UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

# Clasificación de imágenes dermatoscópicas utilizando Redes Neuronales Convolucionales e información de metadatos

Tesis de final de grado

entregada a la

Escuela Técnica de Ingeniería de Telecomunicaciones de Barcelona

Universidad Politécnica de Catalunya

por

**Teresa Domènech Abelló**

En parcial cumplimiento

de los requisitos del grado en

Ingeniería de Telecomunicaciones

en el

*Grupo de procesado de imagen y vídeo*

*Departamento de teoría de la señal y comunicación*

**Supervisores: Verónica Vilaplana y Marc Combalia**

Barcelona, Junio 2019

## Abstract

Convolutional Neural Networks (CNNs) is a technology that is evolving very quickly. The CNNs are used in image recognition to improve computer vision systems in image classification, object detection and segmentation tasks. For all the mentioned features we can apply the CNN in the field of medicine to automate the techniques of obtaining diagnosis and facilitate the tasks of doctors.

The purpose of this project is to classify images of different skin lesions in order to detect possible diseases such as skin cancer (melanoma) through CNNs and metadata information. The images used in this project have been provided by the Dermatology Department of the Clinical Hospital of Barcelona.

## Resum

Les xarxes neuronals convolucionals (CNN) són una tecnologia que està evolucionant molt ràpidament. Les CNN s'utilitzen en el reconeixement d'imatge per millorar els sistemes de visió per computador en tasques de classificació d'imatges, detecció d'objectes i de segmentació. Per totes les característiques mencionades podem aplicar les CNN en l'àmbit de la medicina per automatitzar les tècniques d'obtenció de diagnòstic i d'aquesta manera facilitar les tasques dels metges.

L'objectiu d'aquest projecte és classificar imatges de diferents lesions cutànies per detectar possibles malalties com el càncer de pell (melanoma) a través de CNN i informació de metadades. Les imatges utilitzades en aquest projecte han estat facilitades pel Departament de Dermatologia de l'Hospital Clínic de Barcelona.

## Resumen

Las redes neuronales convolucionales (CNN) son una tecnología que está evolucionando muy rápidamente. Las CNN se utilizan en el reconocimiento de imagen para mejorar los sistemas de visión por computador en tareas de clasificación de imágenes, detección de objetos y de segmentación. Para todas las características mencionadas podemos aplicar las CCN en el ámbito de la medicina para automatizar las técnicas de obtención de diagnóstico y de esta forma facilitar las tareas de los médicos.

El objetivo de este proyecto es clasificar imágenes de diferentes lesiones cutáneas para detectar posibles enfermedades como el cáncer de piel (melanoma) a través de CNN e información de metadatos. Las imágenes utilizadas en este proyecto han sido facilitadas por el Departamento de Dermatología del Hospital Clínico de Barcelona.

## Agradecimientos

Me gustaría expresar mi sincera gratitud a distintas personas. Primero a los dos supervisores de este proyecto, Verónica Vilaplana y Marc Combalia, por sus consejos y su paciencia durante todo el proyecto y por darme la oportunidad de introducirme en el mundo del deep learning y poder aplicar esta tecnología a casos reales, como lo es el campo de la medicina. También quiero agradecer al Hospital Clínic de Barcelona la oportunidad de trabajar en la creación de una base de datos de imágenes de estas características, y de proporcionármela para el proyecto. Finalmente también quiero agradecer a mis amigas y amigos por estos 4 años compartidos de carrera, y también a mi familia por darme soporte en este proceso de aprendizaje.

# Historial de revisión y aprobación del documento

| Revision | Date | Purpose |
|---|---|---|
| 0 | 21/04/2019 | Document creation |
| 1 | 22/06/2019 | Document revision |
| 2 | 24/06/2019 | Document modification |
| 3 | 25/06/2019 | Document revision |
| | | |

DOCUMENT DISTRIBUTION LIST

| Name | e-mail |
|---|---|
| Teresa Domènech Abelló | teredome8@gmail.com |
| Veronica Vilaplana | veronica.vilaplana@upc.edu |
| Marc Combalia | marccombalia@gmail.com |
| | |
| | |
| | |

| Written by: | | Reviewed and approved by: | |
|---|---|---|---|
| Date | 21/04/2019 | Date | 25/06/2019 |
| Name | Teresa Domènech Abelló | Name | Veronica Vilaplana y Marc Combalia |
| Position | Project Author | Position | Project Supervisors |

# Tabla de contenidos

# Lista de Figuras:

# Lista de Tablas:

# 1. Introducción

El cáncer de piel se describe cómo el crecimiento descontrolado de células anómalas de la piel. Es un cáncer frecuente en la sociedad española que está creciendo exponencialmente debido a la exposición de la gente al sol. La incidencia en España ajustada por 100.000 habitantes es de 9,7 por cada 100.000 personas según la Sociedad Española de Oncología Médica (SEOM) [1], y en Estados Unidos 22,2 por cada 100.000 personas según el Instituto Nacional del Cáncer (NIH) [2].

## 1.1. Declaración de intenciones

Existen distintos tipos de cáncer de piel. Estos pueden clasificarse en tres grandes grupos asociados a tres células distintas como se puede ver en la Fig 1:

- Carcinoma de células basales: Es el tipo más común de cáncer de piel. Alrededor de ocho de cada diez casos de cáncer de piel son carcinomas de células basales. Surgen en las partes más expuestas al sol como cabeza o cuello y crecen lentamente.

- Carcinoma de células escamosas: Alrededor de dos de cada diez casos de cáncer de piel son carcinomas de células escamosas. También surgen en las partes más expuestas al sol como las manos, cara, orejas o labios.

- Melanoma: Se origina a partir de los melanocitos, es mucho menos común que los cánceres de piel de células basales o de células escamosas, pero es más propenso a crecer y propagarse si no se trata.

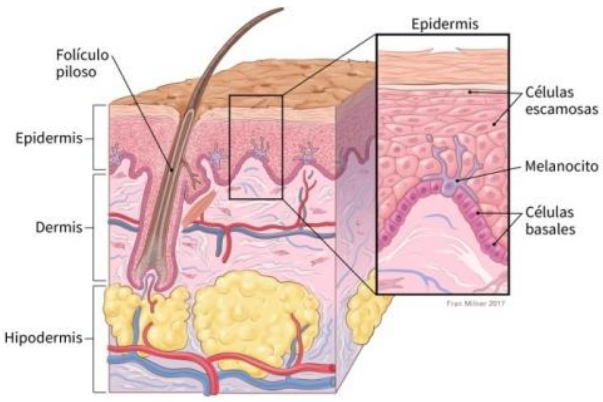



*Fig 1. Imagen de las distintas células de la piel.*

A parte de estos tres tipos de cáncer de piel existen muchos otros tipos menos comunes como: dermatofibroma, diferentes tipos de sarcomas, linfoma cutáneo, queratosis actínicas etc…

Al existir tantos tipos distintos de cáncer de piel, a la hora de hacer una valoración a simple vista puede ser complicado para un médico hacer un buen diagnóstico. Es por esto que los dermatólogos usan una técnica llamada dermatoscopia que permite examinar mejor las lesiones por debajo de la superficie cutánea amplificando in-vivo la imagen sospechosa una vez eliminados los fenómenos de refracción y reflexión de la luz.

Las imágenes dermatoscópicas son también una fuente adecuada para entrenar redes neuronales artificiales para diagnosticar automáticamente lesiones cutáneas pigmentadas y así poder dar al médico una herramienta adicional para facilitar la tarea del diagnóstico.

El propósito de este proyecto es explorar la técnica de deep learning sobre clasificación de imágenes, en concreto la Redes Neuronales Convulocionales (CNN) y obtener un sistema que sea capaz de clasificar distintas imágenes de lesiones cutáneas con una buena fiabilidad. Adicionalmente se ha usado información de metadatos para intentar obtener una clasificación más robusta. Esta clasificación utilizando información de matadatos ha sido realizada con un clasificador basado en *Gradient tree boosting*.

Para poder llevar a cabo esta clasificación se han escogido dos bases de datos de imágenes. La primera correspondiente al *ISIC 2018 Challenge* que está dividida en siete clases, y la segunda corresponde al *ISIC 2019 Challenge* que en este caso se divide en ocho clases. La segunda base de datos de imágenes mencionada ha sido proporcionada por el Departamento de Dermatología del Hospital Clínic de Barcelona y una parte de este proyecto ha sido participar en las tareas de etiquetación de la misma.

## 1.2. Técnicas remarcables

El proyecto se ha empezado a partir de una red pre entrenada con imágenes de ImageNet. El proyecto ha sido desarrollado en Python utilizando la librería Pytorch. Pytorch es una biblioteca de aprendizaje de código abierto, basada en Torch, utilizada para aplicaciones como el procesamiento de lenguaje natural o la clasificación de imágenes.

Para poder ejecutar el programa se ha necesitado el uso de una GPU demandada por la alta necesidad de capacidad computacional. Ha sido proporcionada por el Grupo de Procesado de Imagen del departamento de Teoría de la Señal y Comunicaciones de la Universitat Politècnica de Catalunya.

## 1.3. Estructura

En este capítulo se ha hecho una introducción sobre el cáncer de piel y sobre el contenido técnico de este proyecto. En el segundo capítulo se hace un análisis del estado del arte.

En el capítulo 3 se habla de la metodología y los conceptos teóricos para poder entender el apartado experimental. En el capítulo 4 se muestran y discuten los resultados obtenidos de los distintos experimentos. El presupuesto del proyecto se detalla en el capítulo 5. En el capítulo 6 se muestran las conclusiones obtenidas de analizar el capítulo 4 cómo también las posibles mejoras o implementaciones futuras. Finalmente, podemos encontrar información adicional sobre el proyecto en el apéndice así como el código fuente utilizado.

# 2. Estado del arte

Las tecnologías deep learning aplicadas al campo de la medicina han avanzado mucho a lo largo de estos últimos años. En concreto las Redes Nueronales Convulocionales (CNN) ya sea en técnicas de segmentación como de clasificación. Centrándonos en la clasificación de imágenes de lesiones de la piel nos fijamos que en los últimos años ha habido un gran avance.

El principal problema que había para aplicar técnicas de deep learning en este campo era la falta de diversidad y conjuntos de imágenes dermatoscópicas suficientemente grandes para poder entrenar una red neuronal. Por eso en 2018 *Philipp Tschandl, Cliff Rosendah y Harald Kittler* abordaron este problema creando un conjunto de datos suficientemente grande para poder abordar este tipo de problemas. Estos datos están recolectados en el HAM10000 (Human Against Machine) [3] dotado de 10015 imágenes dermatoscópicas de diferentes poblaciones adquiridas y almacenadas por diferentes modalidades. Para poder obtener las imágenes de la base de datos se emplearon distintas técnicas de adquisición y limpieza desarrolladas con sistemas semi-automáticos, utilizando especialmente redes neuronales preentrenadas.

La publicación de esta base de datos de imágenes se presentó en el *ISIC2018 Challenge.* ISIC2018 Challenge es una competición que ayuda a los participantes a desarrollar herramientas de análisis de imágenes y permitir así el diagnóstico automatizado de distintos tipos de cáncer de piel a partir de imágenes dermatoscópicas. Esta competición tiene tres tareas: la primera, segmentación de lesión, la segunda detección de atributos y la tercera, diagnóstico de tipo de lesión. Centrándonos en la tercera tarea de esta competición podemos destacar algunos de los mejores resultados en clasificación de imágenes de lesiones de la piel, que son el actual estado del arte.

Podemos destacar dos de los sistemas de clasificación con mejores resultados. *Jordan Yap* [4] probó varios tipos de red. Aparte de la base de datos de imágenes HAM10000 utilizó otras 33.644 imágenes propias para el entrenamiento. Primero utilizó pesos balanceados con la fórmula mencionada en [4] ya que por cada clase él número de imágenes era desequilibrado. Adicionalmente uso técnicas de data augmentation para aplicar transformaciones como rotaciones y giros a las imágenes y así obtener más muestras. Para mejorar la calidad de las imágenes aplicó una técnica de constancia de color. Probó distinta arquitecturas cambiando únicamente la *learning rate*, el tamaño de la última capa *fully-connected* y la media utilizada para la normalización, obteniendo los mejores resultados con las siguientes arquitecturas pre-entrenadas con imágenes de

ImageNet: *DPN-92(5k), Resnet-152, Densenet-161* e *Inceptionv3*. Con mejor resultado utilizando *DPN-92(5k)* obteniendo una tasa de acierto balanceada de 0.885. También podemos mencionar el trabajo de *Nils Gessert* [5], en este caso también se aplicó técnicas de *data augmentation* como en el caso anterior y pesos balanceados para solucionar el problema de desequilibrio entre el número de imágenes de cada clase. Los híper-parámetros que se utilizaron en este caso fueron *batch size* de 40 y *learning rate* iniciada a 0,0005 y decayendo 0,2 cada 25 épocas. Se utilizó el optimizado Adam. A diferencia del caso anterior se usó información de metadatos para mejorar los resultados. Los mejores resultados fueron obtenidos con la arquitectura DenseNet121 y SVM (Support Vector Machine) obteniendo una tasa de acierto balanceada de 0.856.

