Úlceras por Presión (UPP) - Colabs: (La última siempre es la versión final)

1. UPP Seg Clases.ipynb

<https://colab.research.google.com/drive/1NjA3BrMN_3io7X5WVwgEly-T-fPdnp-h?usp=sharing>

1. UPP Seg binaria.ipynb

<https://colab.research.google.com/drive/1X2INjizIzmGibhovSLZlD6HbYImzYiAL?usp=sharing>

1. UPP Seg Bin - PreProcesamiento .ipynb

<https://colab.research.google.com/drive/182z3nmorrcEp4btvFD1WOl9xCgAerzH0?usp=sharing>

1. UPP Seg Bin - Baseline.ipynb

<https://colab.research.google.com/drive/1QlQuSnfgtWlstsvTRfKCDYBqbx-6vfXK?usp=sharing>

1. UPP Versión principal.ipynb  
   <https://colab.research.google.com/drive/1TxxNFXByTvRzUhW6oBWue63jNabg-Bei?usp=sharing>
2. UPP Paper Seg Bin.ipynb

<https://colab.research.google.com/drive/1WqUzIdF6MbrWj9hqeVNjMAYYfCtRwgGA?usp=sharing>

1) UPP Seg Clases.ipynb

En primer lugar, me centré en el paper “original”, que busca clasificar cada tejido dentro de las úlceras, así como también el tejido de re-ep que se encuentra alrededor de las mismas:  
<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0264139>

Dataset:

**Data Availability:** Data cannot be shared publicly because of regulation of institutional review board of Far Eastern Memorial Hospital. Data are available from the institutional review board (contact via phone: +886-2-8966-7000 ext. 2152 and the e-mail: irb@mail.femh.org.tw.) for researchers who meet the criteria for access to confidential data. Alternatively, we uploaded the images labeled by our proposed methods (boundary-based and region-based) to the data repository site. The DOIs by two different methods are [S1 Dataset](https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0264139#pone.0264139.s007). Region-based label method <https://doi.org/10.6084/m9.figshare.17206904.v1> and [S2 Dataset](https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0264139#pone.0264139.s008). Boundary-based label method <https://doi.org/10.6084/m9.figshare.17206940.v1>. The images are collected from the open dataset of wound, the Medetec. Those labeled images were part of our testing sets. Researchers can train DL models with these two datasets to replicate our study results.

Trabajé por el momento con S1 y S2 Dataset (son muy pocos datos), más que nada en el preprocesamiento, porque no pude obtener el acceso a los datos reales con los que trabaja el paper (son muchos más datos). Mandé un mail a la dirección que figura en **Data Availability** y también a Che Wei Chang: [b92401068@gmail.com](mailto:b92401068@gmail.com) , el autor principal, pero no recibí respuestas.

2) UPP Seg binaria.ipynb

Por lo mencionado arriba, me basé en otro Dataset, analizado en el siguiente Paper:

<https://www.nature.com/articles/s41598-020-78799-w.pdf>

La principal diferencia es que estos datos sí son muchos más, pero en este caso, las máscaras de segmentación clasifican a los píxeles de forma binaria separando lo que sería “úlcera” de lo que es el “background”. Lo cual puede ser muy útil para poder medir el tamaño de la herida entre otras cosas (en el paper aclaran bien los fines prácticos).

Entonces en este Colab, me centré en analizar algunos aspectos de este dataset.

Los datasets y el código que estudia estos datos están en github:

<https://github.com/uwm-bigdata/wound-segmentation/tree/master/data>

Por ahora usé los que están dentro de la carpeta wound\_dataset, en el .zip [azh\_wound\_care\_center\_dataset\_patches.zip](https://github.com/uwm-bigdata/wound-segmentation/blob/master/data/wound_dataset/azh_wound_care_center_dataset_patches.zip)

También sería interesante usar los que aparecen en la carpeta [Foot Ulcer Segmentation Challenge](https://github.com/uwm-bigdata/wound-segmentation/tree/master/data/Foot%20Ulcer%20Segmentation%20Challenge).

3) UPP Seg Bin - PreProcesamiento .ipynb

Este Colab es bastante parecido al anterior. El Colab 2) fue más que nada para ver cómo eran los datos y probar cosas (más como un borrador), pero lo que sí necesité usar del 2do colab está en este 3er Colab (y en parte, en el 4to también) , de forma mucho más ordenada y con otras cosas agregadas que menciono después, por lo que el 2do Colab podría borrarlo quizás (lo dejo por ahora, por si tengo algo que me puede llegar a servir a futuro.).

El 3er Colab se centra específicamente en el pre-procesamiento de estas imágenes binarias, principalmente buscando crear una carpeta local en mi Drive que contenga las máscaras en formato .npy con un solo canal (HxWx1) que tiene los valores 0 o 1, en lugar del RGB con el que venían las .png . Estos archivos .npy son los que voy a usar después en el entrenamiento.

