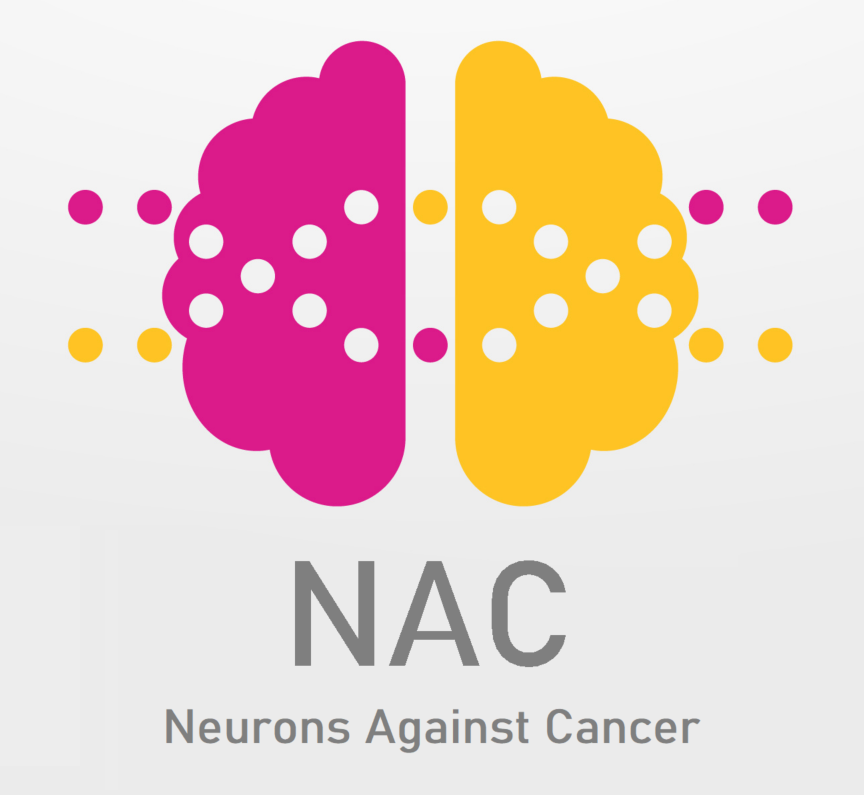
**Denominación del Trabajo de**

**Fin de Máster**



Master in Big Data Science, 2020-21

**Contenido**

[**Índice de tablas y figuras**](#_cyyupuxhk91a) **4**

[Introducción](#_30j0zll) **6**

[**Grupo de Investigación**](#_6qdae0tq4006) **6**

[**Descripción del problema**](#_d6iodr303ic) **6**

[**Introducción al Deep Learning**](#_98ij027itrw8) **6**

[**Estado del arte**](#_ccsz35kgadep) **6**

[**Arquitectura**](#_j3govxbbscly) **6**

[**Dataset**](#_4xxmfowhfxtt) **6**

[**Ubicación y estructura de los datos en el servidor**](#_ejr9tkvmttl4) **6**

[**Descripción y tratamiento del dataset**](#_g8wzucd7m6ne) **7**

[**Extracción de las imágenes: de TIF a PNG**](#_6is0dqt6br1y) **7**

[**EDA**](#_hn0w2ekhppbb) **10**

[**Tratamiento estadístico de las imágenes**](#_ocetnlnyoo0t) **10**

[**Estratificación**](#_wrljkeasl8q4) **10**

[**Aplicación de Deep Learning al problema**](#_8to74hgoot1d) **10**

[**Clasificador**](#_yqilh9i28e2w) **10**

[**Modelo 1 - Paper**](#_4z2zrjbka39a) **10**

[**Modelo 2 - MobileNet**](#_4z2zrjbka39a) **10**

[**Modelo 3 - Inception**](#_4z2zrjbka39a) **10**

[**Comparativa de los modelos**](#_4z2zrjbka39a) **10**

[**Localizador**](#_g5jlx7t2hi31) **11**

[**Modelo 4 - Localización**](#_gw4w0vkysqd6) **11**

[**Conclusiones**](#_604cbhnpuehe) **11**

[**Propuesta de valor**](#_837t1quz3ur) **11**

[**Posibles mejoras**](#_7faw6vxdzltm) **11**

[**Aplicaciones futuras**](#_7faw6vxdzltm) **11**

[**Ética**](#_9zchjrvtlbht) **11**

[**Protección de datos**](#_xia8s87ao69h) **11**

[**“Valores humanos”**](#_xia8s87ao69h) **11**

[**Planificación del proyecto ???**](#_4vi0o1u4jhnj) **11**

[**Bibliografía**](#_4vi0o1u4jhnj) **11**

[**Anexos**](#_4vi0o1u4jhnj) **11**

# Índice de tablas y figuras

**Índice de tablas**

XXX

**Índice de figuras**

XXX