# 3. Marco teórico y Metodología

En este apartado se detallaran y comentaran los métodos usados en cada una de las tres fases en que se ha dividido el proyecto.

- Fase 1: Preparación y etiquetado de las imágenes de la base de datos del Hospital Clínic de Barcelona.
- Fase 2: Definición y entrenamiento de una Red Neuronal Convulocional (CNN).
- Fase 3: Definición y entrenamiento de un sistema de clasificación basado en árboles de decisión utilizando información de imagen y metadatos.

## 3.1. Base de datos

En este sub-apartado se comentaran las dos bases de datos de imágenes utilizadas en este proyecto. Para la base de datos correspondiente al *ISIC Challenge 2019* también se destacara la labor de preparación y etiquetado de las imágenes.

### 3.1.1. ISIC 2018

La primera base de datos de imágenes utilizada ha sido la correspondiente a la competición *ISIC Challenge 2018.* Esta base de datos se presentó en el paper HAM10000 [3]. Contiene 10015 imágenes clasificadas en siete clases diferentes: Melanoma, Melanocytic Nevus, Basal Cell Carcinoma, Actinic Keratosis / Bowen's disease, Benign Keratosis, Dermatofibroma y Vascular Lesions. A continuación en la Tabla 1 se muestra el número de imágenes por clase. Podemos ver como el número de imágenes por clase es muy desequilibrado.

| Clase | # Imágenes |
|---|---|
| Melanoma | 1.113 |
| Melanocytic Nevus | 6.705 |
| Basal Cell Carcinoma | 514 |
| Actinic Keratosis | 327 |
| Benign Keratosis | 1.099 |
| Dermatofibroma | 115 |
| Vascular Lesions | 141 |

*Tabla 1. Distribución de las imágenes de la base de datos para cada clase.*

Todas las imágenes fueron tomadas con un dermatoscopio digital system MoleMax HD.

### 3.1.2. ISIC 2019

La base de datos perteneciente a la competición *ISIC Challenge 2019* ha sido proporcionada por médicos del Departamento de Dermatología del Hospital Clínic. Esta base de datos se ha tenido que ordenar, limpiar, clasificar y etiquetar para poder adaptarla a las necesidades del *challenge*. Gran parte de esta preparación de datos ha sido realizada por los investigadores del Hospital Clínic, concretamente por uno de los supervisores del proyecto, Marc Combalia. En la fase final de esta preparación es dónde he podido participar.

La base de datos presentaba varios problemas. El primer problema era que de un mismo paciente podía haber distintas lesiones y por lo tanto había que identificar que esas lesiones pertenecían a un mismo sujeto (id_lesiones). Por otra parte había sujetos que tenían muestras de una misma lesión pero de momentos diferentes, por ejemplo, dos imágenes de la misma lesión pero con una diferencia de tiempo de dos meses en la que la lesión podía haber cambiado de forma. Este tipo de casos también hacía falta etiquetar-los de alguna forma (id_momento).

Para poder etiquetar estas clases se usaron distintas técnicas de clasificación utilizando redes pre-entrenadas. Pero la clasificación y etiquetación no era perfecta.

Para evitar posibles errores, manualmente se revisaron todas las imágenes mirando que todas las etiquetas id_momento correspondieran a la mismo lesión, es decir que para una misma lesión no hubiera dos etiquetas diferentes ya que las redes neuronales podían detectarlas como diferentes las imágenes pero que en realidad fueran dos imágenes correspondientes a la misma lesión. También manualmente se revisó que todas las imágenes de lesiones correspondientes al mismo sujeto estuvieran identificadas con la misma etiqueta id_lesiones, y que no estuvieran asignadas a dos pacientes diferentes.

Finalmente para la tarea de test del *challenge* se quería proponer utilizar distintas imágenes detectadas como casos complejos. Estos casos complejos corresponderían a imágenes dónde apareciera bolígrafo o que la imagen presentara una tonalidad más azul de lo normal. Podemos ver un ejemplo de este tipo de imágenes en las figuras: Fig 2, Fig 3 y Fig 4.

Tanto para poder utilizar todas las imágenes de tonalidad azul como todas las imágenes que contuvieran bolígrafo se utilizó el mismo método. Este método consistía en buscar manualmente, es decir, a simple vista, un conjunto de imágenes con las características mencionadas y utilizarlas como datos de entrada de un clasificador que detectara de todo el conjunto de imágenes cuales tenían las mismas características que las imágenes de entrada. Este clasificador fue proporcionada por el supervisor Marc Combalia y fue implementado con Keras. [6].

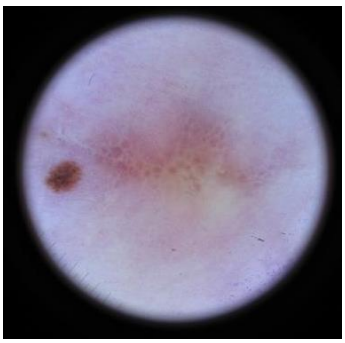


*Fig 2 Imagen con tonalidad azul.*

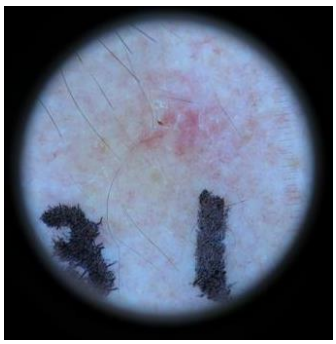


*Fig 3 Imagen con tonalidad azul y con bolígrafo.*

*Fig 4 Imagen sin ninguna alteración.*

Una vez los datos estaban preparados se obtuvo una base de datos con un total de 25.332 imágenes clasificadas en 8 clases: Melanoma, Melanocytic Nevus, Basal Cell Carcinoma, Actinic Keratosis / Bowen's disease, Benign Keratosis, Dermatofibroma, Vascular Lesions y Squamous Cell Carcinoma. Podemos ver el número de imágenes por clase en la Tabla

2. En este caso como ya pasaba en la base de datos del 2018 las clases presentan un gran desequilibrio en cuanto a número de imágenes por clase. En este caso todas las imágenes también fueron tomadas con un dermatoscopio.

| Clase | # Imágenes |
|---|---|
| Melanoma | 4.522 |
| Melanocytic Nevus | 12.875 |
| Basal Cell Carcinoma | 3.323 |
| Actinic Keratosis | 867 |
| Benign Keratosis | 2.624 |
| Dermatofibroma | 239 |
| Vascular Lesions | 253 |
| Squamous Cell Carcinoma | 628 |

*Tabla 2. Distribución de las imágenes de la base de datos para cada clase.*

## 3.2. Red Neuronal Convolucional

En este apartado se explicará en qué consiste una red neuronal convolucional y se detallará la arquitectura usada en este proyecto para realizar la clasificación.

### 3.2.1. Funcionamiento de una Red Neuronal Convolucional

Las CNN's son un clase de red neuronal profunda que se usa comúnmente para clasificar o segmentar. En este caso ha sido usada para tareas de clasificación.

Una CNN al igual que una red neuronal (NN), está formada por neuronas que tienen unos pesos y sesgos que pueden entrenarse para aprender nuevos valores. Para entenderlo mejor se comparará a continuación una neurona biológica con un modelo matemático.

La unidad computacional básica del cerebro es la neurona. Aproximadamente 86 billones de neuronas se pueden encontrar en el sistema nervioso humano y están conectadas con aproximadamente $10^{14}$ - $10^{15}$sinapsis. El siguiente diagrama Fig 5, se muestra un dibujo animado de una neurona biológica (izquierda) y un modelo matemático común (derecha). Cada neurona recibe señales de entrada de sus dendritas y produce señales de salida a lo largo de su único axón. El axón finalmente se ramifica y se conecta a través de sinapsis con las dendritas de las otras neuronas. En el modelo computacional de una neurona, las señales que viajan a lo largo de los axones (por ejemplo, x0) interactúan multiplicativamente (por ejemplo, w0·x0) con las dendritas de la otra neurona según la fuerza sináptica en esa sinapsis (por ejemplo, w0). La idea es que las fortalezas sinápticas (los pesos w) son entrenables y controlan la fuerza de influencia y su dirección: excitación (peso positivo) o inhibitoria (peso negativo) de una neurona en otra. En el modelo básico, las dendritas llevan la señal al cuerpo celular (*cell body*), donde todas se suman. Si la suma final está por encima de un cierto umbral, la neurona puede enviar un disparo, es decir, un impulso a lo largo de su axón. En el modelo computacional, asumimos que los tiempos precisos de los impulsos no importan, y que solo la frecuencia del disparo comunica la información. Basándonos en esta interpretación del código de frecuencia, modelamos la velocidad de disparo de la neurona con una función de activación f, que representa la frecuencia de los impulsos a lo largo del axón. Por ejemplo, en este proyecto se ha usado la función de activación *softmax*, que se detallará más adelante en el sub-apartado 3.2.3.

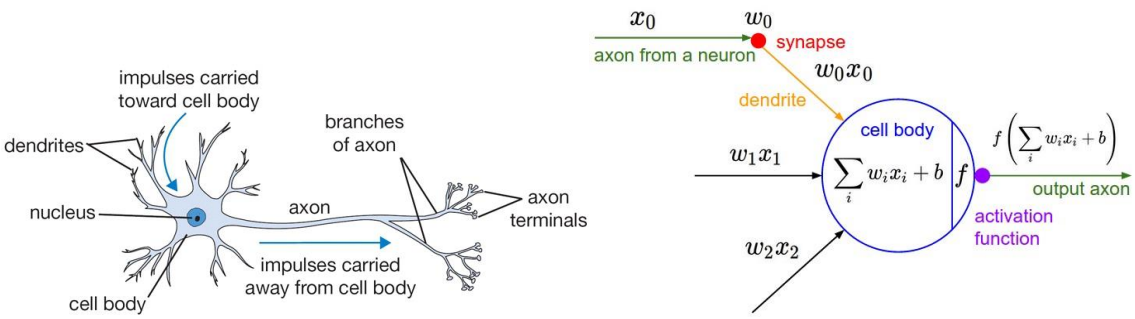



*Fig 5. Representación de una neurona biológica y su modelo matemático análogo.*

Podemos decir en conclusión, que cada neurona realiza un producto escalar entre los valores de entrada y sus pesos, agrega el sesgo y aplica la no linealidad o función de activación.

Cada conjunto de neuronas está conectado en forma de gráfico acíclico. En otras palabras, las salidas de algunas neuronas pueden convertirse en entradas de otras. Estas neuronas se organizan en distintas capas como se puede ver en la Fig 6. El paso de los distintos

valores de entrada a través de las capas obteniendo así los valores de la capa salida se denomina función de avance (*forward pass*).

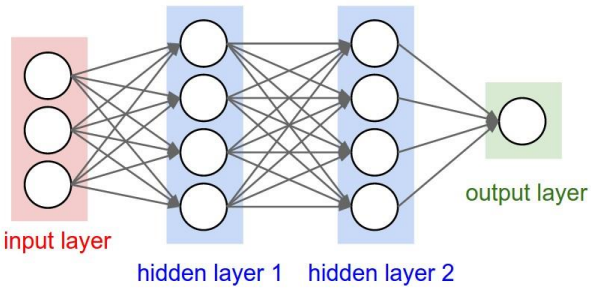



*Fig 6. NN de 3 capas con tres entradas, dos capas ocultas con cuatro neuronas por capa y una capa de salida.*

La principal característica que diferencia la CNN de la red neuronal es que suponen explícitamente que las entradas son imágenes, lo que nos permite codificar ciertas propiedades en la arquitectura. Esto hace que la función de avance (*forward pass*) [8] sea más eficiente de implementar y reduzca enormemente la cantidad de parámetros en la red. Por esta característica las CNN se adaptan perfectamente al problema planteado en este proyecto.

La CNN es una secuencia de capas, y cada capa transforma un volumen de activaciones a otra a través de una función diferenciable. Las diferentes capas de una CNN son: *convolutional layer*, *pooling layer* y *fully-connected layer*. [9] A continuación en la Fig 7 podemos ver un ejemplo de CNN con sus respectivas capas.