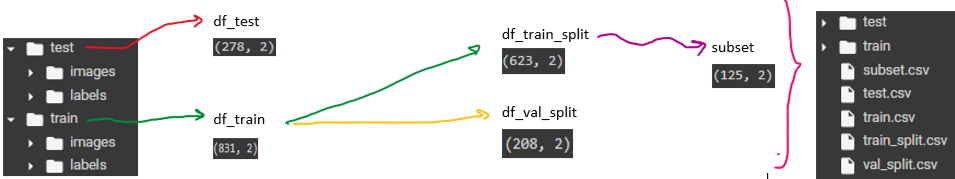
La idea de tenerlo ya en el Drive es poder después abrirlo y usarlo en el 4to Colab, más que nada porque pasar las imágenes de una forma a la otra tarda bastante en procesar (seguramente hay formas mucho más óptimas de hacerlo que la que usé yo y que tardan mucho menos) y con guardarlas en Drive me evito tener que cargarlas todo el tiempo y puedo acceder a ellas desde distintos Colabs.

4) UPP Seg Bin - Baseline.ipynb

En esta última parte creo datasets con pandas con los nombres de los archivos (de las imágenes y de las labels en .npy) para poder trabajar de forma más ordenada, pudiendo acceder luego a partir de sus paths a los archivos originales, dividirlos más fácilmente en train, test y validation y crear subsets para hacer pruebas de forma más rápida.

Luego creo un modelo, lo corro y evalúo algunas métricas, más que nada para tener un baseline como punto de partida (no sé aún cuál sería el modelo más óptimo).

Los datos los dividí de la siguiente manera:



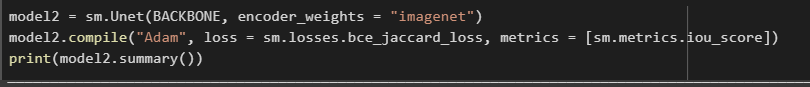
A partir de estos .csv creados, puedo acceder a las imágenes y etiquetas fácilmente según la distribución realizada (train, test, val, subset), asegurándome por otro lado que las imágenes coincidan con sus etiquetas al tener los mismos nombres.

El modelo empleado es una Unet, con backbone de resnet34, usando para el encoder los pesos obtenidos al entrenarse con la base de datos de imagenet (uso transfer learning).

Función de costo: Binary cross entropy (Jaccard)

Optimizador : Adam.

Métrica: IoU



Por el momento, el mejor val\_iou\_score obtenido fue de 0.8524. (entrenando con las 623 de df\_train\_split y validando con df\_val\_split):



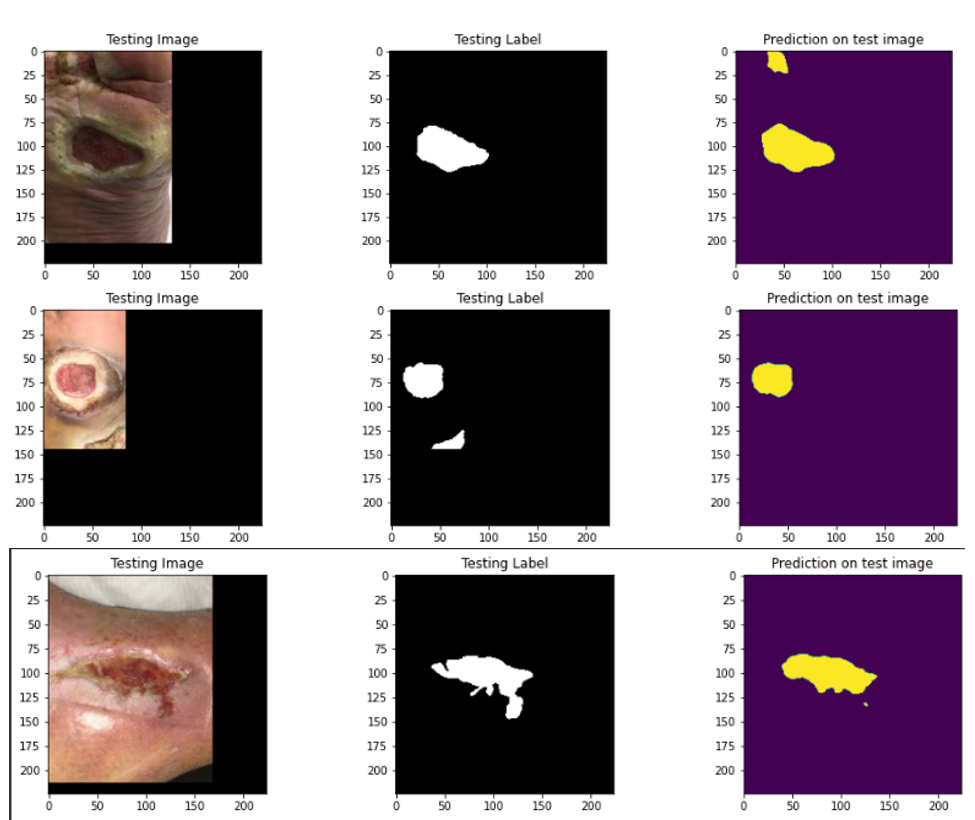
Sin embargo, hice muy pocas pruebas.

