CLASIFICACIÓN DE DEMENCIA POR MEDIO DE UN MODELO BASADO EN REDES NEURONALES HACIENDO USO DE IMÁGENES DE MRI

Juliana Moreno Rada

Bioingeniera

Fundamentos de Deep Learning

Facultad de Ingeniería.

Universidad de Antioquia.

Medellín, 2023.

Para este proyecto, se planteó un problema de clasificación de imágenes de resonancia magnética de sujetos con demencia. Se propuso un enfoque basado en redes neuronales, el proyecto contiene un solo notebook donde se realiza la carga y preprocesamiento de datos, se diseña el modelo y se ejecuta.

- **Datos**

El conjunto de datos utilizado para el diseño y entrenamiento del modelo consiste en un dataset obtenido en el repositorio de Kaggle. Este conjunto de datos consiste en imágenes de resonancia magnética etiquetadas como Mild Demented, Moderate Demented, Non Demented y Very Mild Demented. Estas etiquetas hacen referencia a un estado de demencia en el que se puede encontrar una persona, sin embargo, para efectos de este proyecto solo se usará dos etiquetas, las cuales serán Non Demented y Very Mild Demented, estos son los estados neuronales más alejados por lo que se pueden encontrar más diferencias en las imágenes.

Para poder acceder a los datos será necesario descargar el token de Kaggle para poder hacer uso de la API en colab y descargar los archivos necesarios en el entorno de Google Colab.

1. Crear una cuenta en Kaggle, si ya está creada, acceder a ella
2. Ir a las configuraciones de perfil: https://www.kaggle.com/settings/account
3. Descargar el json dando clic en el botón “Create New Token” (Figura 1)

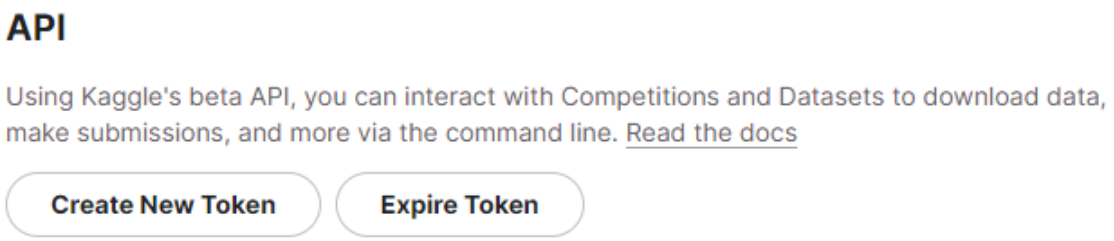

Figura 1. Descarga de token.

4. En el Notebook se indicará cuando subir el json que se descarga (Figura 2)

```
[ ]  from google.colab import files

     # Carga el archivo kaggle.json desde tu máquina local
     files.upload()
```

Figura 2. Carga del archivo json.

Una vez con los datos descargados en el entorno de Colab, se procede a generar el conjunto de entrenamiento y validación. Los datos están almacenados en 4 directorios que corresponden a las clases, para poder separar los dos conjuntos, se recorre cada directorio, se dividen las imágenes en conjunto de prueba y entrenamiento y luego se copian en los dos nuevos directorios (test y train) en sus respectivos directorios de clases. Una vez obtenidos los directorios de prueba y entrenamiento, se generan los conjuntos como objetos de tensor Flow haciendo, instanciando una clase ImageDataGenerator.

Los conjuntos se generan con los siguientes parámetros:

```
image_size = (64, 64)  # se ajusta el tamaño de imagen
batch_size = 1024
#classes = ['MildDemented',
'ModerateDemented','NonDemented','VeryMildDemented']
classes = ['NonDemented','VeryMildDemented']
```

Que indican que la resolución de las imágenes se establecerá en un tamaño de 64x64 pixeles, y se organizarán en baches de 1024 muestras. Además, se normalizan las imágenes en una escala de 0-255 en intensidad.

- **Diseño y arquitectura de los modelos**

Por la naturaleza de los datos, se optó por usar un enfoque basado en redes convolucionales y se propusieron dos arquitecturas.

ResNet

La primera propuesta que se hizo fue hacer transfer learning usando el modelo ya entrenado **ResNet**, disponible en las aplicaciones de Keras. Para esta primera solución, se implementaron las primeras capas del modelo original estableciendo los pesos como no entrenables, y para la salida, se definieron las siguientes capas:

1. Capa Flatten
2. Capa densa, 64 neuronas, función de activación relu
3. Capa densa, 2 neuronas, función de activación softamax

La función de pérdida escogida fue una cros-entropía y un optimizador Adam, se usó una rata de aprendizaje del 0.1 y se entrenó en 10 épocas (el tamaño de bache es el definido a la hora de definir los conjuntos de prueba y entrenamiento)

Métricas

Además de la exactitud, se decidió hacer un *Calsification Report* con el que se obtuvieron las métricas de Precisión, Recall y F1 Score, estas métricas son especialmente útiles en este problema ya que, como se mencionó, se requiere hacer clasificación, y, además, el problema está enfocado en el ámbito médico, es decir, se requiere obtener la mayor cantidad de verdaderos positivos.