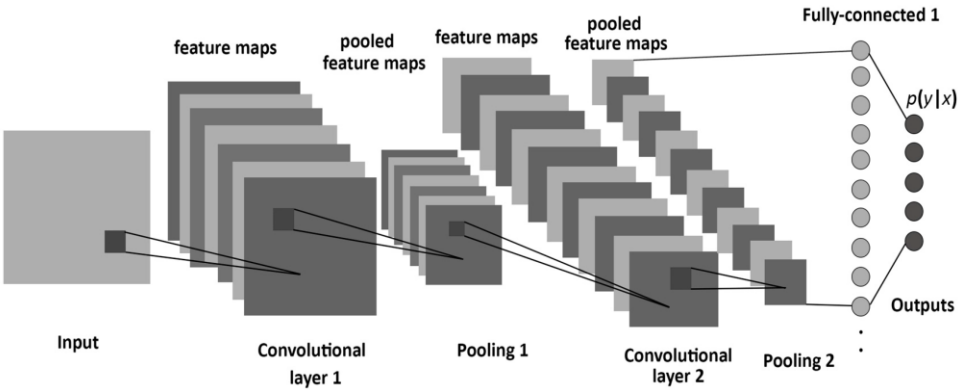



*Fig 7. Ejemplo de una CNN con sus correspondientes capas.*

### 3.2.1.1. Capas de una red neuronal convolucional

Convolutional layer

Los parámetros de la capa convolucional consisten en un conjunto de filtros entrenables. Cada filtro tiene una altura y anchura menor que los parámetros de entrada pero se extiende a través de toda la profundidad del volumen de entrada.

Por ejemplo, un filtro en la primera capa de una CNN podría tener un tamaño de 5x5x3 píxeles con una entrada de volumen 32x32x3 píxeles como podemos ver en la Fig 6.

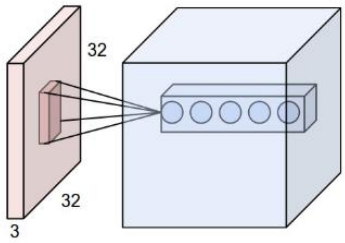



*Fig 8 Ejemplo del volumen de entrada y del volumen del filtro.*

Durante el *forward pass*, deslizamos o más precisamente, convolucionamos cada filtro a través del ancho y alto del volumen de entrada efectuando así una convolución 2D, es decir, se efectúa el producto escalar de los valores de entrada por los coeficientes del filtro. De esta forma se va creando un mapa de activación bidimensional separado para cada filtro que se van apilando a lo largo de la dimensión de profundidad, produciendo así el volumen de salida. Intuitivamente, la red aprenderá los filtros que se activan cuando ven algún tipo de característica visual, como por ejemplo un borde con alguna orientación en concreto o una mancha particular, etc.

En el caso en que las entradas sean imágenes, por lo tanto, se trataría de una entrada de alta-definición, en lugar de conectar las neuronas a todas las neuronas del volumen anterior conectaremos cada neurona solo a una región local del volumen de entrada. La extensión espacial de esta conectividad es un híper-parámetro llamado campo receptivo de la neurona, y es equivalente al tamaño del filtro. [9].

Pooling layer

Es común que haya una *pooling layer* entre las *convolutional layers*. Su función es reducir progresivamente el tamaño espacial de los mapas de activación para reducir la cantidad de parámetros y el cálculo en la red y, por lo tanto, también controlar un posible sobre-entrenamiento. La *pooling layer* opera independientemente sobre cada *depth slice* [9] del volumen de entrada, redimensionando así la altura y anchura, pero manteniendo la profundidad.

Otra ventaja de este tipo de capa es que permite a la red ser invariante frente a pequeñas variaciones, rotaciones y traslaciones de la imagen de entrada. Además, permite que la red llegue a una representación invariante de la imagen. Esto significa que puede detectar objetos dentro de una imagen, sin importar su posición.

Fully-connected layer

En la capa *fully-connected* las neuronas de la capa están conectadas con todas las activaciones de la capa anterior. Por lo tanto, sus activaciones se pueden calcular con una multiplicación de matrices seguida de un desplazamiento de sesgo.

Este tipo de capas se añaden al final de los modelos de redes convolucionales para poder decidir cómo tratar las características extraídas por las capas previas. Es decir, el conjunto de capas convolucionales y de pooling, actuarían como extractoras de características, y las capas fully-connected tomarían decisiones de cómo tratar estas características.

### 3.2.2. Arquitectura ResNet

En este sub-apartado se describe la arquitectura utilizada en este proyecto, la *Residual Neural Network Arquitecture (*ResNet*)*.

La arquitectura ResNet nace del problema de la profundidad de la red. Se podría pensar que cuantas más capas se utilizan para el entrenamiento, los resultados serán mejores, pero no es así. Cuando las redes son más profundas, es decir con un número de capas elevado y empiezan a converger, aparece un problema de degradación: cuanto más profunda es la red, la precisión (*accuracy*) se satura y luego se degrada rápidamente. Produciéndose así un aumento del error de predicción como podemos ver en la Fig 9.

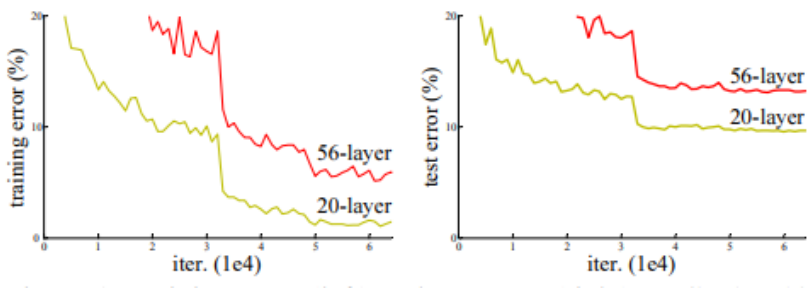



*Fig 9 Error de entrenamiento (izquierda) y error de test (derecha) en CIFAR-10. Con redes "planas" de 20 y 56 capas. La red más profunda tiene mayor error de entrenamiento.*

Para poder solucionar este problema surgió la arquitectura ResNet. Esta arquitectura se basa en dividir una red profunda en fragmentos de tres capas y pasar la entrada a cada fragmento directamente a la siguiente capa, junto con la salida residual del fragmento anterior que pasa a ser la entrada del siguiente fragmento. Este método ayuda a eliminar el problema de degradación mencionado anteriormente.

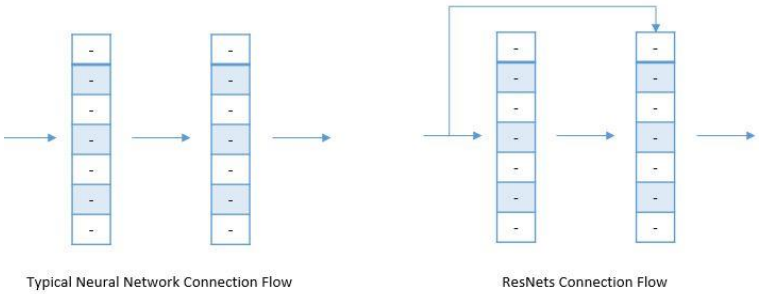



*Fig 10. ResNet vs Red neuronal básica*

En otras palabras, la arquitectura ResNet divide una red neuronal llana muy profunda en pequeños trozos de red conectados a través de conexiones de salto o acceso directo para formar una red más grande. Podemos ver un ejemplo de este tipo de bloque de reconstrucción residual en la Fig 11.

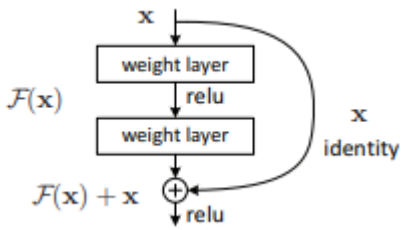



*Fig 11 Bloque reconstrucción de un aprendizaje residual.*

A continuación se detallan los dos tipos de bloques residuales que se utilizan en esta arquitectura:

Bloque identidad

El bloque de identidad es el bloque estándar utilizado en la ResNet y corresponde al caso en el que la activación de entrada tiene la misma dimensión que la activación de salida. A continuación se muestra un ejemplo de bloque de identidad donde la ruta superior es la "ruta de acceso directo" y la ruta inferior es la "ruta principal".

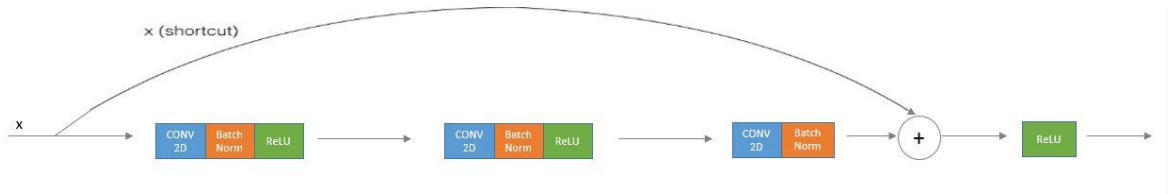



*Fig 12. Ejemplo de bloque identidad de una ResNet.*

Bloque convolucional

Este tipo de bloque se utiliza cuando las dimensiones de entrada y salida no coinciden. Lo que se hace es agregar una capa convolucional en la ruta de acceso directo.

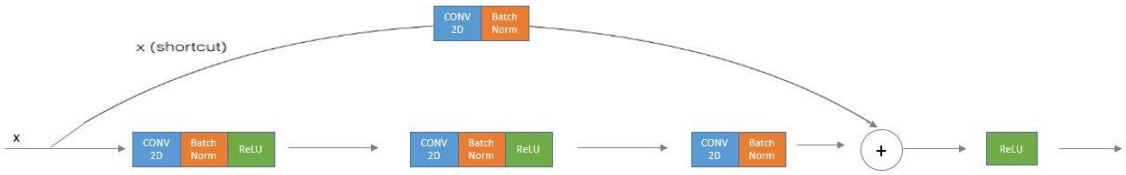



*Fig 13. Ejemplo de bloque convolucionl de una ResNet.*

En la ejecución de este proyecto se han utilizado la ResNet-18, ResNet-50 y la ResNet-101, ya que las bases de datos utilizadas contienen un número de imágenes considerable por lo que se ha ido aumentando el número de capas en los distintos experimentos para comprobar si con redes más profundas se pueden conseguir mejores resultados.

### 3.2.3. Métricas

A continuación se detallan las métricas utilizadas en los experimentos llevados a cabo en el entrenamiento de la CNN.

Función de pérdidas

La función de pérdidas, es la función a optimizar. El objetivo del entrenamiento es conseguir los híper-parámetros ideales que minimizan esta función. La función de pérdidas se evalúa en los *batches* individuales para cada *forward pass*, es decir, para cada época.

En este proyecto se ha utilizada la función *cross-entropy loss* [10]. Esta función para un caso multi-clase se puede expresar cómo:

$$L(\hat{y}, y) = -\sum_{n=1}^{N} y_n \, log(\hat{y}_n)$$

*Ecuación 1*

En esta función $N$ representa el número de clases, $\hat{y}$ y $y$ son vectores de dimensión $N$, dónde $y_n$ corresponde a los valores reales de la clase $n$ y $\hat{y}_n$ corresponde a los valores de salida de la función softmax [11] de la red para esos valores reales. La función softmax se emplea para comprimir un vector K-dimensional de valores reales arbitrarios z, en un vector K-dimensional, de valores reales en el rango [0, 1]. La función está dada por:

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

*Ecuación 2*

La *mean cross entropy* para todo *batch* se calcula como el promedio de la función de coste en cada iteración :

$$L(\hat{y}, y) = -\frac{1}{Y}\sum_{y}\sum_{n} y_n \log(\hat{y}_n)$$

*Ecuación 3*

Donde $Y$ denota las muestras de entrenamiento de cada *batch*.

Para lidiar con el problema de las clases no balanceadas se propuso utilizar *weighted loss* $L_w$ en el cálculo de la función de pérdidas.

$$L_w = \sum_{n=1}^{N} \frac{1}{\alpha_n} L_n$$

*Ecuación 4*

Dónde $L_n$ es la función de pérdidas perteneciente a la clase $n$ y $\alpha_n$ la probabilidad de aparición de una clase en concreto. Se define la probabilidad de aparición como la suma de todas las muestras de una cierta clase dividido por la suma de todas las muestras de cada una de las clases.

$$\alpha_n = \frac{\sum_i y_n^i}{\sum_{n=1}^{N}\sum_i y_n^i}$$

*Ecuación 5*

Podemos observar que invirtiendo la probabilidad de aparición de una cierta clase provocamos que se asigne un peso mayor a aquellas clases con menos muestras y viceversa.

