Parametrización y segmentación de las capas de la retina mediante técnicas de aprendizaje profundo

Christian Mariscal Calvo
ETSIAE, Septiembre 2021.

Explicación de códigos

Estructura de códigos

- Paso 1: Iniciar sesión en Drive con:

  Correo: cmariscalvo99@gmail.com

  Contraseña: Zaragoza123

- En Google Drive/Colab Notebooks, hay 5 códigos:
  - **1.BilayerRetinaSegmentation**: Segmentación binaria de la retina
  - **2.MultilayerRetinaSegmentation**: Segmentación de las capas de la retina
  - **3.Multilabel Image Preprocessing**: Procesamiento del Dataset para multicapa.
  - **4.Thickness calculus**: Cálculo de espesores y gráficas 3D desde predicción.
  - **5. DataExtractionVol**: Cálculo de espesores y gráficas 3D desde archivo .vol

Se comentan todos menos BilayerRetinaSegmentation, ya que es prácticamente igual a MultiLayerRetinaSegmentation

Estructura de códigos

Los 4 primeros códigos se abren desde Colab. El último, desde ordenador propio.

Para usar los 4 primeros, es necesario iniciar sesión, como el propio código requiere en la primera celda.

Hay que clicar en el enlace e iniciar sesión con los siguientes datos:

Correo: cmariscalvo99@gmail.com

Contraseña: Zaragoza123

Para usar el último, es necesario descargar el archivo 1OD1.vol que está en la carpeta TFG de Drive y cambiar la ruta dentro del código *DataExtractionVol.*

MultilayerRetinaSegmentation

- Se compone de varias secciones, que se comentan a continuación.

1. Data preparation:
    1. Descomprime las carpetas de imágenes de Drive. La función *OneHot* aplica OneHot encoding a cada image. *OH2COLOR* devuelve las imágenes en OneHot a su situación original de máscaras coloreadas.
2. Dataset and Dataloader definition
    1. Se crean la clase Dataset y Dataloader para proceder a la carga de imágenes del modelo. Se define en un formulario el tamaño del dataset de entrenamiento y el batch size.
3. Metrics:
    1. La función *IoU* devuelve la métrica IoU, teniendo como inputs los resultados de la predicción y los valores de la máscara original.
    2. La función *accuracy* devuelve la métrica accuracy, teniendo como entrada la máscara predicha y la original.
4. GPU Acceleration
    1. Simplemente define el dispositivo como ‘cuda’ (uso de GPU) en el caso de que esté disponible.
5. Architectures
    1. Definición de las arquitecturas, importándolas de la librería *torchvision.models.segmentation*

MultilayerRetinaSegmentation

6. Multiprocessing
    1. En caso de no disponer de GPU, es conveniente seleccionar la opción ‘Multiprocessing’ para acelerar el proceso.
7. Training functions:
    1. Encontramos dos: *fit* (para GPU) y *fit_multi*(para multiprocessing). Estas funciones realizan el entrenamiento descrito en el trabajo (cálculo de *loss,* propagación y actualización de pesos).
8. Saving data in Drive
    1. Se guardan en Drive la gráfica de evolución del modelo, el diccionario con los parámetros del modelo y sobreescribe las métricas obtenidas en un fichero *Results.txt.*
9. Evaluation
    1. Primero carga una lista de modelos preentrenados, de los cuales el usuario elige uno.
    2. La función *visual_eval* permite mostrar los resultados obtenidos comparando la imagen OCT original, la máscara original y la predicción.
    3. La función *original_shape* hace un ‘resize’ de las imágenes a su tamaño original.
    4. La función *RGB* transforma las predicciones a formato RGB
    5. Si se activa el botón *DownloadResults*, se guarda en drive una carpeta zip con todas las predicciones del test.

Multilabel Image Preprocessing

- Se compone de dos secciones. En la primera, se hace un tratamiento de las imágenes. En la segunda, se obtienen las máscaras con cada capa coloreada y con un ‘resize’ para su entrada en el modelo.
- Por pasos:
- 1. Importar carpetas de Drive y descomprimirlas
- 2. Guardar en listas las imágenes que tengan máscaras de todas las capas.
- 3. Modificar las imágenes para evitar superposición, tal y como se explica en la redacción del trabajo.

Multilabel Image Preprocessing

- 4. Con este tratamiento, se crea un .zip llamada *CorrectDataset,* que incluye sólo imágenes con máscaras de todas las capas y evitando superposición.
- 5. La función *color_mask* colorea las máscaras. La función *treatment* reduce la pérdida de calidad provocada por el *resize* aplicado antes a las imágenes, tal y como se explica en la redacción del trabajo.
- 6. Se aplican las dos anteriores funciones a todas las imágenes de *CorrectDataset* y se llevan al acelerador GPU, obteniendo la carpeta *ColouredMasks.*

Thickness calculus

- Este código se compone de varios pasos (divididos en secciones) que nombro a continuación:
- 1. Previous Actions: Simplemente consiste en ‘montar’ la estructura de carpetas de Drive.
- 2. Architectures definition: procedemos a definir todas las arquitecturas del proyecto, importándolas o definiendo los bloques en el caso de UNet.
- 3. Upload a model: En un formulario seleccionamos la arquitectura deseada, que se elige de una lista de modelos de segmentación binaria preentrenados alojados en Drive.

Thickness calculus

- 4. OCTs Selection: Simplemente se inserta el nombre de la carpeta de OCTs que uno desee, que aparecen como una lista de las disponibles en Drive.
- 5. Se ordenan los elementos de la lista del paso anterior y se realizan sus predicciones binarias, añadiéndolas a la lista *pred_mask_list.*
- *6.* Thickness calculation: Procedemos a realizar el algoritmo descrito en la redacción del TFG, buscando comienzo y fin de retina y almacenando los espesores en un array.
- 7. Data interpolation: Como en la dirección ‘y’ sólo tenemos unas 30 imágenes, hace falta interpolar esos resultados para obtener una curva mucho más lisa, procedimiento realizado en este apartado.
- 8. 3D Graphs. Por último, se emplea la librería PlotLy para graficar los resultados obtenidos tras la interpolación. En concreto, se grafican comienzo y fin de retina, espesores y se proyecta un mapa de calor.

DataExtractionVol

1. El procedimiento es similar al de Thickness Calculus. En este caso, hay que extraer los datos binarios del archivo .vol.
2. Primero se definen las variables, sus tipos binarios, sus tamaños y sus ‘offsets’.
3. Después se recorre en un ‘bucle’ todos estos datos, transformando la información binaria en alfanumérica y guardándola en *value_items.*
4. Después se realiza la transformación de binario a imagen de los distintos OCTs del archivo .vol y también se almacenan las variables necesarias para el cálculo de los gráficos 3D en listas (SegArray, StartX, StartY, EndX, EndY). Estas variables son descritas en la redacción del proyecto.

DataExtractionVol

5. Se generan las listas que definen posiciones de comienzo y final de retina (ILM, RPE).

6. Se define la lista de espesores (RESTA), simplemente restando RPE-ILM. Después se recortan los bordes de imagen que la máquina leía mal.

7. Se procede a interpolar los datos y se grafican en 3D además de crear un mapa de calor, como en el código anterior.

¡GRACIAS POR SU ATENCIÓN!