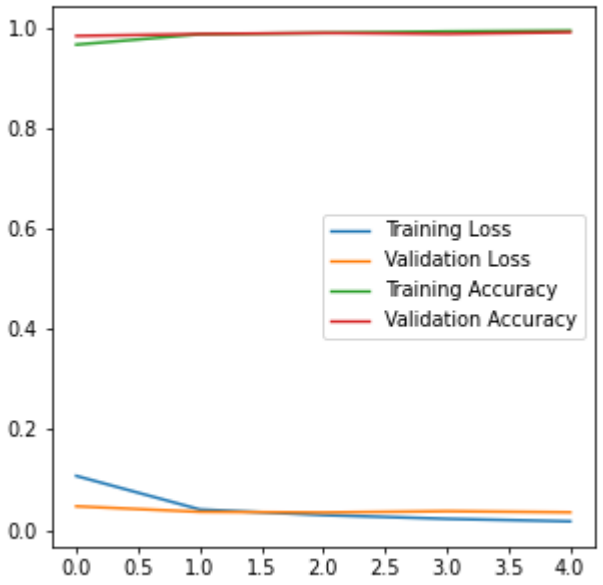
Primero se ha de pulsar el botón “crear modelo” o bien pulsar “cargar modelo” o bien “cargar configuración” para que la información se obtenga del archivo “config/default_config.json” tras esto se selecciona el directorio con las carpetas “test” y “train” de las imágenes (es decir, “image_data”) y se seleccionan el número de épocas (5

son las que se han utilizado en este caso) finalmente se regresa al menú desde el apartado de resultados y ahí se puede elegir o guardar el modelo o predecir números.
El siguiente gráfico es el gráfico de entrenamiento de la red de predicción de imágenes.



7.2 - Ejemplo de ejecución de texto

Se han de seguir pasos equivalentes para este caso.
El siguiente gráfico es el gráfico de entrenamiento de la red de predicción de imágenes.



7.3 - Ejemplo de ejecución de texto e imágenes

La vía más rápida de hacerlo es mediante la carga de parámetros (si se desea entrenar desde 0), por otro lado para observar las predicciones se deberá cargar el modelo ya entrenado “text_and_image.pkl”

8 - Conclusiones

Trás varias pruebas, cambios e iteraciones, parece ser que en este caso lo mejor es entrenar los modelos por separado y luego unirlos, actuando estos como un solo modelo que recibe un vector de datos conformado por texto e imagen y devuelve un vector de clases de longitud equivalente a los tipos distintos de datos, multiplicado por el número de clases. Tras esto se debe separar dicho vector y presentar cada salida. Este es el caso para el ejemplo particular de los números porque no hay una relación real entre el número dibujado y el número escrito, pero de ser otro caso, en vez de una concatenación en las últimas capas, se podrían unir las salidas de ambas redes para su posterior procesado en una tercera red.

8 - Anexos

[Anexo 1] - El script se encuentra en el directorio “scripts”
[Anexo 2] - https://github.com/golbin/TensorFlow-MNIST/tree/master/mnist/data
[Anexo 3] - El diagrama de clases se encuentra en “docs/diagrams”
[Anexo 4] - El diagrama de la gui se encuentra en “docs/diagrams”