Matriz de confusión

La matriz de confusión mencionada anteriormente es una tabla que permite evaluar si nuestro sistema de clasificación está etiquetando incorrectamente a una clase como si fuera otra. Cada fila de la tabla representa la clase verdadera y cada columna la clase predicha. La diagonal de esta matriz nos indica los casos TP. Esta métrica nos da una idea de cómo se puede mejorar un modelo. Las matrices de confusión en este documento presentarán la siguiente estructura:

| ***G \ P*** | **1** | **2** | **3** |
|---|---|---|---|
| **1** | | | |
| **2** | | | |
| **3** | | | |

*Tabla 3. Ejemplo de estructura de una matriz de confusión.*

Dónde G significa la clase a la que corresponde una imagen y P la predicción que hace la red sobre esa imagen.

Tasa de acierto balanceada

La tasa de acierto o precisión [12] es una medida que nos indica los casos en que las diferentes imágenes de entrada se han clasificado correctamente según su clase. La tasa de acierto balanceada se utiliza en esos casos en que los datos de entrenamiento presentan un número de muestras de entrada para cada clase desequilibrado, como es en este caso. La precisión balanceada se define como la precisión media obtenida para cada clase [12]. Basándonos en la matriz de confusión (descrita posteriormente), la precisión balanceada se calcula como:

$$tasa\ de\ acierto\ balaceada = \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right)$$

*Ecuación 6*

Dónde $P$ significa acierto en la predicción, $N$ error en la predicción y el significado de $TP$ y $TN$ se muestra en la Tabla 4.

| | actual | |
|---|---|---|
| | + | − |
| predicted+ | *TP* | *FP* |
| predicted− | *FN* | *TN* |

*Tabla 4. En esta tabla se muestra el significado de True positive (TP), True Negative (TN), False Positive (FP) y False Negative (FN), respecto los valores actuales y los valores predichos.*

### 3.2.4. Esquema de entrenamiento

Como hemos comentado en el apartado 3.1 las bases de datos utilizadas presentan un desequilibrio respecto al número de imágenes por clase. Por lo tanto, el primer objetivo será calcular los pesos de forma balanceada dándole más importancia a las imágenes pertenecientes a aquellas clases con un menor número de muestras.

El primer entrenamiento se realizara con un número pequeño de épocas y unos híper-parámetros inicializados a valores estándares, analizando las pérdidas y la precisión de la red, para poder obtener la época donde la red es más óptima, es decir, dónde el valor de pérdidas sea más bajo y el valor de predicción más alto. A partir de estos resultados se irán ajustando los híper-parámetros para obtener los mejores resultados posibles.

Las pruebas se harán tanto para unas muestras de entrenamiento como de validación.

## 3.3. Información de Matadatos

Para poder mejorar los resultados obtenidos con el entrenamiento de la red, se propuso utilizar la información de metadatos. Se quería comprobar si esta información nos permitiría mejorar los diagnósticos de alguna de las enfermedades ya que existen patrones que se repiten. Por ejemplo: Es más probables que una mujer entre 50 y 60 años con una lesión localizada en el torso anterior padezca melanoma que otro tipo de persona con otras características. Los metadatos utilizados en este proyecto han sido: la edad, el sexo y la localización de la lesión.

La estrategia llevada a cabo ha consistido en extraer los vectores de características de la red neuronal para cada imagen, eliminando la última capa *fully-connected*. Este vector ha sido concatenado con la información de metadatos correspondiente a cada imagen, y cada vector ha sido introducido como parámetro de entrada a un clasificador basado en el

algoritmo *gradient descent boosting trees*, que sustituye a la última capa eliminada de la red. Podemos ver esta estrategia en forma de diagrama en la Fig 14.

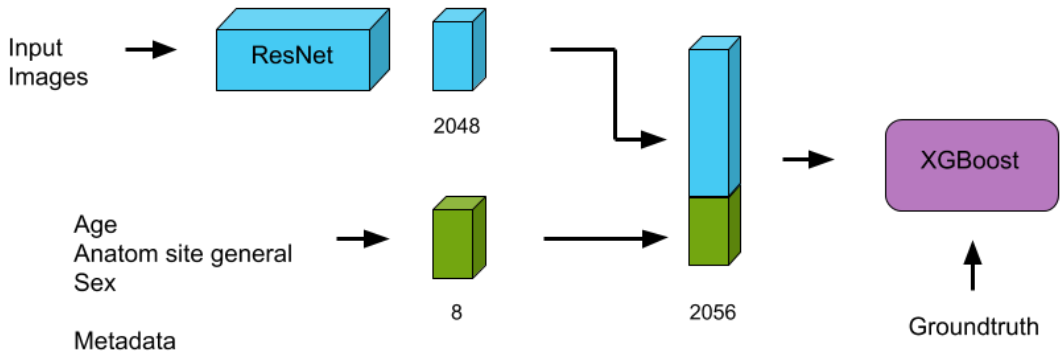



*Fig 14. Diagrama de la arquitectura de red para la clasificación utilizando información de metadatos.*

### 3.3.1. Obtención y transformación de los datos

La información de metadatos correspondiente a cada imagen se encontraba con un formato inadecuado para su uso, por lo que era necesario aplicar una transformación a cada dato para poder normalizarlo y así obtener un rango único para todos los datos, en este caso [0,1].

Los metadatos correspondientes a la edad se encontraban en un rango de 15 a 95, por lo que se normalizaron de la siguiente forma:

$$\overline{age} = \frac{age}{100}$$

*Ecuación 7*

Donde $age$ corresponde a la edad y $\overline{age}$ corresponde al valor de la edad normalizada.

La localización de la lesión podía encontrarse en el torso anterior, torso superior, torso lateral, extremidades superiores, extremidades inferiores y en la cabeza o cuello. En este caso para transformar los datos a valores dentro del rango [0,1] se propuso utilizar la codificación *one-hot-vector.* Los vectores *one-hot-vector* se definen como un grupo de valores los cuales sólo uno de ellos puede corresponder al número uno y todos los demás deben tomar el valor zero. Se ha asignado cada uno de los distintos tipos de localización a cada una de las posiciones del *one-hot-vector* de la siguiente forma:

| **Localización** | anterior_torso | upper_extremity | posterior_torso | lower_extremity | head / neck | lateral_torso |
|---|---|---|---|---|---|---|
| **Posición** | 1 | 2 | 3 | 4 | 5 | 6 |

*Tabla 5. Asignación de los distintos tipos de localización de la lesión a cada una de las posiciones del one-hot-vector.*

Finalmente para adaptar los datos correspondientes al sexo, al sólo tomar dos valores, hombre o mujer, estos se han transformados a valores binarios, zero o uno respectivamente.

A continuación, en la Tabla 6, se muestra un ejemplo de la transformación completa de los metadatos correspondientes a una imagen en concreto.

| | ***Image*** | ***Age approx*** | ***Anatomia general site*** | ***Sex*** |
|---|---|---|---|---|
| ***Metadatos*** | ISIC_0000000 | 55 | Anterior torso | female |
| ***Metadatos normalizados*** | ISIC_0000000 | 0,55 | [1,0,0,0,0,0] | 1 |

*Tabla 6. Ejemplo de una transformación completa a valores normalizados de la información de metadatos.*

Como se ha comentado para realizar esta clasificación de datos es necesario extraer el vector de características para cada imagen de la última capa de la red. Para llevarlo a cabo se ha creado una función *forward* modificada en la arquitectura ResNet. Esta función modificada es exactamente la misma que la original pero eliminando la útima capa *fully connected*. El valor que devuelve esta función son las características de la red para cada imagen de entrada.

### 3.3.2. XGBoost

XGBoost es un sistema de clasificación basado en árboles de decisión que utiliza la técnica de boosted trees. A continuación se definirá y detallará el funcionamiento de este sistema.[13]

Para empezar definiremos que son los árboles de decisión. Este modelo consiste en un conjunto de árboles de clasificación y regresión (CART). A continuación, en la Fig 15 se muestra un ejemplo simple de un árbol de decisión:

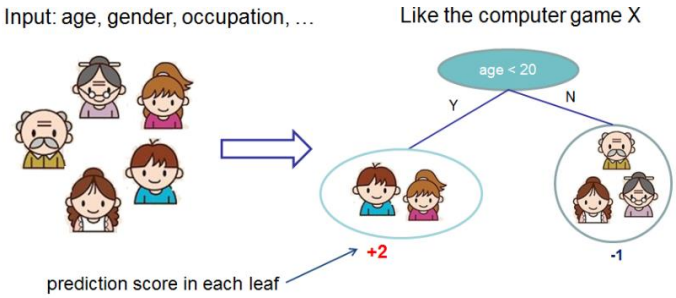



*Fig 15 Ejemplo de un árbol de decisión que clasifica si a alguien le gustará un hipotético juego x de ordenador.*

En este ejemplo clasificamos a los miembros de una familia en diferentes hojas y les asignamos la puntuación de la hoja correspondiente. CART es un poco diferente a los árboles de decisión, en los que la hoja solo contiene valores de decisión. En CART, una puntuación real está asociada con cada una de las hojas, lo que nos brinda interpretaciones más completas. Por lo general, un solo árbol no es lo suficientemente fuerte como para ser utilizado. En la práctica, lo que realmente se usa es un modelo conjunto, que suma la predicción de varios árboles a la vez. Aquí en la Fig 16 podemos ver un ejemplo:

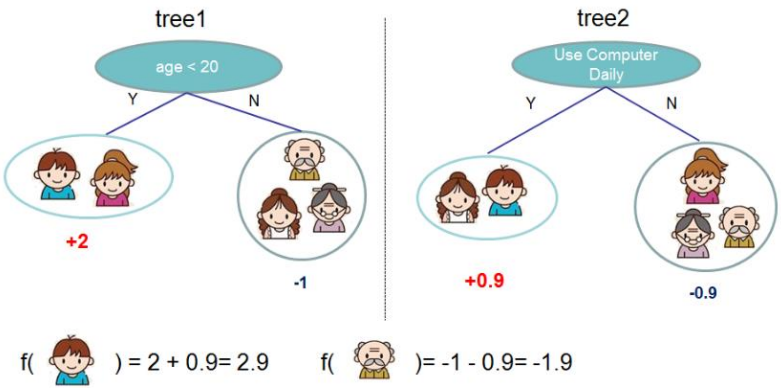



*Fig 16 Ejemplo de un CART que clasifica si a alguien le gustará un hipotético juego x de ordenador.*

Las puntuaciones de predicción de cada árbol individual se suman para obtener la puntuación final. Si miramos el ejemplo, un hecho importante es que los dos árboles intentan complementarse entre sí. Matemáticamente, podemos escribir nuestro modelo de la forma:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \epsilon F$$

*Ecuación 8*

Dónde $x_i$son los datos de entrada, $\hat{y}_i$ es la predicción de $y_i$, $K$ es el número de árboles, $f$ es una función en el espacio funcional $F$, y $F$ es el conjunto de todos los CART's posibles. La función objetivo a optimizar está dada por:

$$obj(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

*Ecuación 9*

Dónde $\theta$ son los parámetros y Ω el término de regularización.

Los *boosted trees* funcionan de la misma forma que los CART, a diferencia de que utilizan un método aditivo para tratar los árboles ya que computacionalmente no se pueden tratar todos a la vez como se ha descrito anteriormente. Este método aditivo consiste en corregir lo que se está aprendiendo y agregar a la vez un nuevo árbol. El árbol que se agrega será aquel que optimice nuestro objetivo. [14].