**Para seguir:**

- Ver que batchsize usar, como preprocesar las imágenes antes de meterlas al modelo (según el preprocesamiento específico que espere el modelo (normalizar, reshape, etc))

- Quizás pueda simplificar mucho el código de preprocesamiento, no hace falta que maneje todo por los índices iguales entre image-label, si después cuando cargo las imágenes en 4), creo los dataframes con los paths y ahí ya hago que coincidan. También puedo optimizar la función que pasa de RGB a 1 canal binario, a partir del link que dejé donde explicaban como hacerlo más fácil creo.

-Ver gráficamente los resultados obtenidos y chequear cómo dan las predicciones.



-Ver como reflejar esto en wandb

<https://wandb.ai/piahiba-estudiantes/ulceras-por-presion/runs/269titc4?workspace=user-pedromrisky>

-Entrenar con más datos y ver de testear con datos distintos también (puedo agarrar el S1 Dataset y unir los tejidos para tener la úlcera segmentada y usar eso para testear y ver cómo generaliza el modelo a estas úlceras que creo que son más variadas.).

- Ver de otras formas de acceder a los datos para clasificar tejidos

-Ordenar colabs (unificarlos ?), optimizar preprocesamiento (seguro hay formas más rápidas y directas de hacer lo que hice). Probar nuevos modelos, fine-tuning, regularization, data augmentation, etc.

5) UPP Versión principal.ipynb

Unifiqué todos los Colabs en uno solo donde tenga todo lo importante, sin depender de otros Colabs y simplifiqué bastante el código.

Primero creo los dataframes con strings de nombres y etiquetas de imágenes para hacer los splits. A partir de esos df creo variables tipo numpy con las imágenes y etiquetas en .npy y las proceso para que me queden de las dimensiones que necesito y en float32 en lugar de unit8. Creo un tensor a partir de esas variables que me sirva para alimentar el modelo que elija para entrenar.

6) UPP Paper Seg Bin

- Usé partes del código de github del paper de segmentación binaria para cargar un modelo que ellos proponen (DeeplabV3 con backbone de MobileNetV2), para subir métricas que usan y para crear una clase generadora de datos (clase Datagen) que después puedo instanciar para los fit y que además tiene otras funciones que me resultaron útiles.

-Agregué un dataset nuevo, ahora tengo 2: el de "Foot Ulcer" y el de "Azh" y los preprocesé para que queden de la forma que espera recibirlos del data generator. Todavía me quedaría agregar un dataset más de los que figuran en ese github que son 3. También unifiqué los datasets para poder entrenar con todo junto, con lo cual puedo elegir si quiero entrenar con Azh, Foot Ulcer o con ambos unificados, según el path que le pase al datagenerator.

-Mejoré bastante la parte de Wandb (antes estaba bastante mal y desorganizada), poniendo las partes necesarias para cargar los configs y callbacks de cada entrenamiento (después seguro tenga que agregar muchos hiperparámetros más que no agregué y más aún al ir queriendo hacer pruebas nuevas)

-Ordené todo por solapas, entre ellas: Análisis, Preprocesamiento, Entrenamiento, Evaluación , Visualización , etc.

-Agregué la opción de guardar los modelos una vez entrenados (en hdf5) y de poder cargarlos después y también guarda un plot y un json con métricas. Esta parte por ahí teniendo WandB no sé si hace falta, tengo entendido que se pueden guardar los modelos con WandB ?), me iba a fijar después

-Mejoré un poco la parte de evaluación y de visualización de resultados o las predictions

Para las pruebas de los distintos modelos puedo cargar el del paper que mencioné (DeeplabV3) con distintos hiperparámetros, también los modelos de la librería segmentation\_models que permite cargar 4: Unet, Linknet, FPN y PSPNet, con un montón de backbones (VGG, ResNet, MobileNet, EfficientNet y muchas otras ), o modelos de Keras.Applications.

Lo que me pasaba últimamente es que no llegaba a entrenar mucho sin que se me desconecte de la GPU de Colab o que me ande bastante lento. Podría probar hacer las pruebas con subsets de los datos y menores batch size.