## Estructura de la organización en el servidor

Para conectarnos al servidor lo hacemos a través de jupyter, con el que podemos escoger distintos paneles, según qué tarea queramos realizar (etiquetaje / trabajo general).

En él tenemos un directorio con las imágenes */home/jovyan/images/* (el cual llamaremos directorio images) y otro directorio de trabajo /*home/jovyan/work/* (el cual llamaremos directorio work) que será donde podremos crear, modificar y ejecutar nuestros archivos.

En el directorio *image* encontramos que hay un archivo que continene los metadatos, llamado CRCpath.csv, y las 1331 imágenes que nos han facilitado distribuidas en diferentes carpetas:

* 220G N → Normal: Donde hay imágenes con tejido sano
* 1.3T T → Tumor: Donde hay imágenes con tejido tumoral
* 727G TN → TumorNormal: Donde hay imágenes con tejido tumoral y sano
* 4.8G U → Undefined: Donde hay imágenes que no están definidas

Tenemos que tener en cuenta que el peso del dataset (imágenes en formato TIF) és de 2.2T

El 15 de abril el Dr. Víctor Moreno nos comentó que llegaba una nueva GPU, con las mismas características que la que ya había, y que la podríamos utilizar para realizar el proyecto. Estaríamos sin acceso entre 1-2 días (según como fuera la instalación).

## Acceso a los datos

Cuando tuvimos acceso a los datos, a las imágenes originales en formato TIF, vimos que había imágenes muy dispares con referente al peso. La más pequeña pesaba 1M mientras que la de mayor peso llegaba a 6.5GB. Dado a la disparidad de estos datos, el Dr. Víctor Moreno comprobó si la transferencia de los datos había ido correctamente, ya que no se había hecho ninguna comprobación, y no vió nada anómalo, todo había ido correctamente, así que las imágenes estaban como en origen.

## Extracción de las imágenes: de TIF a PNG

Una vez hemos obtenido las imágenes en formato TIF (EXPLICAR FORMATO), tenemos que ver cómo son (contienen entre 11 y 12 capas diferentes resoluciones, una capa con imagen en blanco y negro).

Les comandes estan repes per veure quin és el format que ens agrada més

|  |
| --- |
| # Code Blocks: auto → tomorrow (base) mpuiggros@uic.es:~/work/data/T\_10M$ identify /home/jovyan/images/TN/I21198.tif /home/jovyan/images/TN/I21198.tif[0] TIFF64 1251x3685 1251x3685+0+0 8-bit sRGB 2.77733GiB 0.000u 0:00.007 /home/jovyan/images/TN/I21198.tif[1] TIFF64 1251x3685 1251x3685+0+0 8-bit Grayscale Gray 0.000u 0:00.007 /home/jovyan/images/TN/I21198.tif[2] TIFF64 81920x64512 81920x64512+0+0 8-bit sRGB 0.000u 0:00.005 /home/jovyan/images/TN/I21198.tif[3] TIFF64 40960x32256 40960x32256+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[4] TIFF64 20480x16128 20480x16128+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[5] TIFF64 10240x8064 10240x8064+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[6] TIFF64 5120x4032 5120x4032+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[7] TIFF64 2560x2016 2560x2016+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[8] TIFF64 1280x1008 1280x1008+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[9] TIFF64 640x504 640x504+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[10] TIFF64 320x252 320x252+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[11] TIFF64 160x126 160x126+0+0 8-bit sRGB 0.000u 0:00.004 |

|  |
| --- |
| *# Code Blocks: bash → atom-one-light* (base) mpuiggros@uic.es:~/work/data/T\_10M$ identify /home/jovyan/images/TN/I21198.tif /home/jovyan/images/TN/I21198.tif[0] TIFF64 1251x3685 1251x3685+0+0 8-bit sRGB 2.77733GiB 0.000u 0:00.007 /home/jovyan/images/TN/I21198.tif[1] TIFF64 1251x3685 1251x3685+0+0 8-bit Grayscale Gray 0.000u 0:00.007 /home/jovyan/images/TN/I21198.tif[2] TIFF64 81920x64512 81920x64512+0+0 8-bit sRGB 0.000u 0:00.005 /home/jovyan/images/TN/I21198.tif[3] TIFF64 40960x32256 40960x32256+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[4] TIFF64 20480x16128 20480x16128+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[5] TIFF64 10240x8064 10240x8064+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[6] TIFF64 5120x4032 5120x4032+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[7] TIFF64 2560x2016 2560x2016+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[8] TIFF64 1280x1008 1280x1008+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[9] TIFF64 640x504 640x504+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[10] TIFF64 320x252 320x252+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[11] TIFF64 160x126 160x126+0+0 8-bit sRGB 0.000u 0:00.004 |

|  |
| --- |
| # Code Blocks: auto → atom-one-light (base) mpuiggros@uic.es:~/work/data/T\_10M$ identify /home/jovyan/images/TN/I21198.tif /home/jovyan/images/TN/I21198.tif[0] TIFF64 1251x3685 1251x3685+0+0 8-bit sRGB 2.77733GiB 0.000u 0:00.007 /home/jovyan/images/TN/I21198.tif[1] TIFF64 1251x3685 1251x3685+0+0 8-bit Grayscale Gray 0.000u 0:00.007 /home/jovyan/images/TN/I21198.tif[2] TIFF64 81920x64512 81920x64512+0+0 8-bit sRGB 0.000u 0:00.005 /home/jovyan/images/TN/I21198.tif[3] TIFF64 40960x32256 40960x32256+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[4] TIFF64 20480x16128 20480x16128+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[5] TIFF64 10240x8064 10240x8064+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[6] TIFF64 5120x4032 5120x4032+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[7] TIFF64 2560x2016 2560x2016+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[8] TIFF64 1280x1008 1280x1008+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[9] TIFF64 640x504 640x504+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[10] TIFF64 320x252 320x252+0+0 8-bit sRGB 0.000u 0:00.004 /home/jovyan/images/TN/I21198.tif[11] TIFF64 160x126 160x126+0+0 8-bit sRGB 0.000u 0:00.004 |