Fijándonos en a fig x podemos ver como se calculan las puntuaciones. Básicamente, para una estructura de árbol dada, empujamos las estadísticas $g_i$ y $h_i$ a las hojas a las que pertenecen, sumamos las estadísticas y usamos la fórmula descrita en la Fig 17 para calcular la precisión del árbol. Este resultado es análogo a la medida de impureza de árbol de decisión, excepto que también tiene en cuenta la complejidad del modelo.

$$g_i = \partial_{\hat{y}^{i(t-1)}} l(y_i, \hat{y}_i{}^{(t-1)})$$

*Ecuación 10*

$$h_i = \partial^2{}_{\hat{y}^{i(t-1)}} l(y_i, \hat{y}_i{}^{(t-1)})$$

*Ecuación 11*

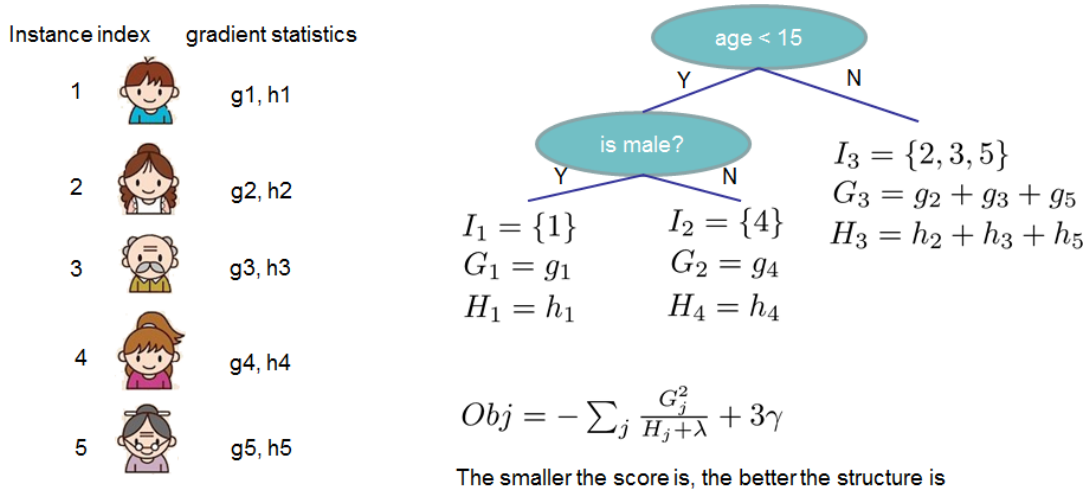



*Fig 17. Esquema del funcionamiento del método gradient boosting con el ejemplo de clasificación de si a alguien le gustará un hipotético juego x de ordenador.*

# 4. Experimentos y Resultados

En este capítulo se presentan y comparan los resultados obtenidos de la aplicación de las metodologías mencionadas en el capítulo anterior. El capítulo se divide en 4 partes: la preparación de las imágenes de las bases de datos para el entrenamiento, la presentación de los resultados del entrenamiento de la ResNet con la base de datos de 2018, la presentación de los resultados del entrenamiento de la ResNet con la base de datos de 2019, la presentación de los resultados del entrenamiento con las características de la red y la información de metadatos de la base de datos del 2018, la presentación de los resultados del entrenamiento con las características de la red y la información de metadatos de la base de datos del 2019 y la discusión de todos los distintos resultados obtenidos con los métodos mencionados.

## 4.1. Base de datos

Como se ha comentado en el capítulo 3.1 en este proyecto se han utilizado dos bases de datos, la correspondiente al *ISIC Challenge 2018* [3] y la correspondiente *al ISIC Challenge 2019* [4]. Para las dos bases de datos todas las imágenes se encontraban mezcladas en una misma carpeta, por lo que primero se tuvo que dividir las imágenes en distintas carpetas según su clase, para poder tratarlas correctamente. En los dos casos las bases de datos han sido divididas en un 80% entrenamiento y un 20% validación. En el primer caso la partición ha sido de 8.012 imágenes de entrenamiento y 203 imágenes de validación. En el segundo caso se han utilizado 20.265 imágenes de entrenamiento y 5.066 imágenes de validación. La partición está hecha de forma que se mantienen las proporciones de cada clase tanto en el entrenamiento como en la validación.

Las imágenes de la base de datos de 2018 tenían un tamaño de 600x450 píxeles, en cambio, las imágenes de la base de datos de 2019 tenían un tamaño de 512x384 píxeles. Tanto para el entrenamiento como para la validación todas las imágenes fueron re-escaladas a un tamaño de 224x224 píxeles, ya que la arquitectura ResNet sólo permite imágenes de entrada con una dimensiones múltiples de 224, además de esta forma utilizamos menos memoria para entrenar al red. A todas las imágenes de entrada se les ha aplicado giros verticales y horizontales además de rotaciones en distintos grados: 90, 180 y 270.

## 4.2. Entrenamiento

Este apartado está dividido en dos sub-apartados en que se presentaran los experimentos y resultados tanto del entrenamiento de la red con la arquitectura ResNet, como del entrenamiento de la información de metadatos y de los vectores de características de la red con la técnica basada *en gradient descent boosting trees* 3.3.2 para las dos bases de datos.

### 4.2.1. ResNet

El entrenamiento de la red se basa principalmente en la buena elección de los híper-parámetros que la componen, por lo que una elección incorrecta de estos puede llevar al sobre-entrenamiento de la red (*overffiting*) o simplemente al no entrenamiento de la red. Cada experimento con buenos resultados realizado para una de las bases de datos se ha repetido para la otra con las mismas condiciones para ver con cual se obtienen los mejores resultados. Primero se ha iniciado la red a unos híper-parámetros iniciales que han sido un *learning rate* de $10^{-4}$ , que le dice al optimizador, en este caso el optimizador Adam [15], cuánto deben moverse los pesos en la dirección del gradiente y un *batch size* de 12, que nos indica cuantas imágenes se tratan a la vez en el *forward pass.* El número de épocas varía dependiendo de cada experimento y es un número fijo. Para todos los experimentos se ha usado de arquitectura base la ResNet, variando el número de capas dependiendo de cada experimento.

Durante el entrenamiento de la red se han ido recopilando datos de las pérdidas y las tasas de acierto balanceadas obtenidas a cada época para la fase de entrenamiento y validación, de esta forma podíamos saber si el entrenamiento estaba evolucionando favorablemente. Para evaluar cuál de los experimento era el mejor se ha utilizado el valor obtenido de la tasa de acierto balanceada, es decir, el mejor experimento será aquel con una tasa de acierto balanceada mayor.

#### 4.2.1.1. Base de datos ISIC 2018

En este sub-apartado se detallaran los resultados de los distintos experimentos exitosos realizados para la base de datos de 2018.

El primer experimento que se muestra ha sido realizado con la arquitectura ResNet-50, con los siguientes híper-parámetros: *learning rate* con un valor fijo de $10^{-6}$ , un *batch size* de 36 y utilizando el optimizador Adam. Se ha calculado la función de pérdidas con los denominados *weighted loss*, (Ecuación 4). A continuación, se muestran las gráficas obtenidas de la tasa de acierto balanceada y de las pérdidas Fig 18:

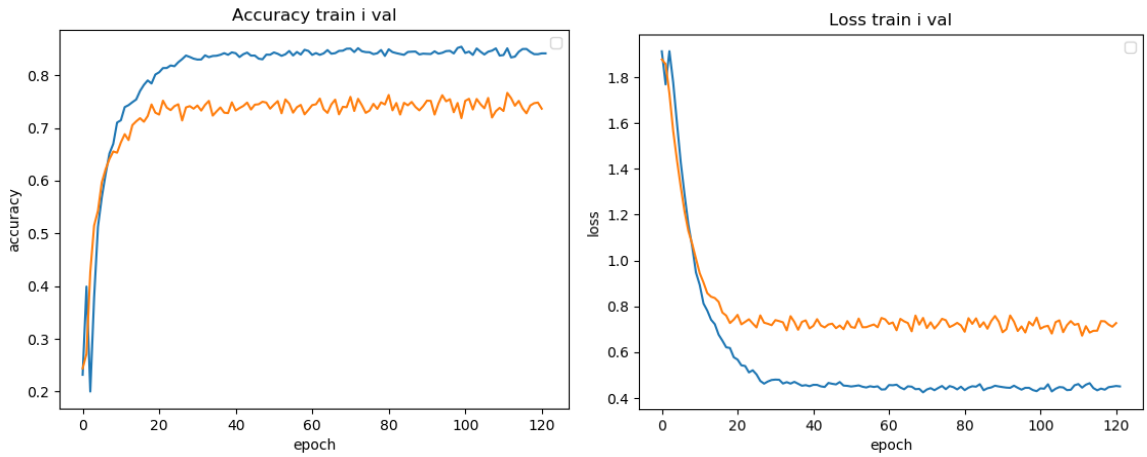



*Fig 18 Graficas de acierto y de pérdidas obtenidos con los híper-parámetros comentados anteriormente, una arquitectura ResNet-50 y con 120 épocas. Los valores de la fase train corresponden al color azul y los valores de la validación están marcados en naranja.*

Fijándonos en las dos gráficas vemos que empiezan a converger hacia la época 30, obteniendo una tasa de acierto balanceada de 0.759.

A continuación se muestra la matriz de confusión obtenida para este experimento:

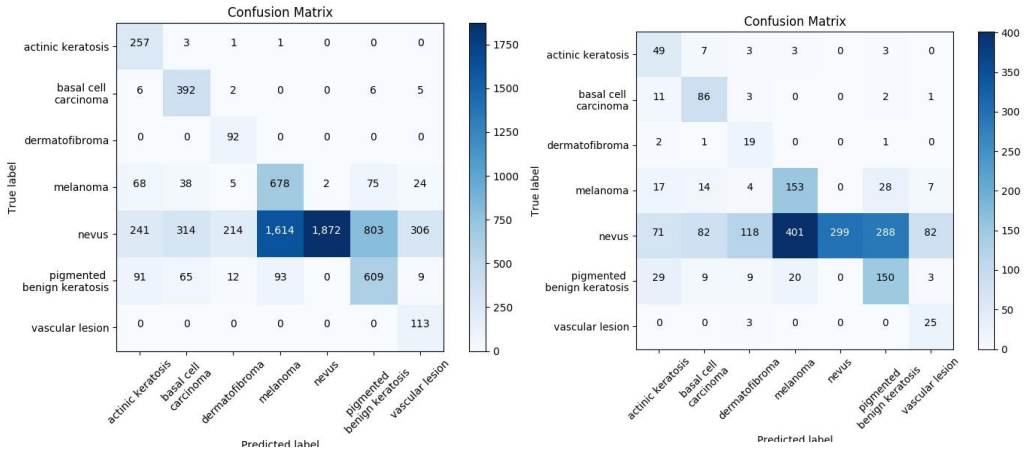



*Fig 19 Matriz de confusión de la fase de entrenamiento (derecha) y de la fase de validación (izquierda).*

Si nos fijamos en la matriz de confusión de la fase de validación (derecha), vemos que el clasificador funciona correctamente para las clases con pocas imágenes como por ejemplo, *vascular lesion* donde la clasificación es perfecta o *dermatofibroma*. En cambio podemos observar que para la clase con más imágenes de entrada, la clase *nevus,* la clasificación es incorrecta ya que detecta más imágenes como melanoma que como *nevus.*

El segundo experimento realizado ha sido con la arquitectura ResNet-101. En este caso la elección de los híper-parámetros ha estado condicionada por parte del experimento descrito anteriormente. En este caso se ha utilizado una *learning rate* adaptativa, es decir, se ha ido actualizando el valor de este híper-parámetro cada un cierto número de épocas. Concretamente se ha inicializado a un valor de $10^{-5}$ y se ha ido disminuyendo cada 10 épocas $10^{-1}$. El *batch size* utilizado ha sido de 14. Se ha reducido respecto al experimento anterior ya que al aumentar el número de capas de la red, en consecuencia, el coste computacional para entrenar la red es mayor, en consecuencia se ha reducido el número de imágenes de entrada para cada *forward pass*. En este caso también se ha utilizado el optimizar Adam y *weighted loss* (Ecuación 4) para calcular la función de pérdidas. En este caso el número de épocas de entrenamiento se ha reducido considerablemente, en concreto a 40 épocas, ya que en experimento anterior hemos visto que las gráficas empezaban a converger hacia la época 30. A continuación se muestran las gráficas obtenidas de la tasa de acierto balanceada y de las pérdidas:

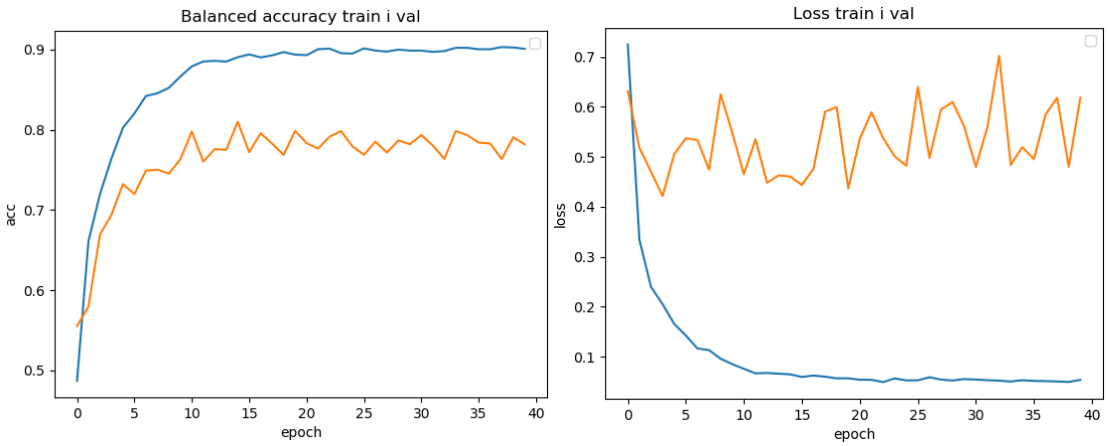



*Fig 20 Graficas de acierto y de pérdidas obtenidos con los híper-parámetros comentados anteriormente, una arquitectura ResNet-101 y con 40 épocas. Los valores de la fase train corresponden al color azul y los valores de la validación están marcados en naranja*

En este experimento vemos que la tasa de acierto balanceada empieza a converger hacia la época 15 dónde prende su valor más elevado de 0.810. También en esa época vemos que la función de pérdidas prende un valor de 0.422, que es un valor aceptable ya que es suficientemente pequeño.