Modelo Secuencial

Como segunda alternativa, se propuso una adaptación del modelo VGG16 con las siguientes capas:

1. Un bloque de dos capas convolucionales de 64 filtros con un tamaño de kernel de 3 seguidas de una capa de MaxPooling
2. Un bloque de una capa convolucional de 128 filtros con un tamaño de kernel de 3 seguida de una capa de MaxPooling
3. Un bloque de dos capas convolucionales de 512 filtros con un tamaño de kernel de 3 seguidas de una capa de MaxPooling
4. Un bloque de predicción compuesto por:
    a. Capa de Flatten
    b. Una capa Densa, seguida de una capa de Dropout
    c. La capa de salida, que consiste en una capa densa de dos neuronas y activación Softmax.

La función de pérdida escogida fue una cros-entropía y un optimizador Adam y las métricas usadas para evaluar el modelo fueron las mismas que las usadas para la anterior arquitectura

- **Iteraciones**

Una vez se corrieron los modelos por primera vez, se hicieron las siguientes modificaciones:

- Agregar más capas convolucionales
- Variar el tamaño de Kernel y número de filtros de las capas convolucionales
- Variar la ventana del pooling
- Variar la tasa de Dropout
- Variar la función de activación de la capa densa

- Agregar o quitar capas densas
- Variar los parámetros globales de entrenamiento del modelo, como el número de épocas, el tamaño del "*batche*" y la rata de aprendizaje.

- **Resultados**

Los resultados obtenidos con el modelo ResNet ya pre-entrenado fueron satisfactorios. Las métricas arrojadas por el reporte de clasificación demuestran que no hay desbalanceo de clases y que el modelo logra clasificar satisfactoriamente ambas clases en cuestión, aunque la exactitud no sea muy alta. Esto puede deberse a que las imágenes con las que se entrenó el modelo son muy diferentes a las usadas en este problema, y además, por cuestiones de recursos computacionales, la resolución de imágenes usada en este caso fue baja.

Por otro lado, analizando las diferentes variaciones del modelo secuencial alternativo que se planteó, se encontraron los siguientes resultados:

Agregar más capas convolucionales en realidad no afecta mucho el rendimiento del modelo, sin embargo, lo que si representa cambios significativos es variar la cantidad de filtros y el tamaño del kernel. Por requerimientos computacionales, se decidió usar un tamaño de imagen pequeña, por lo que al modificar el tamaño de filtro y aumentarlo, resultaba en un menor poder de clasificación del modelo, por lo tanto, un tamaño del filtro pequeño es útil para capturar características locales.

Además, al aumentar o disminuir mucho la ventana del pooling no se obtienen mejores resultados, así que se optó por una ventana de pooling de 2 la cual fue con la que mejores resultados se obtuvieron. De forma similar, se encontró que disminuir o aumentar mucho la tasa de dropout afectan los resultados del modelo, y se decidió dejar en 0,3 que había sido la propuesta inicialmente.

Con la mejor combinación de parámetros, se pudo obtener un modelo con una exactitud del 73 % y nuevamente, al observar el reporte de clasificación (Figura 3)

```
[ ]  report = classification_report(y_true, y_pred_labels)
     print(report)
```

```
                  precision    recall  f1-score   support

               0       0.80      0.64      0.71      1920
               1       0.68      0.83      0.75      1792

        accuracy                           0.73      3712
       macro avg       0.74      0.73      0.73      3712
    weighted avg       0.74      0.73      0.73      3712
```

Figura 3. Resultados del reporte de clasificación

Para terminar, a modo de conclusión, se encontraron buenos resultados con los enfoques planteados para dar solución al problema de clasificación. Por un lado, el transfer learning es una técnica que permite adaptar modelos muy complejos a problemas nuevos, con conjuntos de datos diferentes y escogiendo bien el modelo pre-entrenando y teniendo en cuenta la naturaleza de los datos que se están usando, se pueden obtener buenos resultados.

Por otro lado, las arquitecturas de modelos secuenciales basados en redes neuronales pueden ser muy amplias, lo que permite dar solución a una amplia gama de problemas. Escoger la combinación optima de parámetros y capas es una búsqueda en la que se debe tener en cuenta que objetivos se quieren cumplir para poder diseñar un modelo que cumpla con lo esperado.