**Agradecimientos**

# Introducción

## Grupo de Investigación

## Descripción del problema

## Introducción al Deep Learning

## Estado del arte

## 

# Arquitectura

# Dataset

## Ubicación y estructura de los datos en el servidor

Como se ha comentado en la sección “Arquitectura”, para acceder a los datos es necesario conectarse al servidor facilitado y este acceso será mediante un docker específico, dependiendo de la tarea que se quiera realizar. En este caso, al querer tratar los archivos del dataset, se accede mediante el docker “*GPU*”. Este docker realiza la conexión a través de Jupyter, con el que podemos realizar distintas tareas a través de la terminal o del Jupyter Notebook.

En el servidor se puede encontrar un directorio con las imágenes */home/jovyan/images/* (en adelante *images*) y un directorio de trabajo /*home/jovyan/work/* (en adelante *work*) que será donde se podrán crear, modificar y ejecutar los archivos.

En el directorio *images* se encuentra un archivo que contiene los metadatos, llamado CRCpath.csv, y las 1331 imágenes facilitadas para el estudio y distribuidas en diferentes carpetas, mostradas en la siguiente tabla:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Num | % | Peso |
| T | **702** | **52** | **1.3T** |
| N | **264** | **20** | **220G** |
| TN | **362** | **27** | **727G** |
| U | **3** | **0** | **4.8G** |
| **Total** | **1,331** | **100** | **2.2T** |

Donde:

* N es el directorio Normal que contiene imágenes con tejido sano.
* T es el directorio Tumor que contiene imágenes con tejido tumoral.
* TN es el directorio TumorNormal que contiene imágenes con tejido tumoral y sano.
* U es el directorio Undefined que contiene las imágenes que no están definidas.

## Descripción y tratamiento del dataset

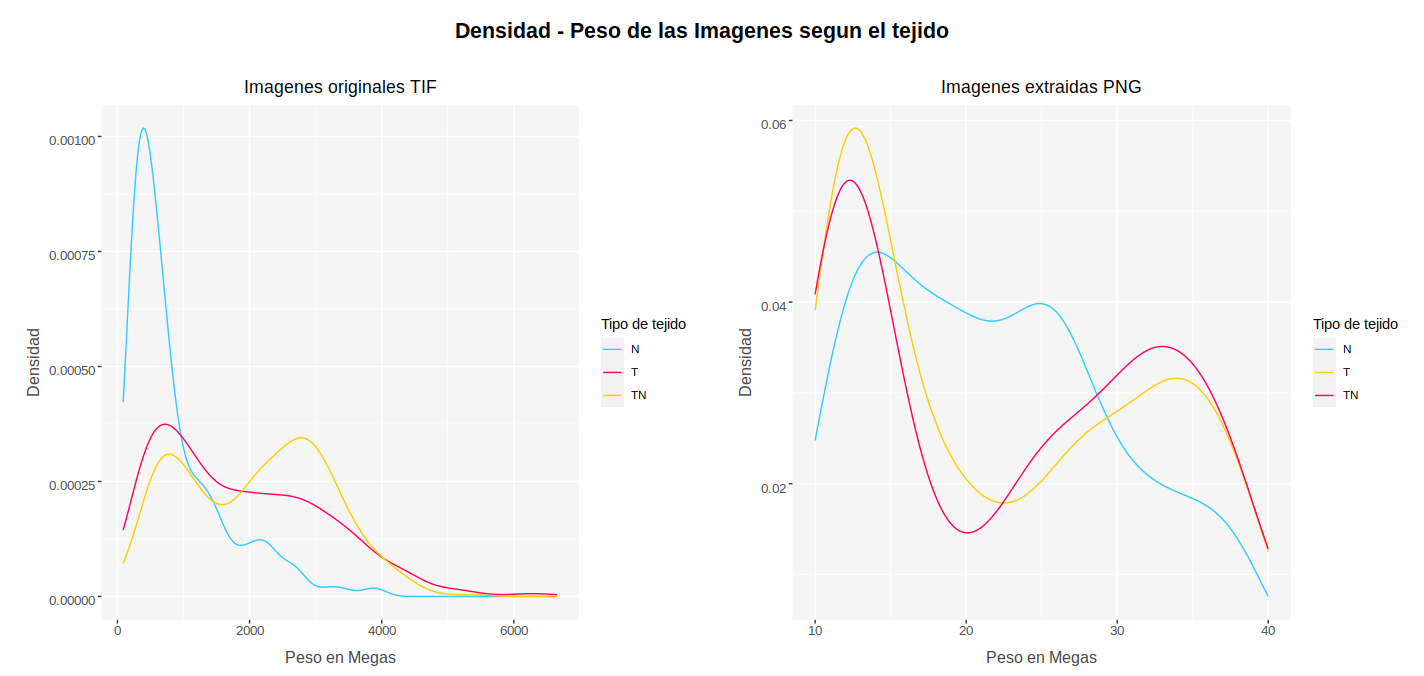
Como ya se ha comentado anteriormente, el dataset consta de 1331 imágenes, las cuales están en formato TIF (Tagged Image File Format) (EXPLICAR FORMATO). Las imágenes TIF de este dataset constan de 11 capas, cada una de ellas con una resolución distinta. A continuación se muestra un ejemplo de cómo se ven las capas en una imagen TIF [**Figura X**]:

# 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

**Figura X:** Información de cada capa del formato TIF

El peso total del dataset es de 2.2TB de memoria, lo que implica que se debe tener en cuenta que se está tratando con Big Data en el momento de procesar los datos.

Tras una primera valoración de los datos, estudiando el peso de la imágenes, se vió que el peso de las imágenes fluctuaba entre 1MB, la imagen con menor peso, a 6.5GB, la imagen con un peso mayor [Figura X]. Es decir, que había una gran variabilidad del peso de las imágenes teniendo en cuenta que era un conjunto de imágenes similares con el mismo tipo de información. Tras esta observación, se comprobó que la transmisión de las imágenes fue realizada correctamente, no hubo ningún problema referente a la transmisión de datos.



**Figura X:** Densidad - Peso de las imágenes según el tejido

Una vez comprobado que los archivos no estaban corruptos, se inició el proceso de extracción de las capas. Primero se determinó que la imagen extraída debería tener un peso mínimo de 10MB (peso mínimo óptimo para poder realizar el modelo), con lo que se hicieron un conjunto de extracciones de diferentes resoluciones (capas) en diferentes imágenes escogidas aleatoriamente para determinar la resolución que más se aproximaba a un peso de 10MB.

Una vez determinada la resolución, se debe llevar a cabo la extracción de una capa para cada imagen. Se debe tener en cuenta que estamos trabajando con un gran volumen de datos (Big Data), así que cualquier operación que sea necesaria se tendrá que ver el coste que conlleva para poder adecuar y organizar el trabajo.

Considerando que la extracción de una capa en una imagen son alrededor de 6 segundos [**Figura X]**:

|  |
| --- |
| *# Conversión de una capa a formato PNG* time -p convert /home/jovyan/images/TN/I21198.tif[6] I21198\_6.png real 6.65 user 5.14 sys 0.27 |

**Figura X:** Extracción de una capa con el tiempo transcurrido

calculamos el tiempo total que conllevaría la extracción de una capa para todas las imágenes:

1331 \* 6 = 7986 segundos = 2.2 horas → formato fórmula

Es decir, si se mira estrictamente el tiempo de ejecución, se tardarían alrededor de 2.2 horas en extraer una capa de cada imagen. De modo que hacerlo manualmente, contando el tiempo de búsqueda del nombre de la imagen y la preparación del comando, conllevaría alrededor de unas 3 horas. De modo que para facilitar el trabajo se decidió crear un script y agilizar así el proceso.

PER AQUI ESTIC --------------------------

El script llamado *extract\_img.sh* (Anexo), 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 en el Anexo].

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]

## EDA

# Tratamiento estadístico de las imágenes

## Estratificación

# Aplicación de Deep Learning al problema

## Clasificador

### Modelo 1 - Paper

### Modelo 2 - MobileNet

### Modelo 3 - Inception

### Comparativa de los modelos

## Localizador

### Modelo 4 - Localización

# Conclusiones

# Propuesta de valor

## Posibles mejoras

## Aplicaciones futuras

# Ética

## Protección de datos

## “Valores humanos”

# Planificación del proyecto ???

# 

# Bibliografía

# Anexos