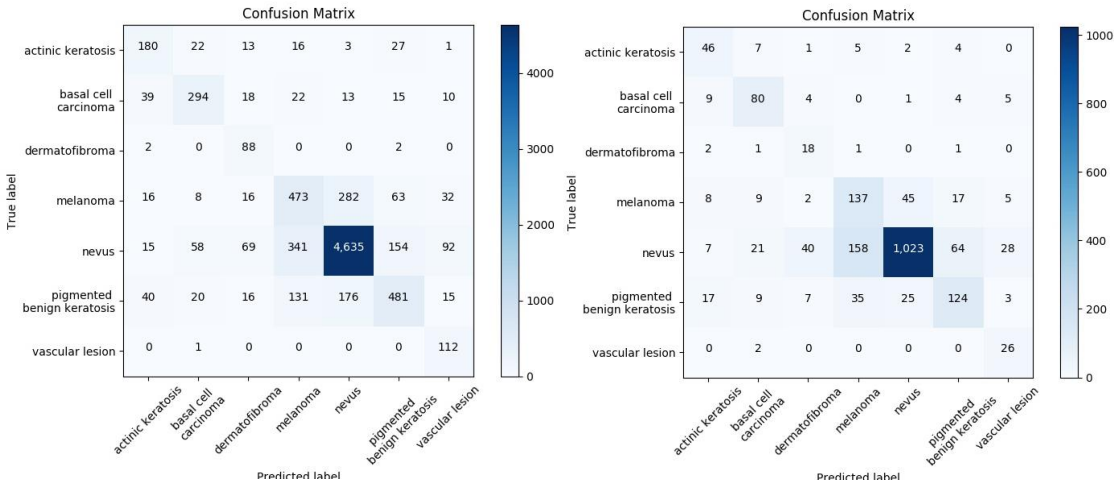



*Fig 21. Matriz de confusión de la fase de entrenamiento (derecha) y de la fase de validación (izquierda).*

En este caso analizando las matrices de confusión vemos que la clasificación ha mejorado considerablemente. Se siguen clasificando mejor las clases con menos imágenes, pero en este caso ha mejorado considerablemente la clasificación para las clases con más imágenes como *nevus*, *melanoma* o *pigmented benign keratosis*.

Para resumir los resultados obtenidos en estos dos experimentos, se ha creada la Tabla 7, dónde podemos comparar las tasas de acierto balanceadas obtenidas en cada caso.

| Modelo | Tasa de acierto balanceada |
|---|---|
| *ResNet-50* | 0.759 |
| *ResNet-101* | 0.810 |

*Tabla 7 Tabla comparativa de las tasas de acierto balanceadas obtenidas.*

La mejor tasa de acierto obtenida ha sido con el entrenamiento de la ResNet-101, con un valor es de 0.810. Si comparamos este resultado con los resultados remarcados en el apartado 2 correspondientes al estado del arte actual, vemos que la diferencia es pequeña, por lo que se considera un buen resultado.

#### 4.2.1.2. Base de datos ISIC 2019

En este sub-apartado se detallaran los resultados de los distintos experimentos realizados con la base de datos correspondientes al *ISIC Challenge 2019*.

El modelo de red fue entrenado con esta base de datos posteriormente al entrenamiento con la base de datos correspondiente al *ISIC Challenge 2018,* por lo que sólo se repitieron aquellos experimentos que hubieran sido exitosos.

A continuación, en la Fig 22 se presenta el primer experimento realizado con la arquitectura ResNet-50 y con los híper-parámetros de *batch size* 36, *learning rate* fijo de $10^{-6}$ y optimizador Adam. En este caso también se han utilizado *weighted loss* Ecuación 4 para el cálculo de la función de pérdidas como se había hecho para los experimentos con la base de datos del 2018.

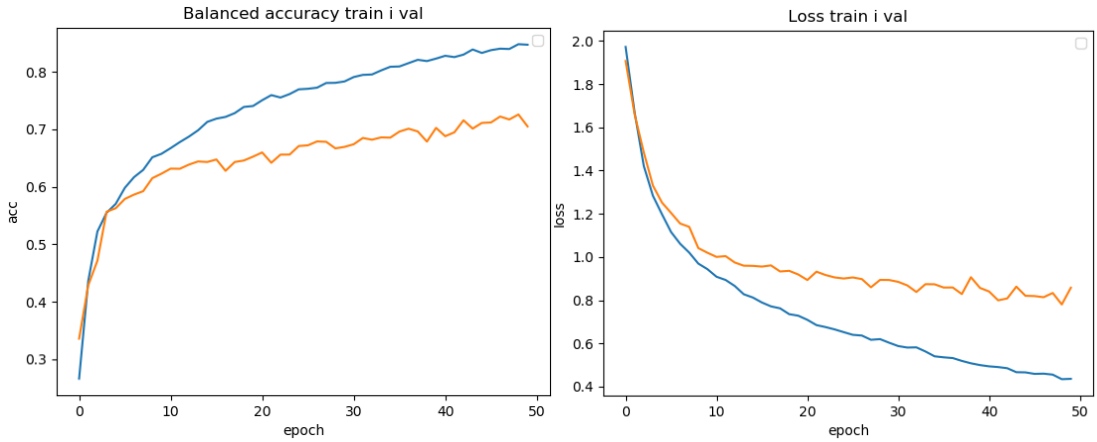



*Fig 22. Graficas de acierto y de pérdidas obtenidos con una arquitectura ResNet-50 y con 50 épocas. Los valores de la fase train corresponden al color azul y los valores de la validación están marcados en naranja.*

Si nos fijamos primero en la gráfica de la tasa de acierto balaceada vemos que no llega a converger del todo, la gráfica de pérdidas empieza a converger prematuramente a un valor de pérdidas bastante elevado, alrededor de 0.8. Para este experimento se ha obteniendo una tasa de acierto balanceada de 0.722.

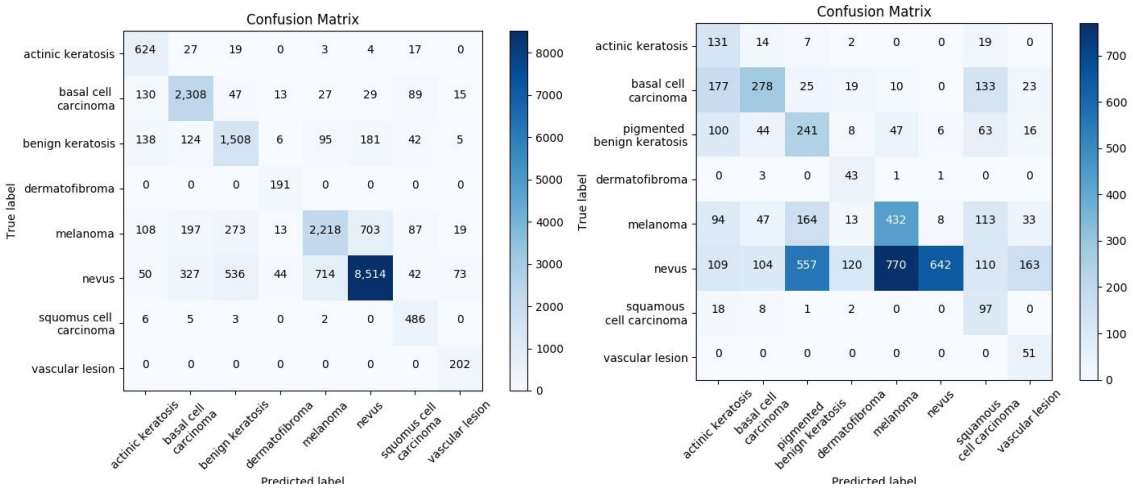



*Fig 23. Matriz de confusión de la fase de entrenamiento (derecha) y de la fase de validación (izquierda).*

La matriz de confusión para la fase de entrenamiento muestra una buena clasificación para todas las clases, y en concreto en las clases con pocas imágenes. En el caso de la matriz de confusión de la fase de validación, como ya pasaba para el entrenamiento de la ResNet-50 con la base de datos del 2018, muestra una clasificación incorrecta de las

clases con más imágenes, pero sigue manteniendo una clasificación casi perfecta para las clases con menos muestras de entrada.

El segundo experimento se ha realizado con la arquitectura ResNet-101. Se han utilizado los híper-parámetros y características mencionados en el segundo experimento de la base de datos del 2018. A continuación, se muestran las gráficas obtenidas de la tasa de acierto balanceada y de las pérdidas Fig 24:

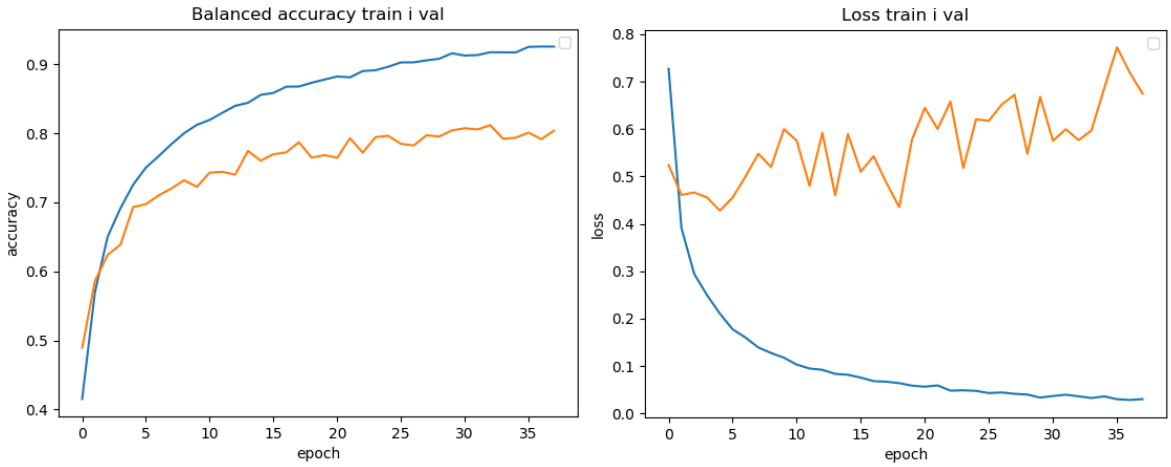



*Fig 24 Graficas de acierto y de pérdidas obtenidos con una arquitectura ResNet-101 y con 40 épocas. Los valores de la fase train corresponden al color azul y los valores de la validación están marcados en naranja.*

La gráfica correspondiente a las tasas de acierto balanceada empieza a converger hacia la época 30, obteniendo un valor de 0.812, en cambio vemos que en esta época en concreto la gráfica de pérdidas tiene un valor bastante elevado.

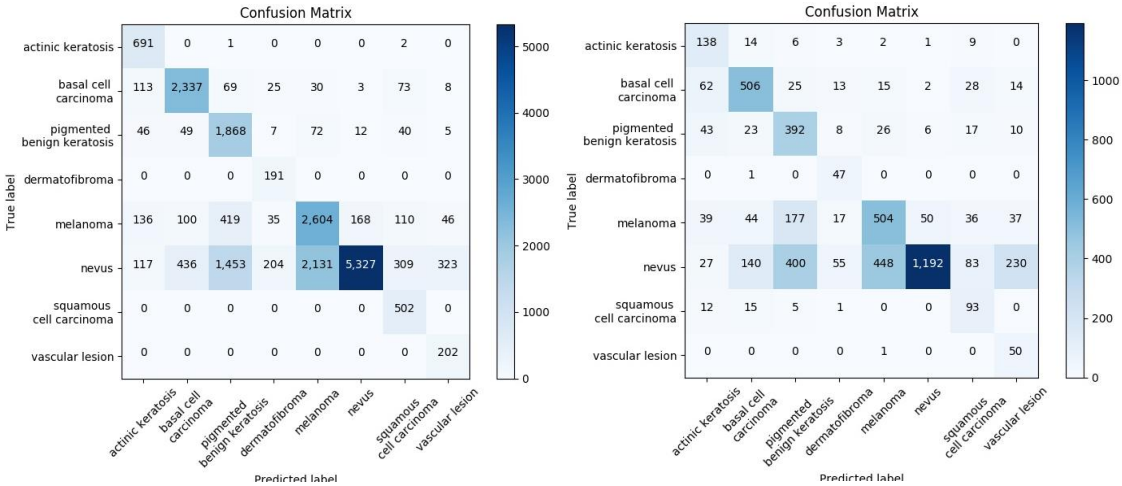



*Fig 25. Matriz de confusión de la fase de entrenamiento (derecha) y de la fase de validación (izquierda).*

En este caso los resultados de la matriz de confusión para la fase de validación han mejorado considerablemente respecto a los resultados anteriores y se obtiene una mejor clasificación para las clases con más imágenes. En todos los casos hemos observado una mejor clasificación para las clases con menos imágenes, esto es debido a que a las imágenes que pertenecen a estas clases se le asignan unos pesos más grandes, como se ha comentado en el sub-apartado 3.2.3.