Al tener un conjunto de resoluciones de la misma imagen tenemos que escoger con qué capa vamos a trabajar ⇒ ~ 10M, un peso con el que se puede trabajar para hacer el modelo. Para poder ver cual de las resoluciones se ajusta a esta medida se extraen algunas capas de un par de imágenes aleatoriamente escogidas y se identifica cual es la resolución aproximada que se adapta al peso escogido.

Cuando sabemos la resolución tenemos que llevar a cabo la extracción de una capa para cada imagen y tenemos que tener en cuenta que extraer una capa es costoso, son alrededor de 6 segundos. Con lo que al tener 1328 imágenes, extraer una capa por cada una de ellas, necesitaríamos alrededor de 1328\*6 segundos = 7968 segundos ⇒ 2,5 horas.

|  |
| --- |
| # Code Block: bash → atom-one-dark  time -p convert /home/jovyan/images/TN/I21198.tif[6] I21198\_6.png real 6.65 user 5.14 sys 0.27 |

|  |
| --- |
| *# Code Block: bash → atom-one-light* time -p convert /home/jovyan/images/TN/I21198.tif[6] I21198\_6.png real 6.65 user 5.14 sys 0.27 |

|  |
| --- |
| *# Code Block: auto → atom-one-dark* time -p convert /home/jovyan/images/TN/I21198.tif[6] I21198\_6.png real 6.65 user 5.14 sys 0.27 |

Como hacerlo manualmente llevaría más de 2 horas y media, ya que el cálculo está hecho con el tiempo riguroso de la extracción, no tiene en cuenta el tiempo de búsqueda del nombre de la imagen ni la preparación del comando, decidimos hacer un script para facilitar el trabajo y agilizarlo.

El script llamado *extract\_img.sh*, a parte de automatizar la extracción de una capa (la escogida según la resolución adecuada en cada imagen), usa la paralelización de ejecuciones para reducir el tiempo de ejecución. Así que para ejecutar el script se le tienen que pasar 2 parámetros:

* Parámetro 1: El nombre, sin añadir la ruta, del directorio donde están las imágenes en formato tif.
* Parámetro 2: El número de procesadores deseados para poder paralelizar el proceso.

Con el script conseguimos reducir el tiempo de ejecución notablemente. Para la extracción de 362 imágenes (las correspondientes a la carpeta TN), si ejecutamos el script con 10 CPUs conseguimos extraer las imágenes en 225 segundos (es decir 3,75 minutos), pero en cambio si lo ejecutamos con 20 CPUs lo reducimos a 172 segundos (2,9 minutos). A continuación mostramos un ejemplo con dos ejecuciones, la primera paralelizando con 10 CPUs y la segunda con 20 CPUs, las dos sobre la carpeta TN que, como ya hemos comentado, contiene 362 imágenes.

|  |
| --- |
| *# CON 10 threads* mpuiggros@uic.es:~/work/data$ time -p ./extract\_img.sh TN 10 &> log\_TN real 225.98 user 906.86 sys 47.33  *# CON 20 threads* mpuiggros@uic.es:~/work/data$ time -p ./extract\_img.sh TN 20 &> log\_TN real 172.37 user 907.55 sys 46.99 |

Para almacenar el output, el programa crea un directorio con el nombre “parametro1\_\_10M”, donde se guardarán las imágenes en PNG (la capa seleccionada) de cada imagen TIF, y dichas imágenes conservan el nombre original del archivo TIF añadiéndole el número de la capa correspondiente (ejemplo: I21198\_6.png)

**PASO 2, verificar resolución**

Una vez extraídas las capas según la resolución escogida, tenemos que verificar que la imagen de la resolución escogida tiene un peso mínimo de 10M. En caso contrario, tenemos que coger la capa más cercana que cumpla el requisito.

Para llevar a cabo este proceso, también se ha realizado un script, llamado *extract\_img\_next\_layer.sh*, donde pasandole el listado de las imágenes que se tienen que corregir, coge la siguiente resolución de la imagen original TIF para poder extraer una imagen en PNG con un peso mayor o igual a 10M. Este proceso, también paralelizado del mismo modo que el anterior, lo repetimos las veces que sea necesario hasta llegar a obtener todas las imágenes con el peso deseado.

**Tecnología usada**

Todo este proceso lo hemos llevado a cabo utilizando el lenguaje de programación bash, ya que es el lenguaje más directo para poder tratar todo lo referente a la información que atañe al sistema operativo, como es el peso de las imágenes, el listado de los ficheros, a la conversión de formato, etc.

Como ya hemos comentado anteriormente, el nombre de los scripts utilizados se muestran a continuación, así como la manera en que se ejecutan [el código de los scripts se puede ver adjunto a la memoria].

1. extract\_img.sh

* ./extract\_img.sh [N/TN/T] [#CPUs] &> log\_file

1. extract\_img\_next\_layer.sh

* ./extract\_img\_next\_layer.sh dir/list\_name\_file [#CPUs]