Igual que en el sub-apartado anterior, para resumir los resultados obtenidos en estos dos experimentos, se ha creada la Tabla 8, dónde podemos comparar las tasas de acierto balanceadas obtenidas en cada caso.

| **Módelo** | **Tasa de acierto balanceada** |
|---|---|
| *ResNet-50* | 0.722 |
| *ResNet-101* | 0.812 |

*Tabla 8 Tabla comparativa de las tasas de acierto balanceadas obtenidas.*

La tabla nos muestra que el mejor resultado ha sido obtenido con la ResNet-101.

### 4.2.2. XGBoost

Para realizar el entrenamiento se han dividido los datos de entrada compuestos por los parámetros de características de la red extraídos de la última capa y de la información de metadatos en datos de entrenamiento (80%) y datos de test (20%).

De la misma forma que para el entrenamiento de la CNN utilizando la arquitectura ResNet, el éxito del entrenamiento se basa principalmente en la buena elección de los híper-parámetros que la componen. En este caso los híper-parámetros relevantes en la ejecución de los distintos experimentos han sido: el híper-parámetro eta, análogo a la *learning rate* en los casos descritos en los sub-apartados anteriores. Este parámetro puede tomar valores dentro del rango [0,1], y hace que el modelo sea más robusto al reducir los pesos en cada paso. El Siguiente híper-parámetro es la profundidad máxima que puede tener un árbol, utilizada para controlar el ajuste excesivo, ya que una mayor profundidad permitirá al modelo aprender relaciones muy específicas para una muestra en particular. Los valores típicos son entre 3-10; Y el híper-parámetro *scale position weight,* se debe utilizar un valor mayor a 0 en los caso de desequilibrio entre clases, ya que ayuda a una convergencia más rápida.

En los experimentos detallados posteriormente los híper-parámetros utilizados han sido: eta de 0.3, profundidad máxima del árbol de 6 y *scale position weight* de 5 ya que son los valores que se han obtenido aplicando la técnica de *grid search,* que nos dice que híper-parámetros son los más óptimos para clasificar correctamente los datos.

Finalmente el objetivo de optimización nos define la función de pérdidas que tiene que ser minimizada, en este caso se ha utilizado la función *softmax*, ya que es la más típica para casos multiclase.

En este caso al igual que en el sub-apartado anterior, se divide la presentación de los experimentos y resultados en la base de datos del 2018 y la base de datos de 2019. Para las dos bases de datos se han realizados los mismos experimentos. En concreto se presentan dos experimentos para cada una. El primer experimento consiste en entrenar el clasificador con únicamente como datos de entrada los vectores de características extraídos del entrenamiento de la ResNet. El segundo experimento integra como datos de entrada los vectores de características con la información de metadatos para cada imagen. Se han realizado estos dos experimentos para poder comprobar si realmente la información de metadatos aporta al entrenamiento, o de lo contrario, no es un dato significativo para aumentar la tasa de acierto balanceada.

#### 4.2.2.1. Base de datos ISIC 2018

El primer entrenamiento realizado ha sido como hemos dicho únicamente con los datos de entrada referentes a los vectores de características de la red. En concreta las características extraídas del entrenamiento mostrado en el segundo experimento del sub-apartado 4.2.1.1, correspondiente al entrenamiento con la ResNet-101 ya que tenía una mayor tasa de acierto balanceada.

Para este entrenamiento se ha obtenido una tasa de acierto balanceada de 0.689.
El segundo experimento realizado corresponde al entrenamiento del clasificador con la información de metadatos además de los vectores de características de la red. En este caso se ha obtenido una tasa de acierto balanceada de 0.672.

| **Esquema de clasificación** | **Tasa de acierto balanceada** |
|---|---|
| *Características –ResNet-101* | 0.689 |
| *Características –ResNet-101+ Información de metadatos* | 0.672 |

*Tabla 9. Tabla de las tasas de acierto balanceadas obtenidas en los experimentos descritos en el sub-apartado 4.2.2.1.*

En este caso podemos ver que la clasificación añadiendo la información de metadatos ha ido peor que la clasificación únicamente de las características de la red. Lo se nos indica que para este modelo de datos no se obtienen buenos resultados.

#### 4.2.2.2. Base de datos ISIC 2019

Para esta base de datos se han realizado los mismos experimentos que para la base de datos del 2018.

En el primer experimento se ha obtenido una tasa de acierto balanceada de 0.636 y para el segundo experimento se ha obtenido una tasa de acierto balanceada de 0.676.

Podemos concluir observando la Tabla 10, que la clasificación mediante el modelo XGBoost no ha funcionado correctamente con estos datos de entrenamiento, ya que se ha obtenido mejores resultados con el modelo basado en la arquitectura ResNet comentado en el sub-apartado 4.2.1.

| **Esquema de clasificación** | **Tasa de acierto balanceada** |
|---|---|
| *Características –ResNet-101* | 0.636 |
| *Características –ResNet-101+ Información de metadatos* | 0.676 |

*Tabla 10. Tabla de las tasas de acierto balanceadas obtenidas en los experimentos descritos en el sub-apartado 4.2.2.2.*

# 5. Presupuesto

Este proyecto se ha llevado a cabo en el Grupo de Procesamiento de Imagen del departamento de Teoria de la Señal y Comunicaciones de la Universidad Politécnica de Catalunya, UPC.

El Deep learning exige unos términos de computación elevados, por lo que se necesitaba una GPU para la ejecución. La GPU GeForce GTX Titan Black tiene un costo aproximado de 920 euros, sin embargo, la UPC nos la proporcionó sin coste alguno.

La mayoría del software utilizado ha sido libre (*Python, Pytorch*). El único software con un coste en la licencia ha sido *PyCharm*, con una licencia anual de 199.00 €/año, pero nuevamente no ha aportado ningún coste al proyecto al formar parte de la comunidad universitaria UPC.

Por lo tanto, el coste principal de este proyecto proviene del salario de los investigadores y del tiempo invertido en él. El equipo para el desarrollo de este proyecto está formado por dos profesores que me aconsejaron como ingenieros superiores y yo como ingeniera junior. La duración total del proyecto ha sido de 20 semanas. A continuación en la Tabla 11 se muestra el presupuesto detallado del proyecto:

| | ***Cantidad*** | ***Salario / hora*** | ***Dedicación*** | ***Total*** |
|---|---|---|---|---|
| ***Ingeniero júnior*** | 1 | 8 € / h | 25 h / week | 4.000 € |
| ***Ingeniero sénior*** | 2 | 20 € / h | 2 h / week | 1.600 € |

*Tabla 11. Presupuesto detallado del Proyecto.*

# 6. Conclusiones y trabajo futuro

En primer lugar se quiere remarcar la dificultad de la preparación de los datos para un entrenamiento de estas características. En este proyecto se han preparado dos clases de datos de entrenamiento, la base de datos de imágenes del 2019 y la información de metadatos normalizada tanto para la base de datos 2018 como para la base de datos de 2019. Puede parecer una parte fácil de ejecutar, pero sobretodo en la parte de la preparación de la información de los metadatos han surgido bastantes problemas en cuanto a formato y también a la hora de juntar los vectores de características de cada imagen con su respectiva información de metadatos normalizada.

En segundo lugar, se comentan los resultados obtenidos en el apartado 4. En la Tabla 12 podemos ver la tasa de acierto balanceada obtenida para cada uno de los experimentos detallados anteriormente. Podemos ver que de todos los esquemas de clasificación el que obtiene mejores resultados es el entrenamiento de la ResNet-101 con la base de datos del 2019. Este resultado se debe a que la base de datos del 2019 cuenta con 10500 imágenes más que la base de datos del 2018, debido en parte a que cuenta una clase más, por lo que a mayor número de imágenes de entrada, mayores tasa de acierto balanceada se espera. También podemos ver que la ResNet-101 tiene el doble de capas que la ResNet-50, por lo que también favorece a mejorar los resultados que se habían obtenido con el entrenamiento de la ResNet-50. Cabe destacar que los resultados obtenidos con el entrenamiento de la arquitectura ResNet-101 son prácticamente los mismos para las dos bases de datos de imágenes, esto nos dice que aunque la base de datos del 2018 tenga menos imágenes se obtienen buenos resultados con ella.

Para el entrenamiento con el clasificador XGBoost basado en árboles de clasificación se han obtenido mejores resultados con la base de datos del 2018. A diferencia que el entrenamiento con la arquitectura ResNet.

Si comparamos los resultados del entrenamiento con los distintos tipos de arquitectura ResNet y del entrenamiento de los vectores de características extraídos de la red con el clasificador XGBoost, vemos que es más robusta la clasificación con la arquitectura ResNet que con el clasificador ya que los resultados son notablemente mejores.

Finalmente, como hemos comentado en el sub-apartado anterior 4.2.2 la introducción de la información de metadatos al entrenamiento del clasificador con datos de entrada los vectores de características extraídos de la red, mejora los resultados obtenidos, pero esta mejora es muy pequeña. Concluimos que el modelo basado en la arquitectura ResNet ha

obtenido mejores resultados que el modelo XGBoost, por lo que se podrían probar otras formas de combinar los datos para poder mejorar los resultados que se han obtenido con el modelo XGBoost, ya que, desafortunadamente no se han podido mejorar los resultados obtenidos. Por lo que se puede concluir que la información de metadatos no ha sido de ayuda para mejorar los diagnósticos.

| Base de datos | Esquema de clasificación | Tasa de acierto balanceada |
|---|---|---|
| **2018** | ResNet-50 | 0.759 |
| | ResNet-101 | 0.810 |
| | Características ResNet-101 + XGBoost | 0.689 |
| | Características ResNet-101 + Información de metadatos + XGBoost | 0.672 |
| **2019** | ResNet-50 | 0.722 |
| | ResNet-101 | **0.812** |
| | Características ResNet-101 + XGBoost | 0.636 |
| | Características ResNet-101 + Información de metadatos + XGBoost | 0.676 |

*Tabla 12.Tabla comparativa de las tasas de acierto balanceadas obtenidas en los experimentos detallados anteriormente.*

## 6.1. Futuras mejoras

Una vez realizado el proyecto se presentan algunas posibles implementaciones para mejorar los modelos utilizados. Una posible mejora de robustez para el modelo de entrenamiento de la arquitectura ResNet seria dividir las bases de datos de imágenes de forma distinta, utilizar el método *k-folds,* que consiste en dividir a base de datos en k grupos de x imágenes, cada grupo se utiliza una vez como validación, mientras que los grupos k-1 restantes forman el conjunto de entrenamiento.

Para mejorar el entrenamiento de la clasificación con los vectores de características extraídos de la red y la información de metadatos se podrían utilizar otros tipos de

clasificador como por ejemplo, el *Support Vector Machine* (SVM) [20]. Por otra parte se podría probar otro tipo de combinación de datos ya que para cada imagen se tiene un número de características de la red mucho más grande que el número de metadatos, podría reducirse el número de características extraídas de la red.

# Referencias:

[1] Sociedad Española de Oncología Médica (SEOM), Depósito Legal: M-2172-2017. URL https://seom.org/seomcms/images/stories/recursos/Las_cifras_del_cancer_en_Esp_2017.pdf

[2] National Cancer Institute, 2016. URL: https://seer.cancer.gov/statfacts/html/melan.html

[3] ISIC Challenge 2018. URL https://challenge2018.isic-archive.com

[4] ISIC Challenge 2019. URL https://challenge2019.isic-archive.com/

[5] Tschandl P., Rosendahl C. & Kittler H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci. Data 5, 180161 doi.10.1038/sdata.2018.161 (2018).

[6] Aleksey Nozdryn-Plotnicki, Jordan Yap, and William Yolland. Ensembling Convolutional Neural Networks for Skin Cancer Classification. MetaOptima Technology Inc. Vancouver, BC, Canada, 2018.

[7] Nils Gessert, Thilo Sentkerac, Frederic Madestaac, Rudiger Schmitz, Helge Kniepag, Ivo Baltruschataef , Rene Werner and Alexander Schlaeferb. Skin Lesion Diagnosis using Ensembles, Unscaled Multi-Crop Evaluation and Loss Weighting. DAISYLab, Institute of Medical Technology at Hamburg University of Technology (TUHH), 2018.

[8] Keras API.URL https://keras.io/

[9] Fei-Fei Li, Justin Johnson, and Serena Yeung. Cs231n: Convolutional neural networks for visual recognition. 2017. URL http://cs231n.stanford.edu/.

[10] Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun. Deep Residual Learning for Image Recognition. Microsoft. December, 2015.

[11] Linear classification, softmax clasifier. URL http://cs231n.github.io/linear-classify/

[12] Brodersen, K.H.; Ong, C.S.; Stephan, K.E.; Buhmann, J.M. (2010). The balanced accuracy and its posterior distribution. Proceedings of the 20th International Conference on Pattern Recognition, 3121-24.

[13] Yap J, Yolland W, Tschandl P. Multimodal skin lesion classification using deep learning. Exp Dermatol. 2018;27:1261–1267.

[14] Sun X., Yang J., Sun M., Wang K. (2016) A Benchmark for Automatic Visual Classification of Clinical Skin Disease Images. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9910. Springer, Cham.

[15] Tianqi Chen, Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. University of Washington, 10 June 2016.

[16] «Wikipedia,» [Online]. URL https://en.wikipedia.org/wiki/Gradient_boosting#Gradient_tree_boosting.

[17] Tianqi Chen. Introduction to boosted trees. University of Washington. 22 October 2014.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980

[19] Analytics Vidhya. Complete Guide to Parameter Tuning in XGBoost with codes in Python. Aarshay jain, marzo 1, 2016 URL https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

[20] Support Vector Machine, SVM. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[21] Thesis. URL https://github.com/tdomenech/skin-lesion-clasification

# Apéndice:

## 6.2. Código del proyecto

El código del proyecto se puede encontrar en el siguiente repositorio de Github [21]. Ha sido desarrollado únicamente con Python utilizando la librería Pytorch.

## 6.3. Workplan

Work packages

| | | |
|---|---|---|
| Project: Project proposal and workplan | WP ref: WP 1 | |
| Major constituent: Documentation | Sheet 1 of 8 | |
| Short description:<br><br>Documentation on project basis, project goals, and project organization (work plan). | Planned start date:05/03/2019<br>Planned end date: 08/03/2019<br><br>Start event:T1<br>End event:T3 | |
| **Internal task T1:** Project definition and description<br><br>**Internal task T2**: Project development plan<br><br>**Internal task T3:** Document review and approval | Deliverables:<br>project_proposal<br>_and_workplan.doc | Dates:<br>10/03/2019 |

| | |
|---|---|
| Project: Information research and documentation | WP ref: WP 2 |
| Major constituent: Information research | Sheet 2 of 8 |
| Short description: | Planned start date: 13/02/2019<br>Planned end date: 11/03/2019 |

| | | |
|---|---|---|
| State-of-the-art analysis (involves acquiring information and some ideas for improvements). | Start event:T1<br>End event:T3 | |
| **Internal task T1:** Do CS231n Stanford course about deep learning and machine learning and the Andrew Ng course about deep learning from coursera.<br><br>**Internal task T2:** Familiarization with Python and the library Pytorch.<br><br>**Internal task T3:** Study of the state-of-the-art on medical image classification using CNN. | Deliverables: - | Dates: - |

| | | |
|---|---|---|
| Project: Image data base preparation | WP ref: WP 3 | |
| Major constituent: Image data base | Sheet 3 of 8 | |
| Short description:<br><br>Prepare the image data base for the competition of the MICCAI 2019 conference and for the project. | Planned start date:04/03/2019<br>Planned end date:07/03/2019 | |
| | Start event:T1<br>End event:T1 | |
| **Internal task T1**: Check the images form the Clinic Hospital data base to avoid duplicates and error tags. | Deliverables: - | Dates: - |

| | |
|---|---|
| Project: Software development | WP ref: WP 4 |
| Major constituent: Software | Sheet 4 of 8 |
| Short description: | Planned start date:12/03/2019 |

| | | |
|---|---|---|
| Learning how to use the appropriate software to classification the images using metadata information according to the different classes. | Planned end date: 15/05/2019<br>Start event:T1<br>End event:T3 | |
| **Internal task T1:** Use transfer learning to adapt a CNN to this case.<br><br>**Internal task T2:** Apply metadata information to have a system more robust.<br><br>**Internal task T3:** Test the implementation with the skin lesions image data base and obtain good results. | Deliverables: - | Dates: - |

| | | |
|---|---|---|
| Project: Critical review | WP ref: WP 5 | |
| Major constituent: Document | Sheet 5 of 8 | |
| Short description:<br><br>Writing of the document discussing the initial work plan and including modifications and reviews according to the real project evolution. | Planned start date: 05/04/2019<br>Planned end date: 12/04/2019<br>Start event:T1<br>End event:T4 | |
| **Internal task T1**: Analyze the project's results and compare to the initial work plan.<br><br>**Internal task T2:** Review of the work plan.<br><br>**Internal task T3:** Write the document.<br><br>**Internal task T4:** Document review and approval. | Deliverables:<br>Critical_review.doc | Dates:<br>12/04/2019 |

| | | |
|---|---|---|
| | | |

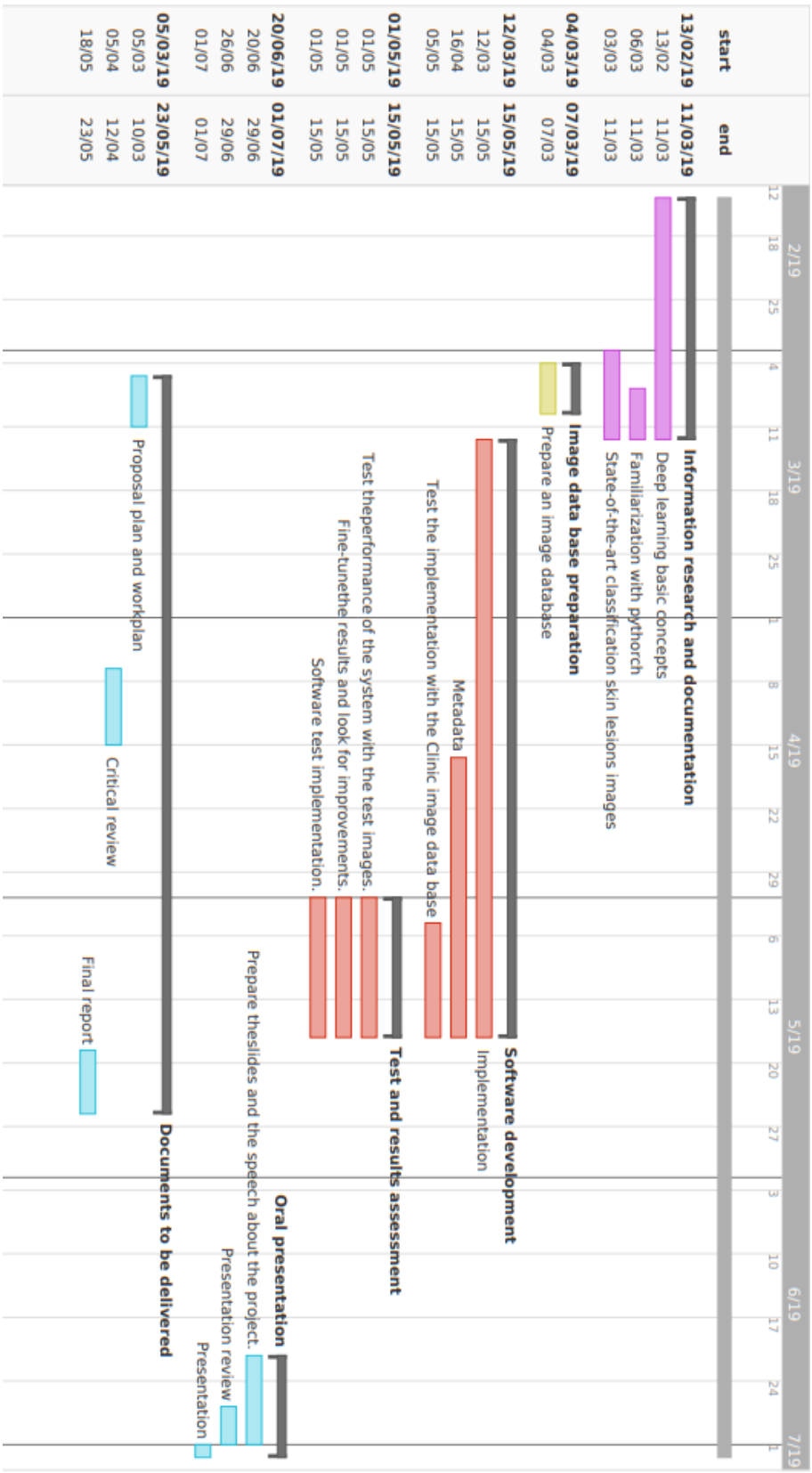
| | | |
|---|---|---|
| Project: Test and results assessment | WP ref: WP 6 | |
| Major constituent: Documentation and software | Sheet 6 of 8 | |
| Short description:<br><br>Get the final results and test the CNN under the skin lesions image data base.<br>Compare these results to other approaches done in the state-of-the-art and later with the competitors of MICCAI 2019. | Planned start date: 1/05/2019<br>Planned end date: 15/05/2019<br><br>Start event:T1<br>End event:T3 | |
| **Internal task T1:** Test the performance of the system with the test images.<br><br>**Internal task T2:** Fine-tune the results and look for improvements.<br><br>**Internal task T3:** Software test implementation. | Deliverables: - | Dates: - |

| | | |
|---|---|---|
| Project: Final report | WP ref: WP 7 | |
| Major constituent: Document | Sheet 7 of 8 | |
| Short description:<br><br>Is the document that analyze the entire project and the development once it has been finished. | Planned start date: 18/05/2019<br>Planned end date: 23/05/2019<br><br>Start event:T1<br>End event:T3 | |
| **Internal task T1:** Analyze the work done along the project. | Deliverables:<br>final_review.doc | Dates:<br>25/05/2019 |

| | | |
|---|---|---|
| **Internal task T2:** Write the document.<br><br>**Internal task T3:** Document review and approval. | | |

| | | |
|---|---|---|
| Project: Oral presentation | WP ref: WP 8 | |
| Major constituent: Presentation | Sheet 8 of 8 | |
| Short description:<br><br>Oral presentation that overviews of the project. It relies on a<br>slide-based presentation document. | Planned start date: 20/06/2019<br>Planned end date: 29/06/2019<br><br>Start event:T1<br>End event:T3 | |
| **Internal task T1:** Prepare the slides and the speech about the project.<br><br>**Internal task T2:** Presentation review.<br><br>**Internal task T3:** Do the presentation. | Deliverables:<br>presentation.pdf | Dates:<br>01/07/19 – 05/07/19 |

Diagrama de gantt



## 6.4. Experimentos adicionales

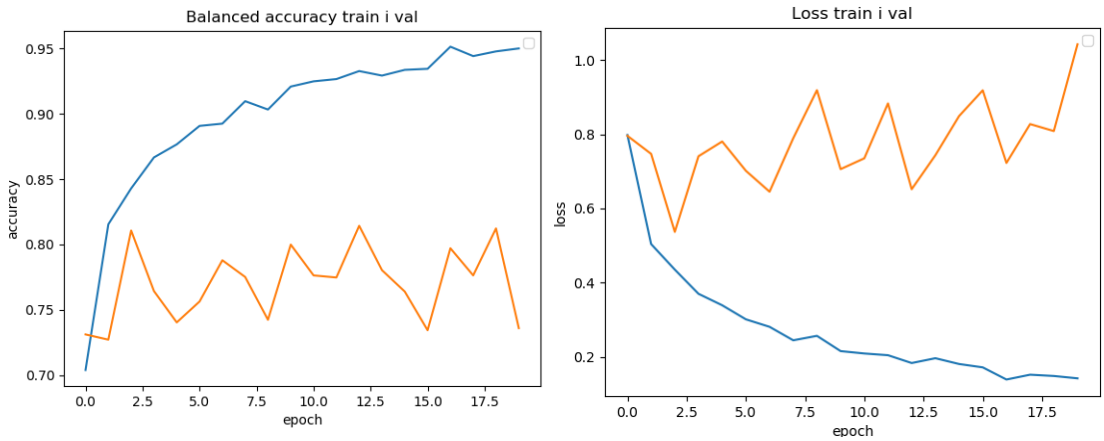
Experimento A:

- Arquitectura: ResNet-18
- Optimizador: Adam
- Learning rate: $10^{-4}$
- Batch size: 14
- Función de pérdidas: *Cross-entropy loss* sin *weighted loss*
- Número de épocas: 20
- Base de datos: 2018

Experimento B:

- Arquitectura: ResNet-50
- Optimizador: Adam
- Learning rate:$10^{-5}$
- Batch size: 36
- Función de pérdidas: *Cross-entropy loss* sin *weighted loss*
- Número de épocas:
- Base de datos: 2018

Experimento A:



Experimento B: