DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Detección Automática de Órganos en adquisiciones de Tomografía Computarizada con Métodos de Machine Learning

Autor: Rafael López González

Tutorizado por: Ángel Alberich Bayarri y Rafael Llobet Azpitarte

Trabajo Fin de Master presentado en el Departamento de Sistemas Informáticos y Computación de la Universitat Politècnica de València, para la obtención del Título de Master de Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital.

Curso 2016-17

Valencia, 15 de septiembre de 2017

Resumen

El objetivo del proyecto es entrenar un clasificador mediante métodos de Aprendizaje Automático que sea capaz de detectar órganos del cuerpo humano en adquisiciones de Tomografía Computarizada de forma autónoma para facilitar la labor de análisis de imágenes médicas a los profesionales del sector sanitario. Para ello se ha etiquetado manualmente una considerable cantidad de adquisiciones, con el propósito de extraer características de dichas muestras etiquetadas para entrenar el clasificador.

Resum

L´objectiu del projecte es entrenar un classificador mitjançant mètodes d'Aprenentatge Automàtic que siga capaç de detectar òrgans del cos humà en adquisicions de Tomografia Computada de manera autónoma per a facilitar la tasca d'anàlisi d'imatges mèdiques als profesionals del sector sanitari. Per a aconseguir l´objectiu proposat s´han etiquetat manualment una considerable quantitat d´aquisicions, amb el propòsit d´extraure característiques d´aquestes mostres etiquetades per a entrenar el classificador.

Abstract

The aim of this project is training a classifier with Machine Learning methods which is capable to automatically locate human body´s organs in Computed Tomographies volumes to ease the medical image analysis for health sector professionals. To achieve this objective a considerable amount of CT volumes have been labeled manually in order to extract features that will be used to train the classifier.

Índice

Capítulo 1. Introducción

1- Motivación

La sanidad es uno de los pilares fundamentales de cualquier sociedad moderna, por tanto, mejorar los procesos relacionados con la misma es crucial debido a que esto repercute de forma muy positiva en la vida de millones de personas.

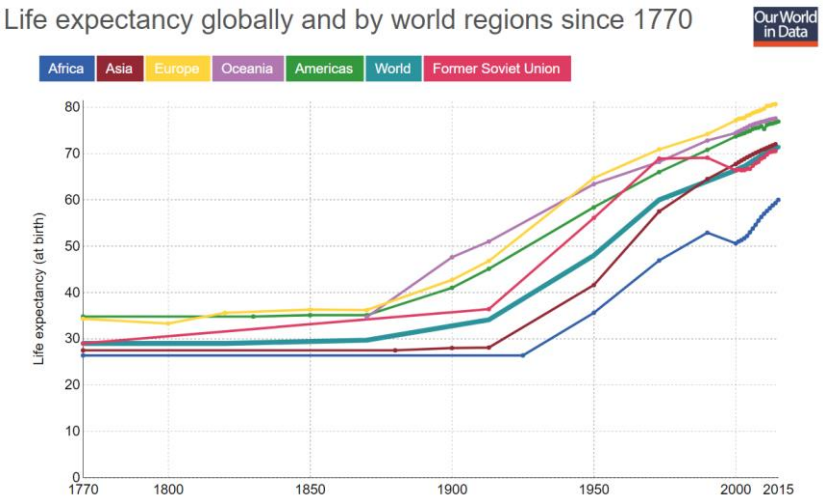

Figura 1: Evolución de la esperanza de vida desde 1770 hasta la actualidad [8]

Como se puede observar en la Figura 1 la esperanza de vida ha aumentado desde el rango de 30-40 años a 60-80 en los últimos siglos. Esta tendencia ha sido propiciada principalmente por los avances en el campo de la salud.

Una vez recalcada la importancia de la sanidad en la sociedad y las ventajas que supone una mejora de la misma es importante destacar las áreas clave de mejora, las cuales son la eficacia y la eficiencia en el diagnóstico.

Por un lado, la eficacia en el diagnóstico supone que se den estimaciones más precisas sobre la presencia/ausencia de enfermedad en los pacientes, lo que brindaría la posibilidad de salvar más vidas en última instancia. Por otro lado, la eficiencia permitiría aumentar el número de diagnósticos que se podrían realizar dedicando los mismos recursos, lo que permitiría aliviar la saturación de algunos sistemas sanitarios.

Mediante técnicas de Machine Learning se pueden desarrollar modelos que permitan que el campo de la salud avance tanto a nivel de eficiencia como de eficacia. Esto es así debido a que al automatizar ciertos procesos del diagnóstico se podría acelerar drásticamente el mismo a la vez que se reduce la carga de trabajo de los profesionales de la sanidad (excesiva en ocasiones). Además, se ha demostrado en diferentes estudios que los modelos entrenados mediante diferentes métodos de Machine Learning pueden

diagnosticar de forma más precisa que profesionales experimentados en algunos casos [9].

Por todo lo mencionado anteriormente la motivación del presente proyecto es la de mejorar el proceso de diagnóstico sanitario empleando técnicas de Machine Learning con el objetivo último de repercutir de forma positiva a la sociedad.

La detección de órganos de forma automática puede mejorar el proceso de diagnóstico por 2 vías principales. Por un lado, permite dotar a los profesionales de la sanidad con una herramienta de diagnóstico focalizada, la cual permitiría que si un radiólogo busca anomalías en un órgano determinado no necesite navegar por toda la tomografía hasta localizar dicho órgano debido a que la herramienta mencionada solo le brindaría los cortes en los que aparece el órgano en cuestión. Esto agilizaría en gran medida el proceso de diagnóstico debido a que el médico no desperdiciaría tiempo buscando la información que necesita dentro de la tomografía ya que la herramienta propuesta le proporcionaría justo el órgano o órganos que requiera.

Por otro lado, la detección automática de órganos también puede utilizarse como un bloque de pre-procesado en un algoritmo que sea capaz de diagnosticar de forma autónoma, para que, una vez segmentados los órganos, otra clase de algoritmos sean capaces de detectar anomalías en ellos. El diagnóstico automático es un problema mucho más complejo que el reconocimiento de órganos debido a que sería necesario tener un set de datos muy extenso de cada una de las enfermedades que se quisieran diagnosticar, y, por tanto, queda fuera de los objetivos del presente proyecto.

Por los motivos explicados anteriormente se ha considera que el proyecto propuesto puede repercutir de forma directa a los profesionales del diagnóstico médico proporcionándoles herramientas que les puedan ayudar a llevar a cabo su trabajo de forma más eficaz, lo que repercutía de forma indirecta en los pacientes, los cuales recibirían un diagnóstico, y por ende un tratamiento, más rápido.

2- Objetivos

Partiendo de la motivación descrita anteriormente se han definido unos objetivos claros para llevar a cabo el proyecto propuesto, los cuales son:

- Analizar el estado del arte de la detección automática de órganos en Tomografías Computarizadas con técnicas de Machine Learning.
- Partiendo del objetivo anterior, elegir la técnica de Machine Learning que se considere que mejores resultados puede proporcionar.
- Etiquetar las 200 tomografías que han sido proporcionadas por el Hospital La Fe de Valencia para generar un dataset con el que pueda trabajar la técnica de Machine Learning que se haya escogido.

- Implementar un software en Python que sea capaz de crear un dataset válido a partir de las tomografías etiquetadas.

- Implementar un software en Python que sea capaz de entrenar un modelo de detección de órganos con la técnica de Machine Learning seleccionada.

La consecución de los objetivos descritos permitirá el desarrollo de una herramienta experimental de detección de órganos en tomografías computarizadas que servirá como prueba de concepto para el futuro desarrollo de un sistema profesional.

Capítulo 2. Estado del arte de la detección de órganos mediante Machine Learning

Debido a la creciente popularidad del Machine Learning en los últimos años para resolver problemas complejos que anteriormente se consideraban inabarcables, se han presentado diversas aproximaciones para resolver la detección de órganos en adquisiciones de TC. Por contrapartida, todas las técnicas que se han desarrollado a día hoy para atacar el problema no han sido más que primeras tomas de contacto que no se han implementado en soluciones industriales finales por sus resultados insuficiente precisos.

Las técnicas que han gozado de más popularidad en el mundo de la investigación en los últimos años para enfrentarse al problema de la detección automática de órganos en TC han sido desarrolladas por el equipo de Antonio Criminisi [2][4][5][7]. Estás técnicas se dividen claramente en dos grupos principales: las que utilizan Bosques de Clasificación como método de detección de órganos, y, por otra parte, las que utilizan Bosques de Regresión.

Por otra parte, desde el año 2012 la popularidad de las redes neuronales no ha dejado de crecer debido a los extraordinarios resultados que están brindando en una inmensa variedad de problemas diferentes, y, aunque su utilización en la detección de órganos no es muy extensa en la bibliografía, se ha considerado que es un método a tener en cuenta para poder afrontar este problema.

Un factor común con el que cuentan las tres técnicas mencionadas, y, todas las técnicas de Machine Learning en general, es que sus resultados van a depender en gran medida de la calidad y extensión del dataset con el que se entrenen. Este es uno de los principales factores limitantes para utilizar Machine Learning en problemas relacionados con el campo de la sanidad, debido a que los datos de los pacientes están enormemente protegidos por cuestiones de privacidad, y, por ello, reunir un dataset extenso de adquisiciones de TC es muy complicado. En la Figura 2 se puede observar el comportamiento típico de los diferentes parámetros para evaluar la calidad de los modelos entrenados con Machine Learning al aumentar el tamaño de las muestras de entrenamiento (dataset). Como puede apreciarse al aumentar el tamaño del dataset mejoran todos los parámetros.

Figura 2: Precisión vs Tamaño del dataset

1- Bosques de Decisión

2.1.1 Explicación de la técnica

2.1.1.1 Árboles de Decisión [10, 11]

Los árboles de decisión son una técnica que se desarrolló hace varias décadas [13], pero su uso se ha popularizado en los últimos 10 años debido a que se ha conseguido aumentar su generalización mediante la creación de conjuntos de árboles diferentes, en lugar de utilizar solo uno. Estos conjuntos de árboles son llamados Bosques de decisión, los cuales serán explicados en detalle en el siguiente apartado.

Un árbol en Ciencias de la Computación es un tipo especial de grafo. Es una estructura de datos formada por una colección de nodos y vértices organizados de un modo jerárquico. Los nodos se dividen entre nodos internos y nodos terminales. Excepto el nodo raíz cada uno de los nodos tiene un solo vértice entrante. A diferencia de los grafos generales un árbol no tiene bucles. Los árboles de decisión son árboles binarios, lo que significa que todos los nodos internos tienen dos vértices salientes.

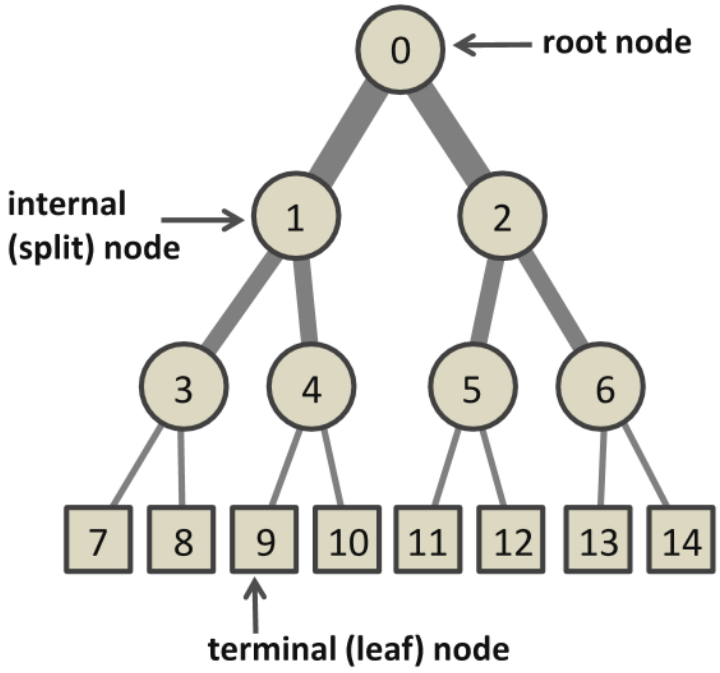

Figura 3: Estructura general de un árbol binario

Para una determinada muestra de entrada un árbol de decisión estima una propiedad desconocida de dicha muestra preguntando sucesivas cuestiones sobre sus propiedades conocidas. Que cuestión preguntar en cada instante depende de las repuestas que se hayan dado anteriormente. Esta jerarquía de preguntas es representada de un modo gráfico como un camino a través del árbol que la muestra de entrada sigue. La decisión se toma a partir

del nodo terminal alcanzado por la muestra de entrada. La Figura 4 muestra un ejemplo del funcionamiento de un árbol de decisión.

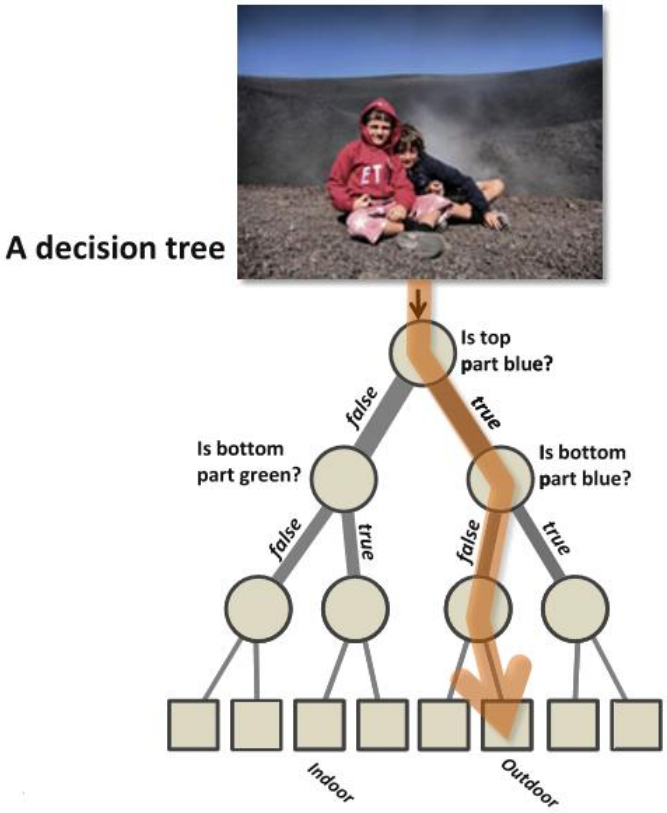

Figura 4: Ejemplo de árbol de decisión

Las preguntas o test que se realizan a lo largo del árbol son aprendidas durante la fase de entrenamiento del árbol como funciones que tendrán como salida respuestas binarias. Cada uno de los nodos no terminales del árbol tiene asociada una de estas funciones. Por tanto, cada nodo dividirá las muestras de entada que le lleguen en 2 subgrupos. Como se puede ver en la Figura 5, las fronteras establecidas por estas funciones pueden ser lineales o no lineales.

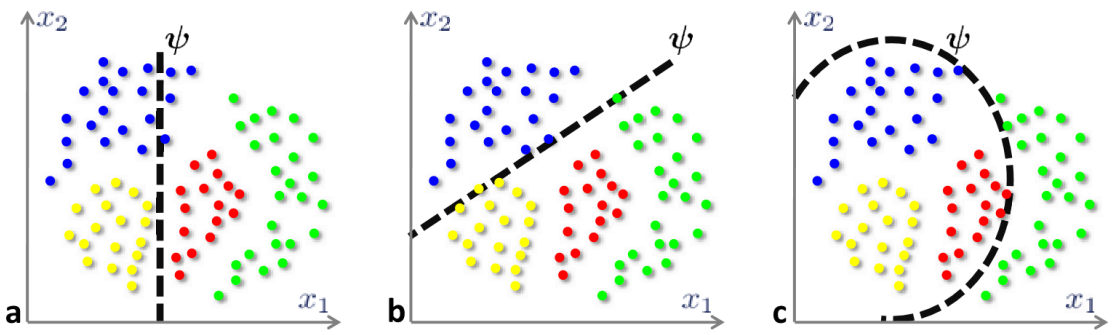

Figura 5: Diferentes funciones frontera

2.1.1.2 Bosques de decisión

Un Bosque de Decisión es un conjunto de árboles de decisión inicializados de forma aleatoria. El aspecto fundamental de esta técnica es que los árboles que forman los bosques son aleatoriamente diferentes entre si, lo cual lleva a una de-correlación entre las predicciones de cada uno de los árboles, y, por tanto, mejora la generalización y la robustez del sistema.

En los Bosques de decisión cada uno de los árboles es entrenado independientemente, y en paralelo a ser posible. Durante la fase de test, cada una de las muestras de entrada es simultáneamente introducida a través de todos los árboles que forman el bosque hasta que llegan a sus correspondientes nodos terminales.

$$p(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|\mathbf{v})$$

Ecuación 1: Combinación de las predicciones en un bosque de decisión mediante promediado

Combinar las predicciones de los distintos árboles puede hacerse con un simple promediado como el de la Ecuación 1 donde $\mathbf{p_t(c|v)}$ representa la distribución de probabilidad a posteriori obtenida por el t-ésimo árbol. Aunque multiplicar las salidas de los árboles también es una opción como se puede observar en la Ecuación 2 siempre y cuando se utilice el factor **Z** para normalizar.

$$p(c|\mathbf{v}) = \frac{1}{Z}\prod_{t=1}^{T} p_t(c|\mathbf{v})$$

Ecuación 2: Combinación de las predicciones en un bosque de decisión mediante productorio

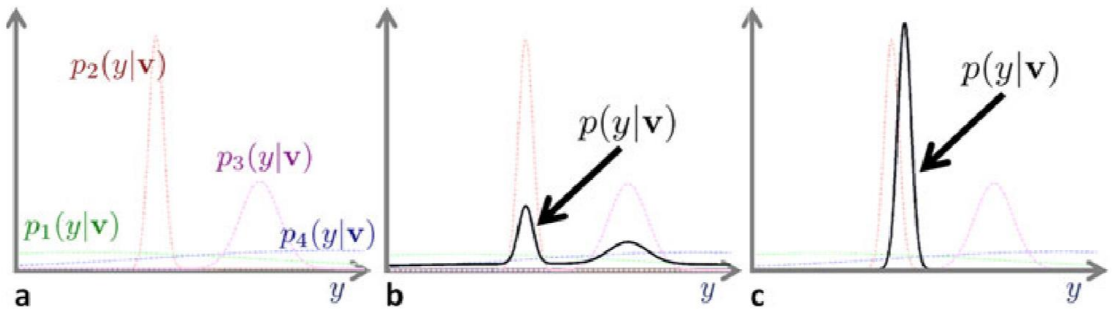

Figura 6: Predicciones en un bosque de decisión

En la Figura 6 se describe un ejemplo del funcionamiento de un bosque de decisión. En la gráfica **a)** se representan las probabilidades a posteriori de cuatro árboles diferentes para predecir la variable **y** dada la entrada **v**. En la gráfica **b)** se representa la probabilidad a posteriori del bosque si se utilizase el promediado como función de combinación de la probabilidad de los árboles. En la gráfica **c)** se representa la probabilidad a posteriori del bosque utilizando la función productorio.

2.1.1.3 Bosques de Clasificación

En este apartado se detalla el uso más común de los bosques de decisión, es decir, la clasificación, cuyo objetivo es asociar una determinada muestra de entrada a una clase concreta. Los bosques de decisión tienen las siguientes propiedades:

- Buen desempeño en problemas con más de dos clases.
- Proporcionan una salida probabilística.
- Generalizan bien para muestras que no se han visto durante el entrenamiento.
- Son eficientes gracias al número de pequeños tests o cuestiones que se le aplican a cada muestra y gracias a que pueden ser ejecutados en paralelo.

La tarea de clasificación puede definirse del siguiente modo: Dado un dataset etiquetado de muestras de entrenamiento debe aprenderse un modelo que asocie correctamente datos que no se han visto en el entrenamiento con sus correspondientes clases.

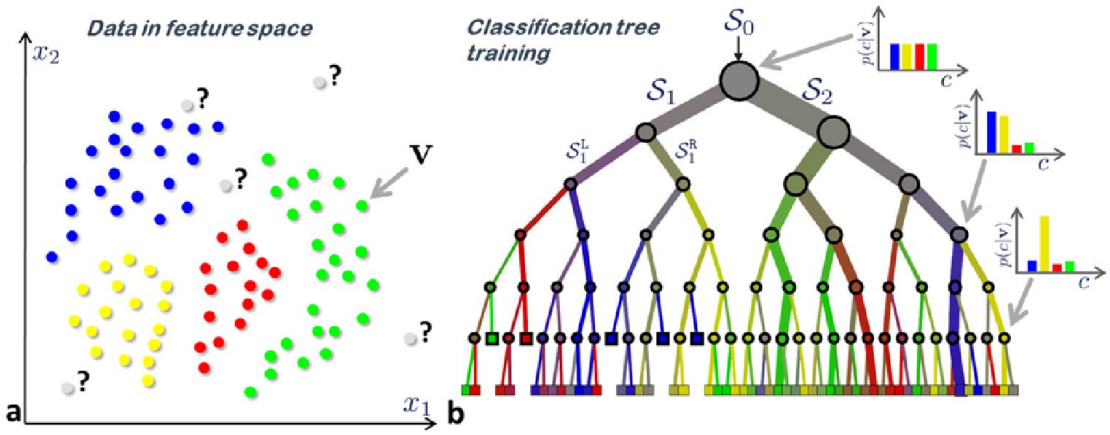

Figura 7: Árbol de clasificación

En los problemas de clasificación la salida de los modelos entrenados es discreta, por tanto, las etiquetas de entrenamiento también lo son. Cada muestra de entrenamiento es definida por un vector de características conocidas y una etiqueta de clase.

En la Figura 7 las muestras de entrenamiento son los diferentes puntos que se encuentran en la gráfica **a)**, donde cada color representa una clase diferente. Las muestras de test (las

cuales no se utilizan para entrenar el modelo **bajo ningún concepto**) son mostradas en color gris (la clase a la que pertenecen no es conocida de antemano por el modelo).

Durante la fase la fase de test el modelo recibe una muestra de entrada y el objetivo es inferir de manera correcta la clase a la que pertenece. Esto se lleva a cabo mediante el cálculo de la probabilidad a posteriori **argmax p(c|v)**, es decir, la clase asignada a la muestra de entrada será a la que el sistema le haya otorgado una mayor probabilidad a posteriori dada la muestra de entrada.

2.1.1.4 Bosques de regresión

Este apartado detalla la utilización de los árboles de decisión para la estimación probabilística de variables continuas. Los Bosques de Regresión son utilizados para la regresión no lineal de variables dependientes dada una entrada independiente, donde tanto las entradas como las salidas deben ser multidimensionales.

Los Bosques de Regresión están relaciones con los de clasificación (aunque gozan de menos popularidad), debido a que comparten algunas de sus ventajas como la eficiencia y la flexibilidad. La diferencia principal entre regresión y clasificación es que las etiquetas de salida asociadas con las muestras de entrada son continuas, por tanto, las etiquetas de entrenamiento también deben ser continuas.

La tarea de regresión puede definirse del siguiente modo: Dada un conjunto de datos de entrenamiento etiquetado se debe entrenar un modelo capaz de asociar correctamente datos de test independientes con su predicción de salida continua dependiente correspondiente.

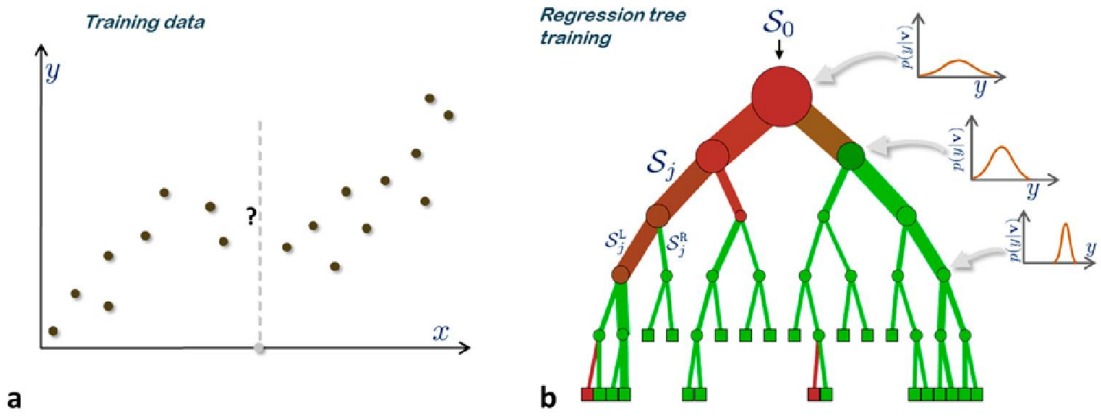

Figura 8: Árbol de Regresión

En la Figura 8 se muestra un ejemplo ilustrativo de datos de entrenamiento para una regresión y su árbol correspondiente. En este caso los datos de entrenamiento son unidimensionales y están representados por **x**, y su etiqueta continua es **y**.

2.1.2 Utilización de Bosques de Clasificación en detección automática de órganos [7]

El método más popular de detección de órganos en adquisiciones de TC que se puede encontrar en la bibliografía es sin duda [7]. En este artículo, que fue publicado en 2009, se presenta un algoritmo probabilístico para el análisis automático de imágenes médicas en 3D (TC).

Dicho algoritmo está basado en Bosques de Clasificación enriquecidos con características visuales aprendidas durante la fase de entrenamiento. Cabe destacar que el sistema del artículo está entrenado para detectar órganos, pero podría entrenarse de igual modo para detectar anomalías (enfermedades o comportamientos extraños). Además, la salida del algoritmo es probabilística, por tanto, es posible modelar la incertidumbre, así como la fusión de múltiples fuentes de información (fusión multimodal). Los órganos que el algoritmo ha sido entrenado para detectar son los siguientes:

- Corazón
- Ojo izquierdo y derecho
- Pulmón izquierdo y derecho
- Riñón izquierdo y derecho
- Hígado
- Cabeza

El dataset con el que el algoritmo ha sido entrenado consta de 39 TC etiquetadas con Bounding Box 3D como se puede ver en la Figura 9.

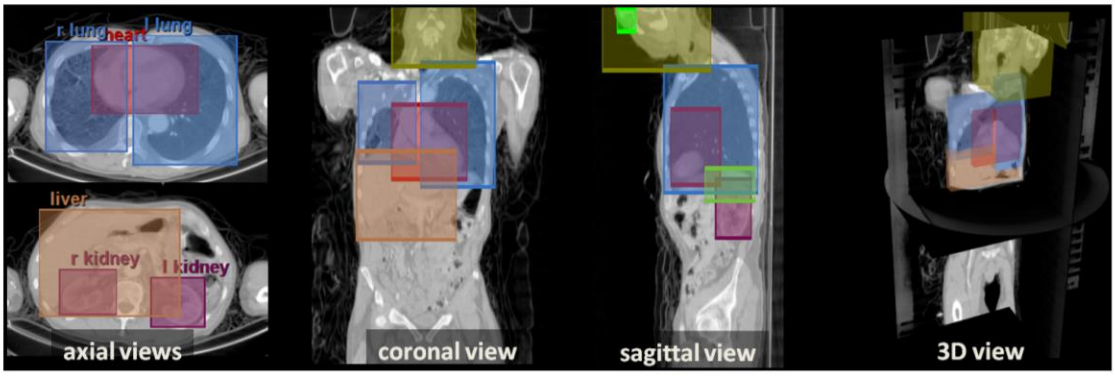

Figura 9: Muestra del dataset etiquetada

El objetivo del algoritmo es encontrar el centro de cada órgano en TCs que no han sido analizadas anteriormente por el sistema. Para ello se ha entrenado un modelo con Bosques de Clasificación y un set de muestras positivas y negativas de centroides obtenido a partir del dataset de 39 TCs etiquetadas que se ha explicado anteriormente.

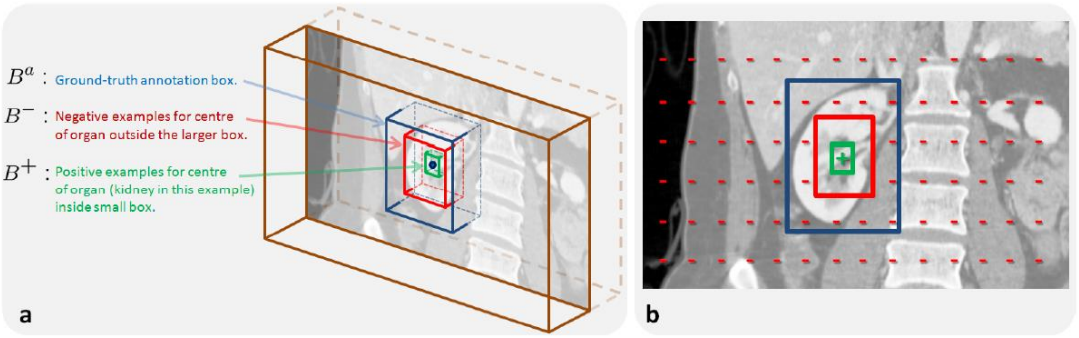

Figura 10: Obtención de muestras positivas y negativas

En la Figura 10 se muestra el proceso de obtención de muestras positivas y negativas del centroide de un órgano determinado, donde a partir de las etiquetas manuales de cada órgano (rectángulo azul) se obtienen subáreas (rectángulo rojo y verde) para identificar que regiones de la TC son tomada como muestras de entrenamientos positivas y cuales negativas. En este caso, los vóxeles (píxeles en 3D) dentro del rectángulo verde son tomados como muestras positivas, y, los que están fuera del rectángulo rojo son tomados como muestras negativas. Cabe destacar que los píxeles que quedan en la región comprendida entre el rectángulo rojo y el verde son ignorados para evitar falsos positivos.

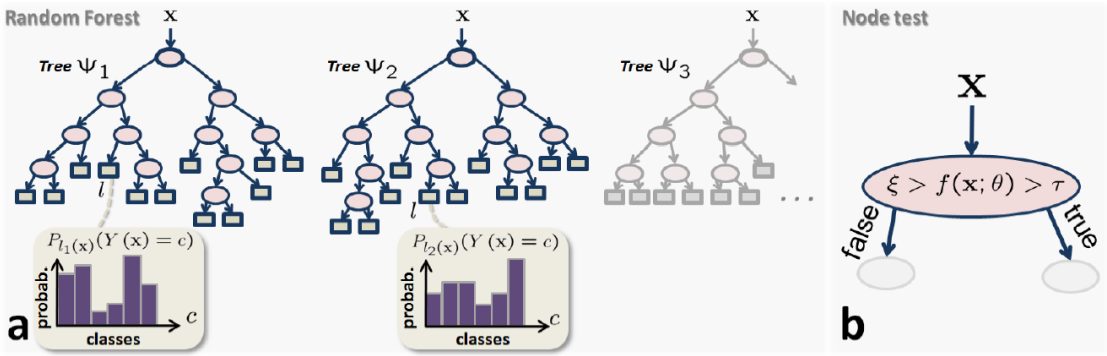

Figura 11: Bosque de decisión

En la Figura 11 se muestra el funcionamiento del modelo creado por el Bosque de Clasificación una vez entrenado, donde la salida de cada nodo terminal es la probabilidad de los píxeles que han llegado hasta dicho nodo de formar parte del centroide de cada uno de los órganos.

En cada uno de los nodos intermedios de los árboles se analizan características visuales de los vóxeles para subdividirlos convenientemente en los siguientes nodos. La obtención de estas características puede verse en la Figura 12.

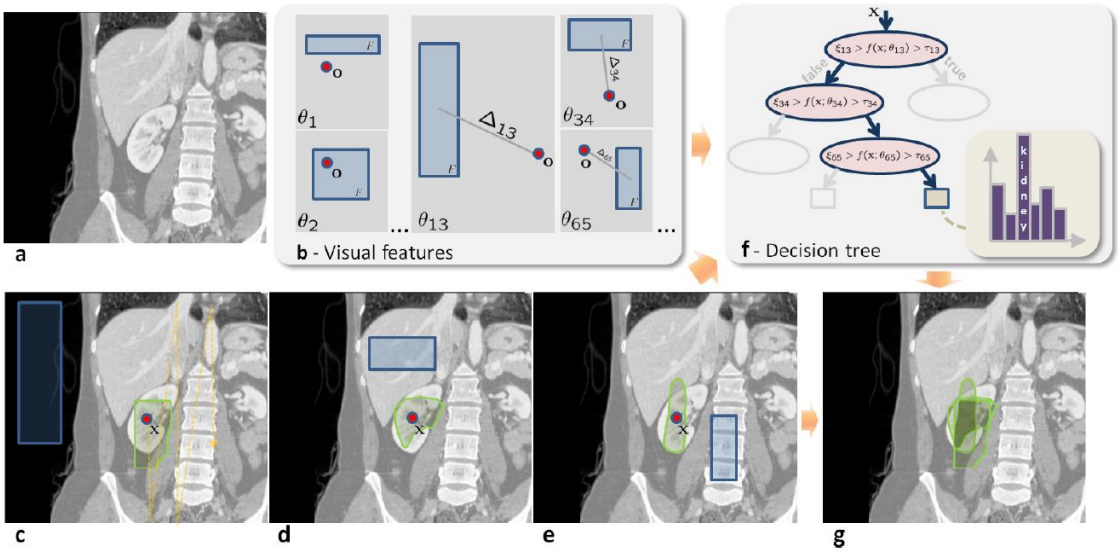

Figura 12: Características Visuales

Las características visuales son aprendidas durante en el entrenamiento para cada nodo, y no son más que dos Bounding Boxes 3D a una distancia determinada del vóxel que se está evaluando. Tanto las dimensiones de las Bounding Boxes como la distancia a la que están del vóxel que se evalúa son parámetro aprendidos en el entrenamiento para cada uno de los nodos. Por tanto, la función que desempeña cada nodo de cada uno de los árboles que forman el bosque es la evaluación de dos regiones que están alrededor de los vóxeles que llegan al nodo, y en función de estas regiones los vóxeles recorrerán el árbol convenientemente.

2.1.3 Utilización de Bosques de regresión en detección automática de órganos [5, 7]

La implementación más popular de los árboles de regresión para la detección automática de órganos también fue propuesta por Antonio Criminisi en el año 2011 en un artículo llamado 'Regression Forests for Efficient Anatomy Detection and Localization in CT Studies'. En dicho artículo se propone un algoritmo que analiza tomografías computarizadas en 3D para detectar estructuras relevantes mediante bosques de regresión entrenados previamente.

Como se ha comentado anteriormente, los árboles de regresión son similares a los árboles de clasificación, con la diferencia de que los árboles de regresión están entrenados para predecir salidas continuas. En el algoritmo de Criminisi se introduce un nuevo método de parametrización continua de la tarea de localización automática de diferentes estructuras de la anatomía (órganos en este caso).

Para el entrenamiento del algoritmo propuesto se han utilizado 100 tomografías [7], en las cuales se han etiquetado mediante Bounding Boxes 3D (como en el algoritmo de los árboles de clasificación) los órganos siguientes:

- Corazón
- Hígado
- Bazo
- Pulmón derecho e Izquierdo
- Riñón derecho e izquierdo
- Vesícula biliar
- Pelvis izquierda y Pelvis derecha

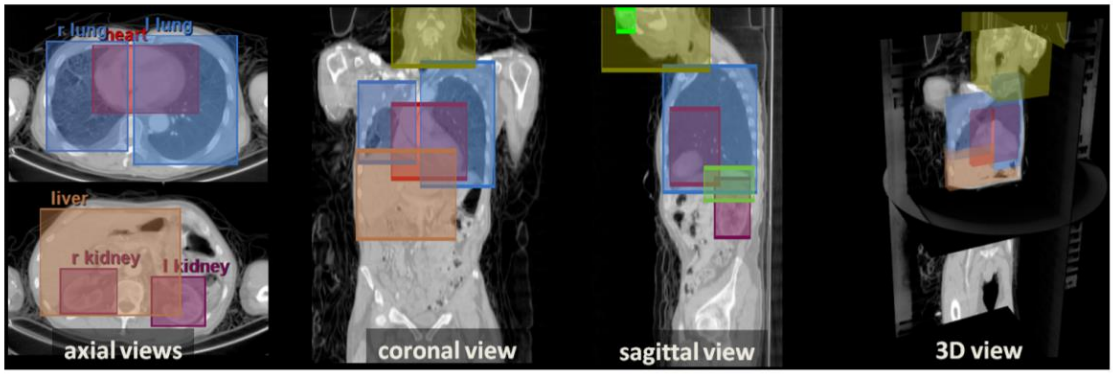

Figura 13: Muestra de training etiquetada

Una de las claves del algoritmo propuesto por Criminisi es que todos los vóxeles (píxeles tridimensionales) de la tomografía contribuyen con un determinado peso a estimar la posición de cada una de las seis paredes de las diferentes Bounding Boxes que determinan la posición de los órganos.

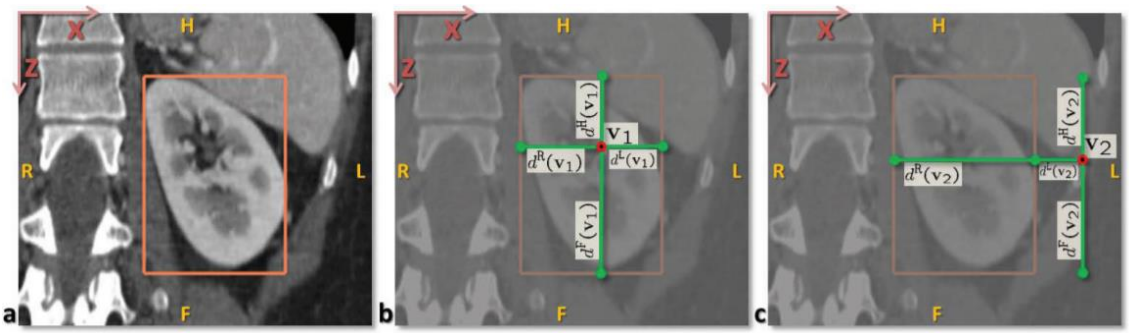

Figura 14: Contribución de un vóxel para estimar las paredes del hígado

Como se puede ver en la Figura 14 distintos vóxeles estiman a que distancia de ellos se encuentran las diferentes paredes de los órganos para contribuir a la predicción final. Por tanto, la predicción de salida del sistema será un vector de 6 dimensiones por cada órgano, donde cada una estas dimensiones indica la posición de una de las 6 paredes de la Bounding Box 3D que estima la posición y el tamaño del órgano.

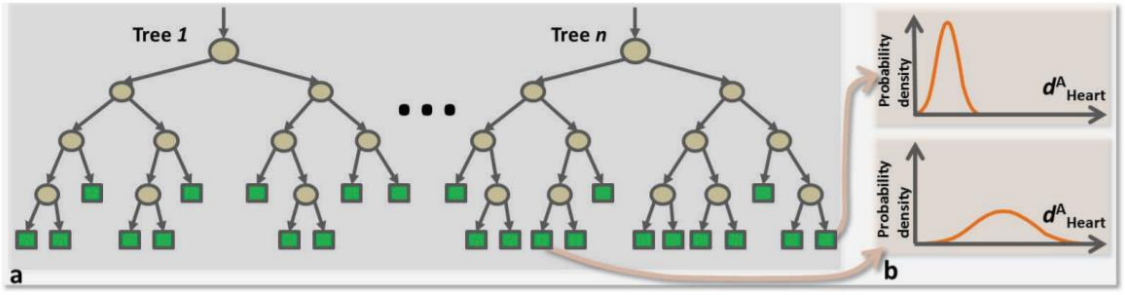

Figura 15: Predicción continua del bosque de regresión

En la Figura 15 se puede observar la salida continua del bosque de regresión, donde d^A_{heart} es la distancia de un determinado vóxel a una de las paredes del corazón.

2- Redes Neuronales

2.2.1 Explicación de la técnica

Las redes neuronales son un modelo computacional basado en un gran conjunto de unidades neuronales simples (neuronas artificiales) que buscan emular de forma aproximadamente análoga el comportamiento observado en los axones de las neuronas en los cerebros biológicos. Cada unidad neuronal está conectada con muchas otras y los enlaces entre ellas pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. Cada unidad neuronal, de forma individual, opera empleando funciones de suma. Puede existir una función limitadora o umbral en cada conexión y en la propia unidad, de tal modo que la señal debe sobrepasar un límite antes de propagarse a otra neurona. Estos sistemas aprenden y se forman a sí mismos, en lugar de ser programados de forma explícita, y sobresalen en áreas donde la detección de soluciones o características es difícil de expresar con la programación convencional.

El objetivo de la red neuronal es resolver los problemas de la misma manera que el cerebro humano. Aunque los proyectos de redes neurales modernas suelen trabajar con unos pocos miles a unos pocos millones de unidades neuronales y millones de conexiones, que sigue siendo varios órdenes de magnitud menos complejo que el cerebro humano.

Un aspecto relevante de estos sistemas es que son impredecibles en su éxito con el auto-aprendizaje. Después del entrenamiento, algunos se convierten en grandes solucionadores de problemas y otros no funcionan tan bien. Con el fin de capacitarlos, se necesitan varios miles de ciclos de iteración.

Las redes neuronales se han utilizado para resolver una amplia variedad de tareas, como la visión por computador y el reconocimiento de voz, que son difíciles de resolver usando la programación ordinaria basada en reglas, por tanto, pueden ser un método muy a tener en cuenta para intentar resolver el problema de detección automática de órganos.

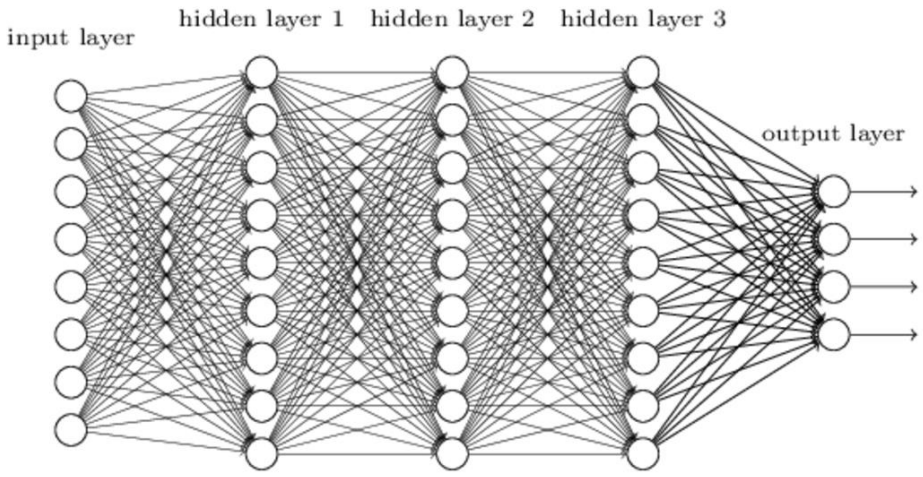

Figura 16: Ejemplo de arquitectura de red neuronal

Entre las ventajas que ofrecen las redes neuronales se encuentran las siguientes:

- Aprendizaje Adaptativo. Capacidad de aprender a realizar tareas basadas en un entrenamiento o en una experiencia inicial.
- Operación en tiempo real. Los cómputos neuronales pueden ser realizados en paralelo; para esto se diseñan y fabrican máquinas con hardware especial para obtener esta capacidad (GPUs).
- Fácil inserción dentro de la tecnología existente. Se pueden obtener chips especializados para redes neuronales que mejoran su capacidad en ciertas tareas. Ello facilitará la integración modular en los sistemas existentes.

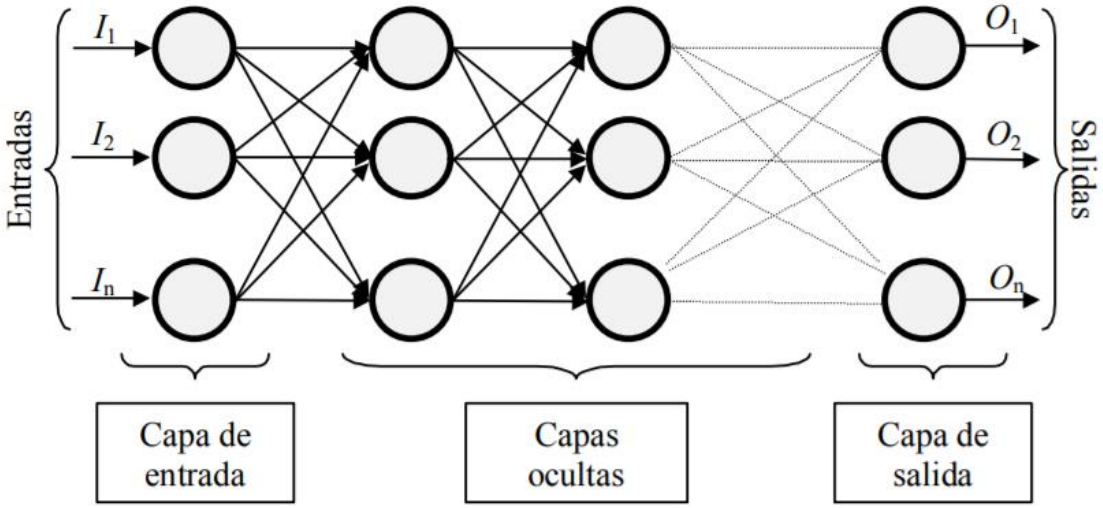

Figura 17: Componentes básicos de una red neuronal

En la Figura 17 se pueden observar los elementos básicos de una red neuronal, los cuales son:

- Capa de entrada: Es la capa por la que se introducen los datos a la red.
- Capas ocultas: Son las capas en las que se combinan operaciones matemáticas con funciones de activación para obtener el resultado final.
- Capa de salida: Es la capa en la que se obtiene el resultado final de la red tras haber introducido unos datos determinados.
- Conexiones: El valor de estas conexiones es lo que se aprende durante el proceso de entrenamiento de la red neuronal.

Una neurona biológica puede estar activa (excitada) o inactiva (no excitada), es decir, que tiene un "estado de activación". Las neuronas artificiales también tienen diferentes estados de activación; algunas de ellas solamente dos, al igual que las biológicas, pero otras pueden tomar cualquier valor dentro de un conjunto determinado. La función de activación es la encargada de determinar el estado de actividad de una neurona artificial a partir de la combinación de las salidas de las neuronas de la capa anterior multiplicadas por los pesos de las conexiones que unen las neuronas anteriores con la actual.

2.2.1.1 Redes Neuronales Convolucionales (CNN)

Una red neuronal convolucional es un tipo de red neuronal artificial donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria de un cerebro biológico. Este tipo de red es una variación de un perceptron multicapa, sin embargo, debido a que su aplicación es realizada en matrices bidimensionales (o tridimensionales en el caso de detección de órganos en TC), son muy efectivas para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones.

Las redes neuronales convolucionales consisten en múltiples capas de filtros convolucionales de una o más dimensiones. Después de cada capa, por lo general se añade una función para realizar un mapeo causal no-lineal (Max-pooling habitualmente). Como redes de clasificación, al principio se encuentra la fase de extracción de características, compuesta de neuronas convolucionales. Al final de la red se encuentran neuronas de perceptron sencillas para realizar la clasificación final sobre las características extraídas. La fase de extracción de características se asemeja al proceso estimulante en las células de la corteza visual. Esta fase se compone de capas alternas de neuronas convolucionales y neuronas de reducción de muestreo. Según progresan los datos a lo largo de esta fase, se disminuye su dimensionalidad, siendo las neuronas en capas lejanas mucho menos sensibles a perturbaciones en los datos de entrada, pero al mismo tiempo siendo estas activadas por características cada vez más complejas.

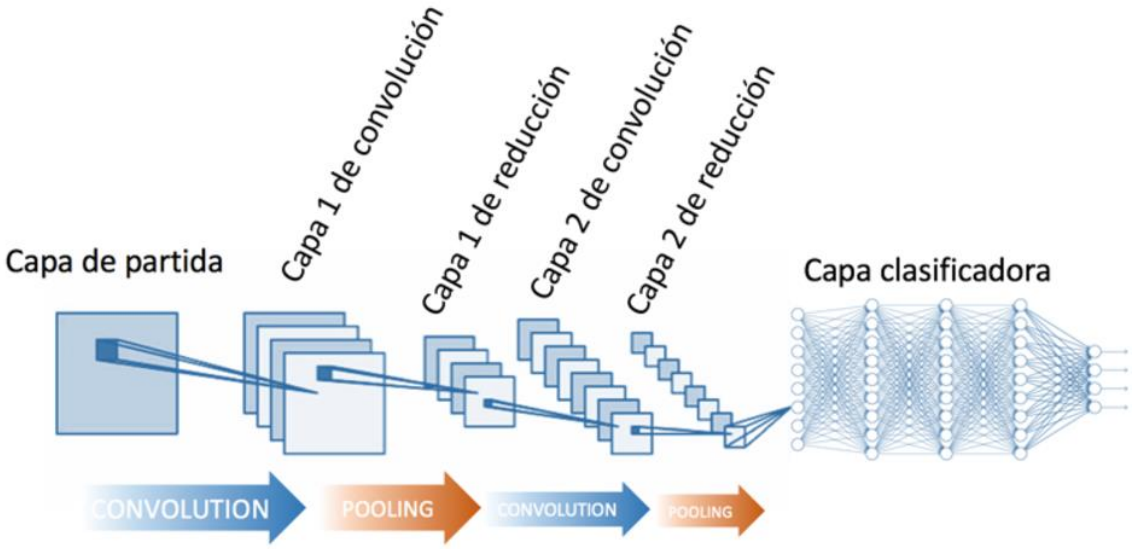

Figura 18: Ejemplo de red neuronal convolucional

Las redes neuronales convolucionales son el tipo de redes neuronales utilizadas para problemas de visión artificial por excelencia debido a los magníficos resultados que están brindando en los últimos años, es por ello, que, a día de hoy, es una de las mejores opciones para afrontar el problema de la detección automática de órganos.

2.2.2 Utilización en detección automática de órganos [14]

Debido a que las redes neuronales han empezado a popularizarle en los últimos años, aún no hay una gran variedad de técnicas que se hayan implementado utilizando este método para resolver la tarea de detección automática de órganos, sin embargo, se pueden encontrar algunas implementaciones. Para el propósito del proyecto que se presenta en esta memoria [14] puede resultar de especial interés.

En el método propuesto en [14] se pretende detectar órganos (hígado, corazón, riñón izquierdo y derecho) en TC mediante Bounding Boxes 3D. Para ello es necesario distinguir las estructuras de los órganos buscados entre ellos mismos, y, además, del resto de estructuras anatómicas del paciente.

Para entrenar el clasificador se utilizó un dataset de 44 TC (30 para training y 14 para test) etiquetados como se puede ver en la Figura 19.

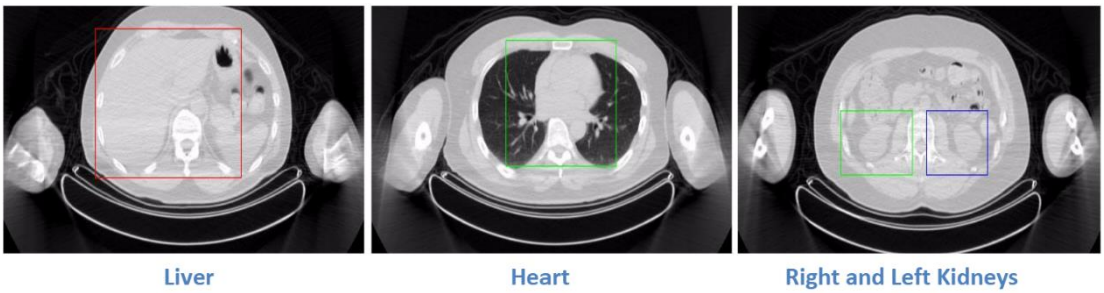

Figura 19: Muestra de entrenamiento etiquetada

Después del etiquetado del dataset, el siguiente paso en el flujo de trabajo que se propone en [14] es dividir cada uno de los cortes de la TC en sub-imágenes para la posterior extracción de características de las mismas como puede verse en la Figura 20.

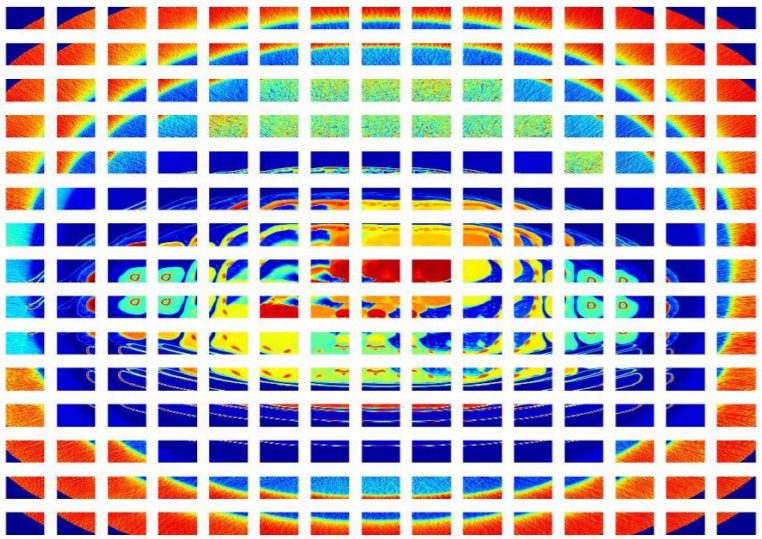

Figura 20: Subdivisión de un corte en sub-imágenes

Estas sub-imágenes son introducidas posteriormente en GoogLeNet, que es una red neuronal convolucional profunda entrenada con millones de imágenes, con el objetivo de extraer características que representen de forma precisa las características de dichas sub-imágenes.

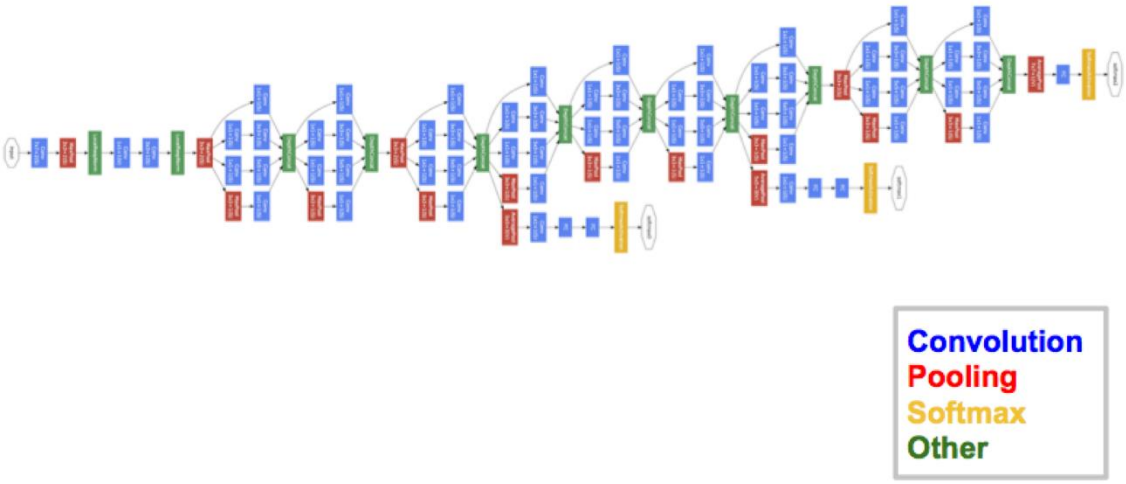

Figura 21: Arquitectura de GoogLeNet

Por último, con las características extraídas de GoogLeNet se entrena un clasificador basado en Supported Vector Machines (método de aprendizaje automático) para clasificar las subimágenes.

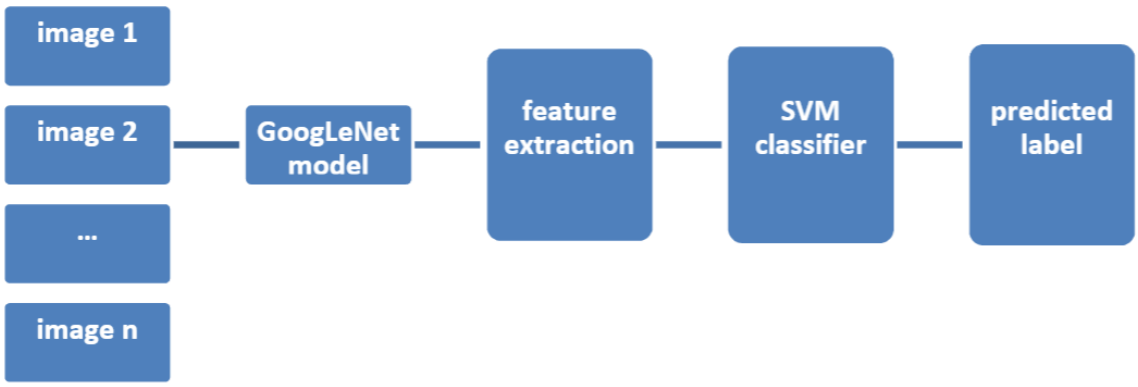

Figura 22: Flujo de trabajo del método propuesto en [14]

Capítulo 3. Análisis del problema y Herramientas utilizadas

1- Elección del método de Machine Learning adecuado

Para llevar a cabo un proyecto utilizando técnicas de Machine Learning hay dos pilares fundamentales que van a afectar de forma drástica a los resultados que se obtengan:

- El tamaño y calidad del **dataset** con el que se van a entrenar los modelos.
- El **algoritmo** (y ajuste de parámetros de dicho algoritmo) con el que se van a entrenar los modelos.

Para realizar el proyecto propuesto se dispone de 200 TC sin etiquetar, por tanto, aparecen dos limitaciones principales. Por un lado, 200 muestras se considera un dataset considerablemente pequeño debido a que habitualmente se suelen entrenar modelos con Machine Learning con miles o incluso millones de muestras. Y, por otro lado, debido a que las TC son imágenes 3D con miles de cortes su etiquetado no es sencillo y puede ser muy costoso a nivel temporal, por tanto, pese a solo disponer de 200 muestras el proceso de etiquetado tiene que ser elegido cautelosamente para que se pueda abarcar en un tiempo realista.

La disponibilidad de solo 200 muestras de TC se debe a que por razones de privacidad y seguridad los hospitales no pueden proporcionar datos de pacientes a gran escala sin pasar por largos procesos burocráticos y de anonimizado de dichos datos, por tanto, el volumen del dataset con el que se trabaja es reducido desde el punto de vista del Machine Learning, pero extenso desde el punto de vista médico, debido a la dificultad de obtención del mismo.

Una vez analizadas las virtudes, defectos y propiedades del dataset del que se dispone es cuando se debe sopesar que técnica de Machine Learning podría ser utilizada para sacarle partido al mismo. En el caso particular del proyecto que se propone para esta memoria se han evaluado los distintos algoritmos detallados en la sección de Estado del Arte y se ha decidido finalmente optar por las redes neuronales convolucionales por los siguientes motivos:

- Con las redes neuronales convolucionales se están obteniendo mejores resultados en la gran mayoría de los problemas de Visión Artificial en la actualidad.
- Se han desarrollado herramientas que facilitan en gran medida su desarrollo e implementación (TensorFlow, Keras, etc).
- Funcionan muy rápido gracias a los últimos avances en chips gráficos.
- Son muy versátiles y se adaptan a una gran variedad de problemas.

Pero, por desgracia, si se pretende entrenar modelos con redes neuronales profundas y complejas nos encontramos con dos desventajas principales ligadas al dataset del que se dispone:

- El dataset tiene muy pocas muestras
- Las TC son datos de una dimensionalidad enorme (más de 200 MB en memoria) por tanto el proceso de análisis puede ser lento.

Para solventar las limitaciones de las redes neuronales a la hora de trabajar con el dataset del que se dispone en este proyecto se propone utilizar redes neuronales convolucionales poco profundas y muy especializadas que solventen las limitaciones detalladas anteriormente del siguiente modo:

- Las redes neuronales poco profundas no necesitan una cantidad tan grande de muestras de entrenamiento para presentar buenos resultados
- Las redes neuronales poco profundas se ejecutan mucho más rápido debido al menor número de operaciones que requieren.

2- Software de etiquetado del dataset

Después de evaluar detenidamente diferentes herramientas diseñadas para trabajar con imágenes médicas en 3D se ha decidido que ITK-SNAP [15] es una herramienta con la que se puede llevar a cabo el etiquetado del dataset con comodidad, soltura y rapidez.

ITK-SNAP es una aplicación software utilizada para segmentar estructuras en imágenes médicas tridimensionales. Es el producto de una década de colaboración de doctores del 'Penn Image Computing and Science Laboratory'(PICSL) y del 'Scientific Computing and Imaging Institute'(SCI), cuyo objetivo era crear una herramienta de segmentación que fuese fácil de utilizar y aprender. ITK-SNAP es gratuito, de código abierto y multiplataforma.

ITK-SNAP proporciona segmentación semi-automática utilizando métodos de contornos activos, así como delineación manual y navegación por las imágenes 3D. Además de estas funcionalidades principales, ITK-SNAP ofrece una gran variedad de herramientas de soporte. Algunas de sus principales ventajas son:

- Interfaz gráfica de usuario moderna basada en Qt
- Soporta diferentes formatos de imagen médica incluyendo DICOM y NifTI
- Gran variedad de tutoriales y extensa documentación
- Segmentación manual en los 3 planos ortogonales (axial, sagital y coronal) al mismo tiempo.

Comparado con otras herramientas de código libre de análisis de imágenes, el diseño de ITK-SNAP está centrado en el problema de segmentación de imágenes, por tanto, todas las funciones que se apartan de este objetivo han sido mantenidas al mínimo. El diseño también enfatiza la interacción y la facilidad de uso.

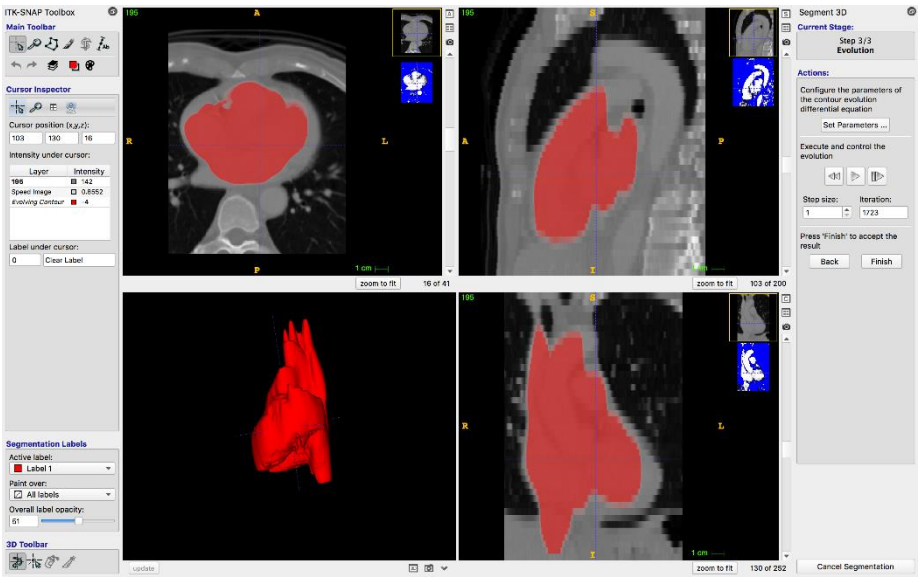

Figura 23: ITK-SNAP

3- Software de desarrollo

3.3.1 Python

El lenguaje de programación elegido para llevar a cabo el presente proyecto ha sido Python debido a que tiene una inmensa variedad de librerías para prácticamente cualquier tarea debido a la gran comunidad de desarrolladores que lo utilizan. Concretamente, la herramienta de TensorFlow (entre otras muchas) desarrollada por Google para el diseño de redes neuronales está desarrollada en Python. Además, Python es un lenguaje de programación muy versátil a la hora de desarrollar código y permite implementar soluciones equivalentes a otros lenguajes de programación (como por ejemplo C++) con menor cantidad de líneas de código.

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

Una característica importante de Python es la resolución dinámica de nombres, es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos). Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

Aunque la programación en Python podría considerarse en algunas situaciones hostil a la programación funcional tradicional del Lisp, existen bastantes analogías entre Python y los lenguajes minimalistas de la familia Lisp como puede ser Scheme.

El intérprete de Python estándar incluye un modo interactivo en el cual se escriben las instrucciones en intérprete de comandos: las expresiones pueden ser introducidas una a una, pudiendo verse el resultado de su evaluación inmediatamente, lo que da la posibilidad de probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa. Esto resulta útil tanto para las personas que se están familiarizando con el lenguaje como para los programadores más avanzados.

Python fue diseñado para ser leído con facilidad. Una de sus características es el uso de palabras donde otros lenguajes utilizarían símbolos. Por ejemplo, los operadores lógicos !, || y && en Python se escriben not, or y and, respectivamente.

Función factorial en C (indentación opcional)

```c
int factorial(int x)
{
    if (x < 0 || x % 1 != 0) {
        printf("x debe ser un numero entero
mayor o igual a 0");
        return -1; //Error
    }
    if (x == 0) {
        return 1;
    }
    return x * factorial(x - 1);
}
```

Función factorial en Python (indentación obligatoria)

```python
def factorial(x):
    assert x >= 0 and x % 1 == 0, "x debe ser
un entero mayor o igual a 0."
    if x == 0:
        return 1
    else:
        return x * factorial(x - 1)
```

Figura 24: Sintaxis de C vs Python

El contenido de los bloques de código (bucles, funciones, clases, etc.) es delimitado mediante espacios o tabuladores, conocidos como indentación, antes de cada línea de órdenes pertenecientes al bloque. Python se diferencia así de otros lenguajes de programación que mantienen como costumbre declarar los bloques mediante un conjunto de caracteres, normalmente entre llaves {}. Se pueden utilizar tanto espacios como tabuladores para indentar el código, pero se recomienda no mezclarlos.

Python tiene una gran biblioteca estándar, usada para una diversidad de tareas. Esto viene de la filosofía "pilas incluidas" ("batteries included") en referencia a los módulos de Python. Los módulos de la biblioteca estándar pueden mejorarse por módulos personalizados escritos tanto en C como en Python. Debido a la gran variedad de herramientas incluidas en la biblioteca estándar, combinada con la habilidad de usar lenguajes de bajo nivel como C y C++, los cuales son capaces de interactuar con otras bibliotecas, Python es un lenguaje que combina su clara sintaxis con el inmenso poder de lenguajes menos elegantes.

Figura 25: Logo de Python

3.3.1.1 Jupyter Notebook

Para escribir una parte del código se ha utiliza un editor de Python llamado Jupyter notebook, el cual permite desarrollar bloques de código de una manera muy cómoda además de permitir comentar el código con una gran variedad de tipografías, tamaños de texto, colores e incluso imágenes.

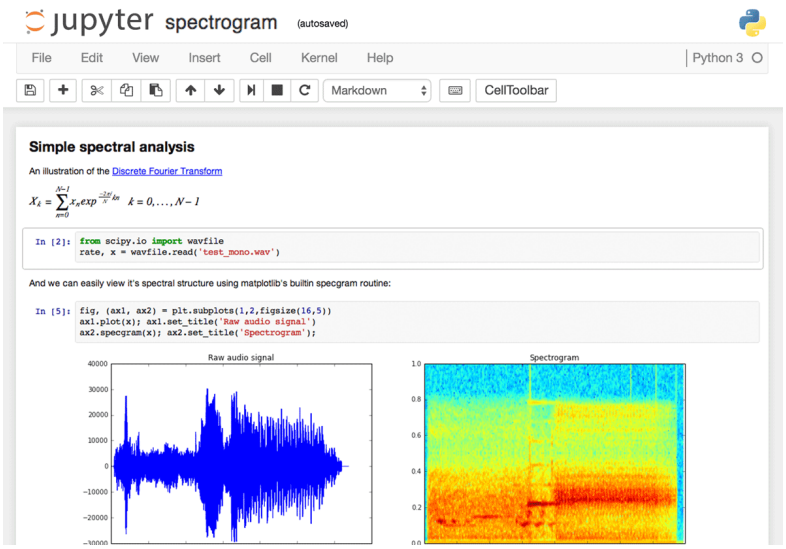

Figura 26: Ejemplo de uso de Jupyter Notebook

Jupyter Notebook es una aplicación de código libre que permite crear y compartir documentos de código que se puede ejecutar por bloques, ecuaciones, visualizaciones y texto explicativo. Entre los usos más comunes de Jupyter Notebook se encuentran:

- Procesado de datos
- Simulaciones numéricas
- Modelado estadístico
- Machine Learning

Figura 27: Jupyter Notebook Logo

3.3.1.2 Spyder

Aunque Jupyter notebook es muy cómodo para ciertas tareas, en ocasiones carece de las prestaciones de un entorno de desarrollo al uso, como por ejemplo un entorno de variables. Es por este motivo que en el desarrollo del presente proyecto también ha decidido utilizarse el entorno de desarrollo para Python Spyder.

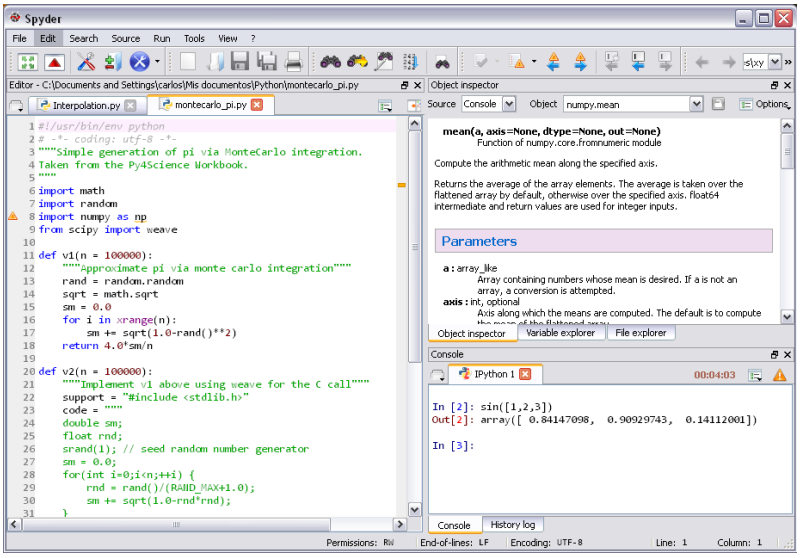

Figura 28: Ejemplo de uso de Spyder

Spyder es un entorno integrado de desarrollo para Python. Tiene capacidad multiplataforma y está diseñado para la programación científica. Integra librerías de Python útiles para la investigación como:

- NumPy
- SciPy
- Matplotlib
- IPython

Se basa en la plataforma Qt ya sea de forma directa o a través de las librerías PyQt o PySide. A partir de la versión 2.3 da soporte tanto a la versión 2.7 de Python como a las nuevas versiones como la 3.5.

Figura 29: Logo de Spyder

3.3.1.3 Librerías

En esta sección se van a detallar las principales librerías de Python utilizadas para llevar a cabo el proyecto.

3.3.1.3.1 NiBabel

NiBabel proporciona acceso de lectura y escritura a ficheros en los formatos más comunes utilizados en imagen médica, centrándose principalmente en NifTI. Mediante esta librería se ha cargado el dataset etiquetado en memoria para su posterior procesado.

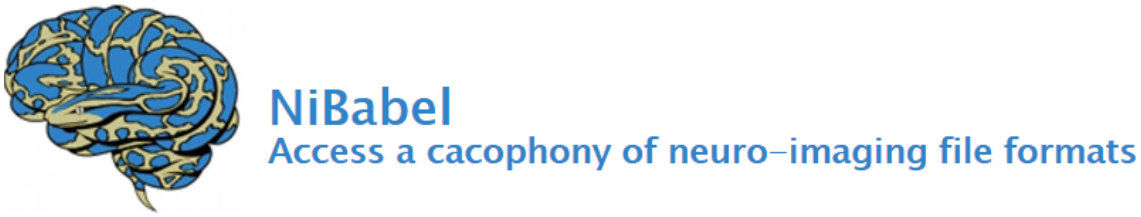

Figura 30: Logo de NiBabel

3.3.1.3.2 SciPy.NDimage

NDimage es un paquete de funciones que se encuentra dentro de la famosa librería de Python para aplicaciones científicas SciPy. NDimage contiene funciones para procesar imágenes multimensionales, es decir, de más de 2 dimensiones. Debido a que las tomografías computarizadas son en 3D esta librería es fundamental en el procesado del dataset para el posterior entrenamiento mediante redes neuronales.

Figura 31: Logo de SciPy

3.3.1.3.3 TensorFlow

TensorFlow es una de las librerías más populares de redes neuronales y, de hecho, se está convirtiendo en el estándar de la industria para tal efecto. TensorFlow es utilizado en el presente proyecto para la gestión de redes neuronales y su pertinente ejecución en la tarjeta gráfica para acelerar la computación en gran medida.

TensorFlow es una librería de código abierto para aprendizaje automático y es desarrollado Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos. Actualmente es utilizado tanto para la investigación como para la producción de productos de Google. TensorFlow fue originalmente desarrollado por el equipo de Google Brain para uso interno en Google antes de ser publicado bajo la licencia de código abierto Apache 2.0 el 9 de noviembre de 2015.

TensorFlow es el sistema de aprendizaje automático de segunda generación de Google Brain. Puede correr en múltiple CPUs y GPUs (con extensiones opcionales de CUDA para informática de propósito general en unidades de procesamiento gráfico). TensorFlow está disponible en Windows, Linux de 64 bits, macOS, y plataformas móviles que incluyen Android e iOS.

Los cómputos de TensorFlow están expresados como 'stateful dataflow graphs'. El nombre TensorFlow deriva de las operaciones que las redes neuronales realizan sobre arrays multidimensionales de datos. Estos arrays multidimensionales son referidos como "tensores".

Entre las aplicaciones para las cuales TensorFlow es la base, está el software automatizado de procesamiento de imágenes 'DeepDream'. Google oficialmente implementó 'RankBrain' el 26 de octubre de 2015, respaldado por TensorFlow. 'RankBrain' ahora maneja un número sustancial de consultas de búsqueda, reemplazando y sustituyendo el algoritmo estático tradicional basado en resultados de búsqueda.

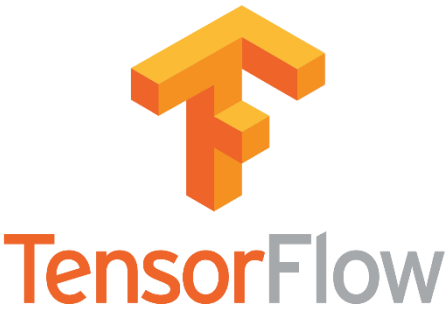

Figura 32: Logo de TensorFlow

3.3.1.3.4 Keras

Debido a que TensorFlow tiene una curva de aprendizaje y complejidad altas se han desarrollado librerías de más alto nivel como Keras, la cual permite diseñar arquitecturas de redes neuronales de un modo mucho más sencillo y con menos código que TensorFlow.

Keras es una librería de redes neuronales de código abierto escrita en Python. Puede ejecutarse por encima de diferentes frameworks de redes neuronales, los cuales son:

- TensorFlow (el framework utilizado en este proyecto)
- MXNet
- Deeplearning4j
- CNTK
- Theano

Keras es una librería diseñada para una experimentación rápida con redes neuronales, por ello el código con Keras es minimalista, modular y extensible. Fue desarrollada como parte de la investigación del proyecto ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), y el principal encargado de su mantenimiento es el ingeniero de Google François Chollet.

En 2017, el equipo de TensorFlow decidió dar soporte a Keras en la librería matriz de TensorFlow. Chollet explicó que Keras fue diseñada para ser una interfaz en lugar de una librería de machine learning end-to-end, debido a que incorpora abstracciones de más alto nivel que hace más sencillo diseñar redes neuronales al contrario que TensorFlow.

Figura 33: Logo de Keras

Capítulo 4. Desarrollo y resultados del trabajo

A lo largo de este capítulo se van a describir los pasos que se han llevado a cabo para desarrollar el proyecto propuesto en esta memoria. Los cuales son:

- Elaboración del dataset
- Diseño del clasificador
- Implementación del clasificador en una ventana deslizante
- Experimentación

1- Elaboración del dataset

4.1.1 Etiquetado de las Tomografías Computarizadas

Como se ha comentado a lo largo de la presente memoria, se dispone de 200 tomografías computarizadas para crear un dataset que permita entrenar un clasificador mediante técnicas de machine learning. Y, como en todo flujo de trabajo de aprendizaje supervisado, el primer paso es analizar los datos de los que se dispone y etiquetarlos debidamente.

En este caso los datos de los que se dispone son tomografías computarizadas, las cuales son imágenes tridimensionales, por tanto, para poder trabajar con dichas imágenes se necesitan herramientas específicas que faciliten trabajar con las mismas. En este caso, como se ha mencionado en la memoria, se ha decidido utilizar ITK-SNAP para visualizar y etiquetar las imágenes.

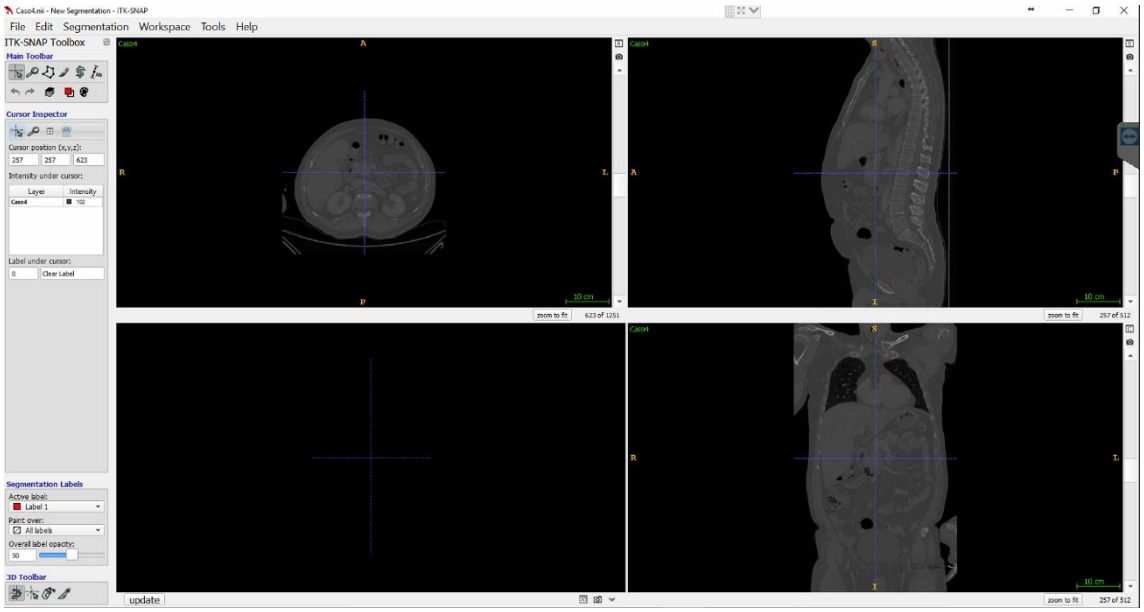

Figura 34: Tomografía computarizada cargada con ITK-SNAP

En la Figura 34 se puede observar una tomografía computarizada cargada en ITK-SNAP, el cual nos permite navegar a través de los distintos cortes de los 3 planos ortogonales (coronal, axil, sagital) de la tomografía además de segmentar diferentes volúmenes de la misma. Otra función útil de ITK-SNAP es que nos permite saber el número de cortes de la tomografía en cada uno de los ejes ortogonales. En este caso nos encontramos con una tomografía de 1251x512x512, es decir, la tomografía es una matriz 3D con 327.942.144 de componentes. Como se puede apreciar, semejante número de dimensiones es muy complejo de abarcar, especialmente con solo 200 muestras, por ello, tanto la fase de etiquetado como la de procesado del dataset deben centrarse en reducir esta dimensionalidad.

Una vez fijado el objetivo de reducir la dimensionalidad, se ha decidido diseñar un clasificador de imágenes que recorra la tomografía con una ventana deslizante 3D, y, para cada sub-ventana el clasificador debe estemir la probabilidad de que el contenido de que esa ventana contenga uno de los órganos con los que se ha entrenado el sistema o si, por el contrario, no es ninguno de esos órganos.

Para entrenar el clasificador descrito en el párrafo anterior nuestro dataset tiene que estar compuesto por ventanas 3D que contengan los órganos con los que se quiere entrenar el sistema y además un conjunto 'background' que contenga unas muestras lo suficientemente representativas de todas las estructuras del cuerpo que no sean dichos órganos.

Etiquetar con ventanas 3D es un proceso lento debido a que lleva varios minutos por cada órgano que se quiere etiquetar en cada muestra, por tanto, para aliviar la carga temporal de etiquetado se ha decidido etiquetar 100 TC para crear el dataset de entrenamiento y dejar las otras 100 tomografías para la fase de validación del sistema. Además, solo se van a etiquetar 3 estructuras, las cuales son:

- Corazón
- Hígado
- Background

Etiquetar estas 3 estructuras en 100 TC es una tarea abarcable para el proyecto que se pretende realizar, por tanto, el siguiente paso es etiquetar el dataset como se muestra a continuación.

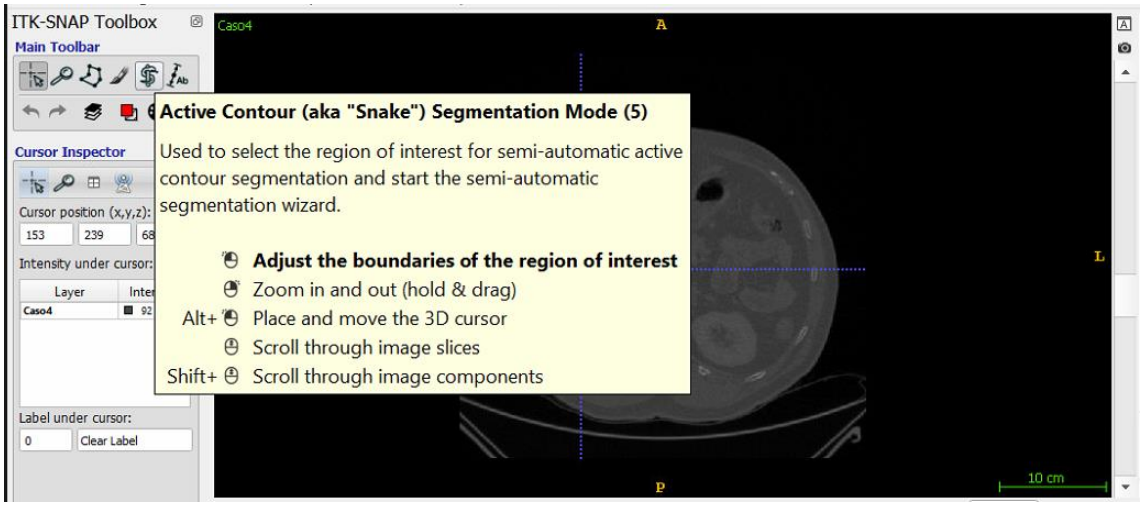

Figura 35: Herramienta de segmentación de ITK-SNAP

Para etiquetar los órganos utilizaremos el selector de regiones de interés como se puede ver en la Figura 35, lo cual nos permite crear las Bounding Boxes 3D que los contengan como se puede ver en las Figuras 36, 37 y 38.

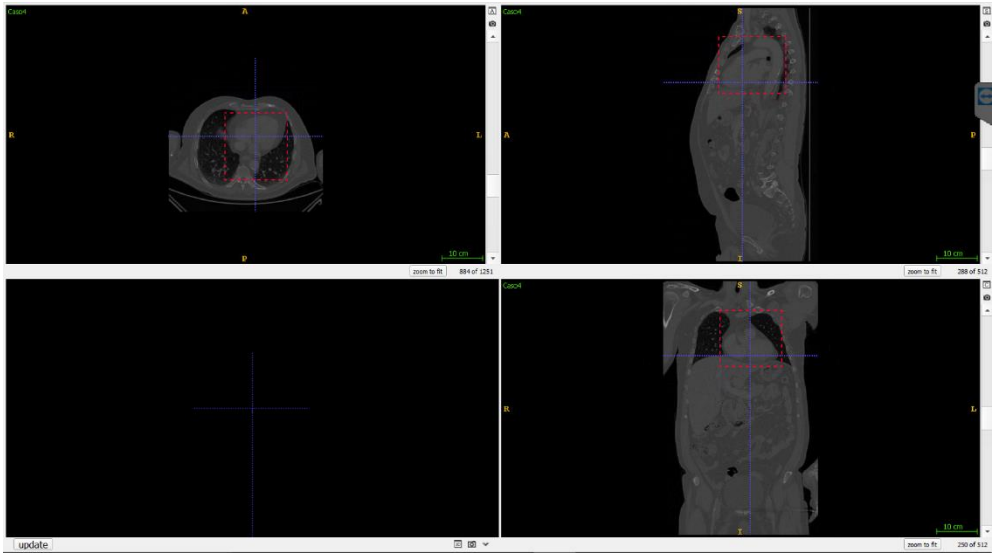

Figura 36: Bounding Box 3D para un corazón

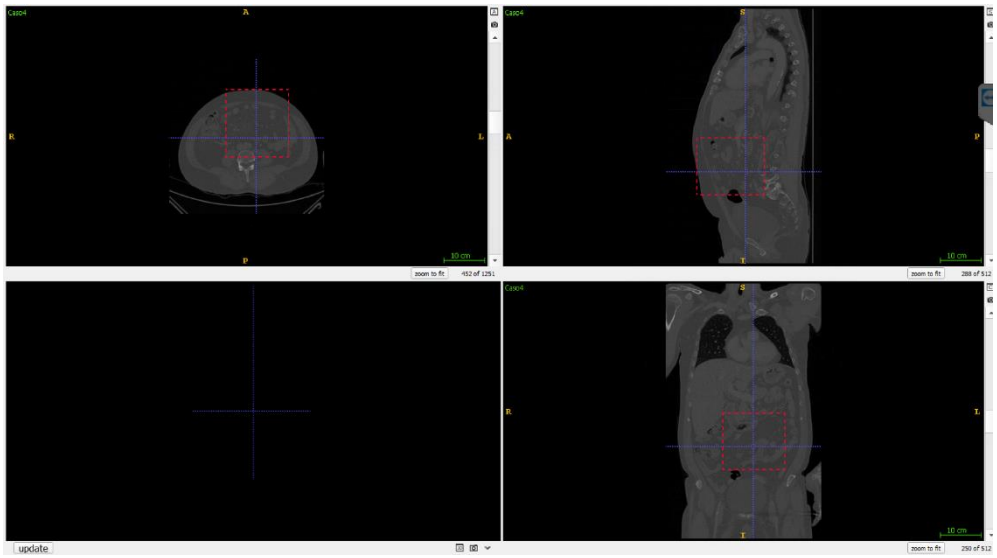

Figura 37: Bounding Box 3D para un área aleatoria de background

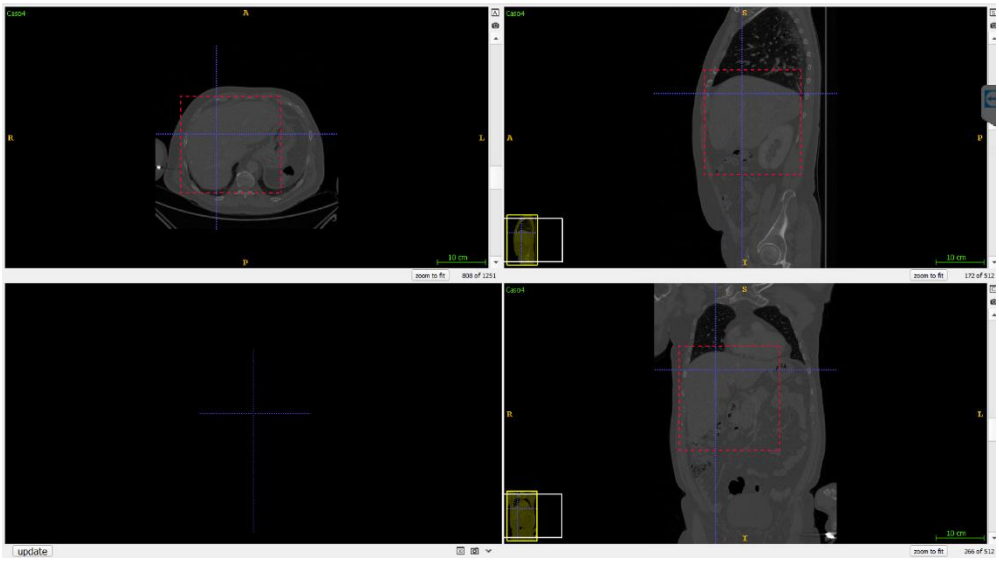

Figura 38: Bounding Box 3D para un hígado

Una vez seleccionadas las regiones de interés el siguiente paso es segmentar dicha región y guardarla como se puede ver en las Figuras 39, 40, 41.

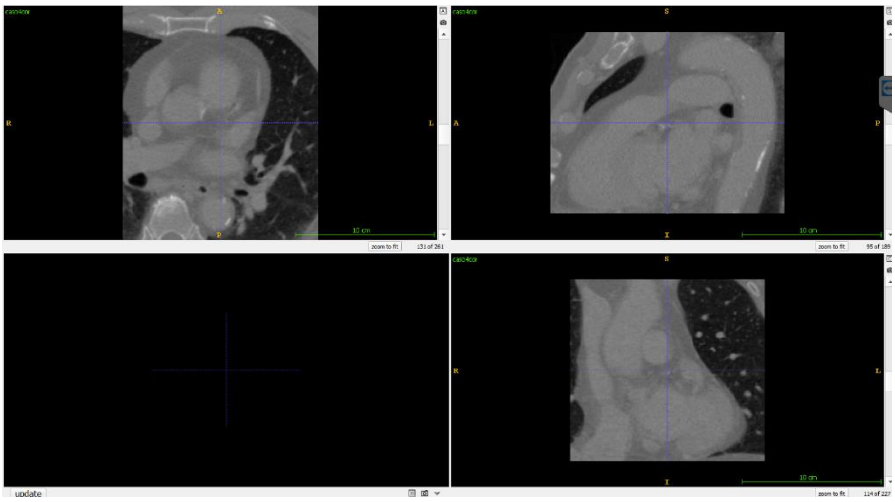

Figura 39: Corazón segmentado

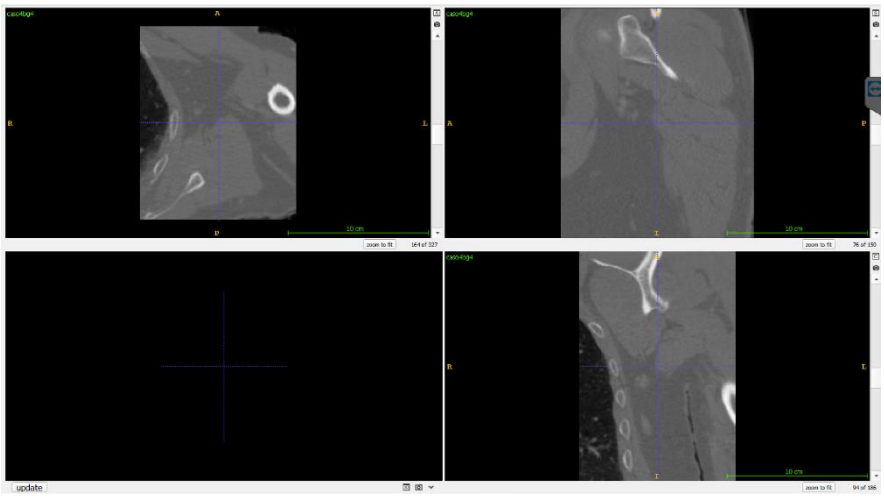

Figura 40: Background aleatoria segmentado

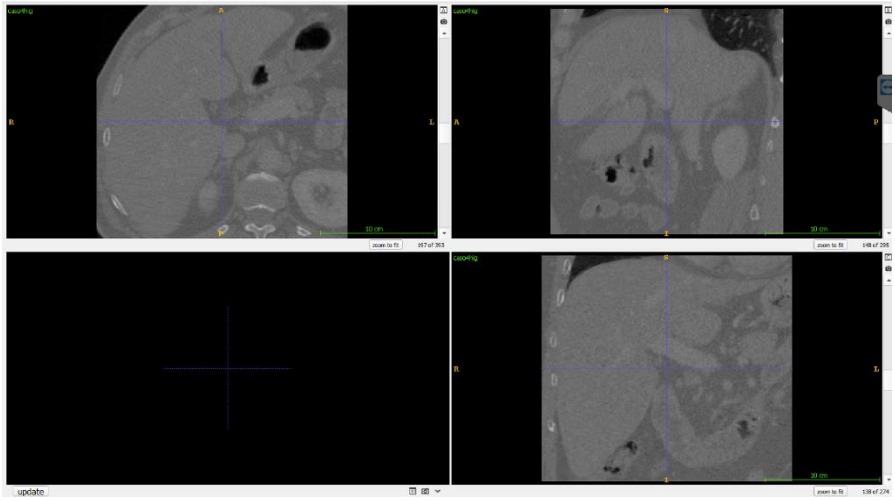

Figura 41: Hígado segmentado

Cabe destacar que hemos pasado de tener una muestra de tamaño 1251x512x512 a tres muestras de tamaño 261x189x227 (corazón), 327x150x186 (background) y 393x295x274 (hígado), por lo que reducimos el tamaño de 327.942.144 características a 52.087.173, es decir, se ha reducido 6 veces la dimensión del problema. Esta reducción sigue siendo insuficiente, por ello se requiere de un procesado de datos que sea capaz de reducir la dimensionalidad en mayor medida. Al finalizar el proceso de etiquetado se han obtenido 300 muestras de entrenamiento, que son 100 corazones, 100 backgrounds, 100 hígados.

4.1.2 Procesado de los órganos etiquetados

El principal objetivo del procesado de los datos es, como se ha comentado anteriormente, reducir drásticamente la dimensionalidad de las muestras de entrenamiento, para ello, después de mucha experimentación, se ha decidido que el procesado de los datos se centre en las siguientes dos tareas:

- Normalización de los valores de intensidad de píxel a un rango entre 0 y 1
- Diezmado (reducción) del tamaño de las muestras de entrenamiento.

La intensidad de píxel de las tomografías computarizadas no está definida en el rango habitual de otros formatos de imagen (de 0 a 255 o de 0 a 1), debido a que dicha intensidad está representada en unidades de Hounsfield que comprenden valores entre -1024 y 3071, y, por tanto, proporcionan 4096 niveles de intensidad de pixel diferentes. Por este motivo cada pixel debe ser representado con 12 bits ($2^{12} = 4096$). Pero, para trabajar en un rango más estándar de niveles de gris, se ha decidido normalizar los valores de intensidad en unidades de Hounsfield a valores de intensidad en 0 y 1.

Por otra parte, se ha decido re-escalar todos los órganos etiquetados a un tamaño de 25x25x25, debido a que se considera que con estas dimensiones sigue siendo posible diferenciar sin problema las diferentes estructuras que se pretende clasificar. Una vez aplicado el diezmado cada una de las muestras de entrenamiento pasa a tener 15625 características, por tanto, nos enfrentamos a un problema mucho más abarcable. En las Figuras 42, 43, y 44 se pueden observar las muestras de entrenamiento después del procesado.

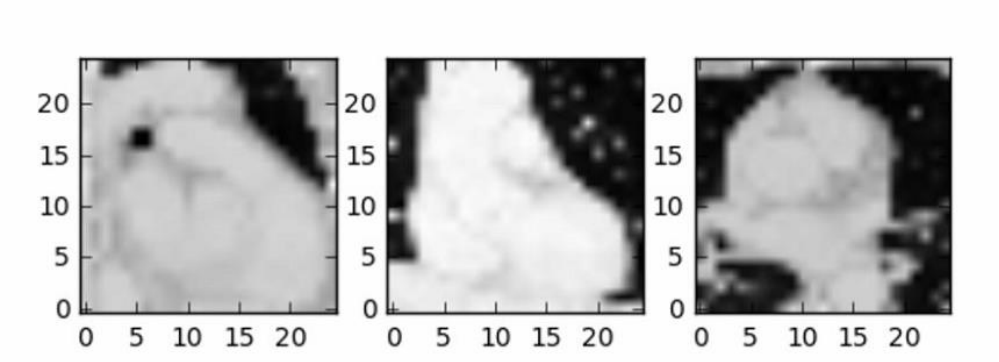

Figura 42: Corazón procesado

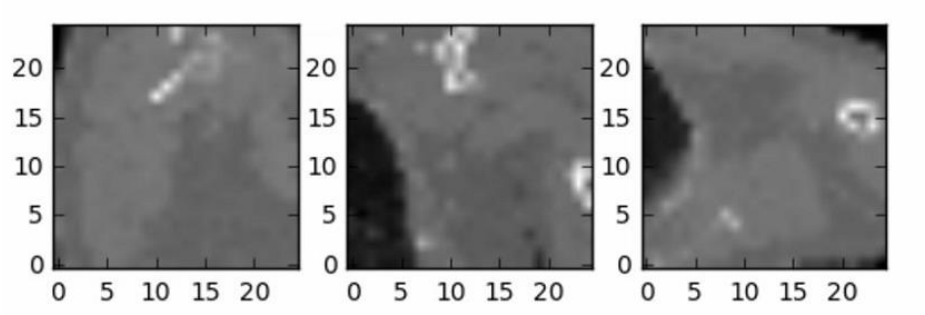

Figura 43: Background procesado

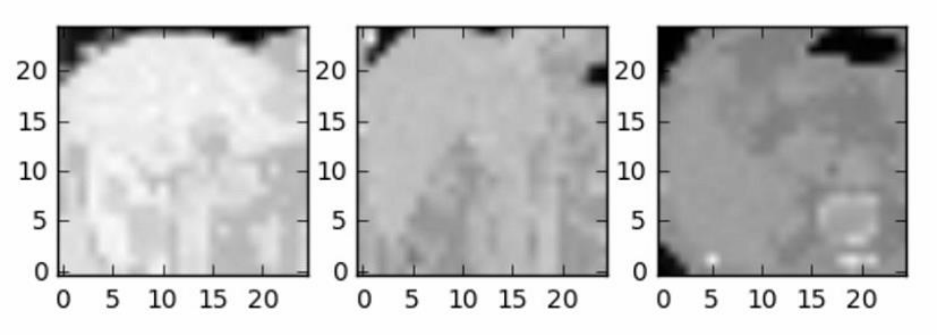

Figura 44: Hígado procesado

Si se compara las Figuras 42, 43 y 44 con las Figuras 39, 40 y 41 se puede ver que la resolución de las muestras se ha reducido sustancialmente, que es lo que se pretendía conseguir en el procesado con el objetivo de recudir la dimensionalidad de las muestras.

Una vez se han procesado todas las muestras solo queda escribir un código sencillo para guardar el dataset con sus respectivas etiquetas para poder pasar a diseñar el clasificador. Es importante a la hora de guardar las muestras que se ordenen de forma aleatoria, para que no estén todas las muestras del mismo tipo unas a continuación de las otras, con el objetivo de mejorar la precisión del modelo entrenado.

2- Diseño del clasificador

Como se ha comentado anteriormente, uno de los elementos clave a la hora de diseñar la arquitectura de la red neuronal que se va a entrenar es que solo se cuenta con 300 muestras de entrenamiento, por tanto, la complejidad de la red no puede ser muy elevada.

Otro elemento clave para diseñar la arquitectura es tener en cuenta que las tomografías computarizas son imágenes tridimensionales, por tanto, la mejor opción es utilizar una red neuronal con filtros convolucionales 3D como se explicará a continuación.

Por tanto, teniendo en cuenta los detalles mencionados en los párrafos anteriores, se ha decido que la red neuronal que se va a utilizar va a atacar el problema tenga la arquitectura de la Figura 45.

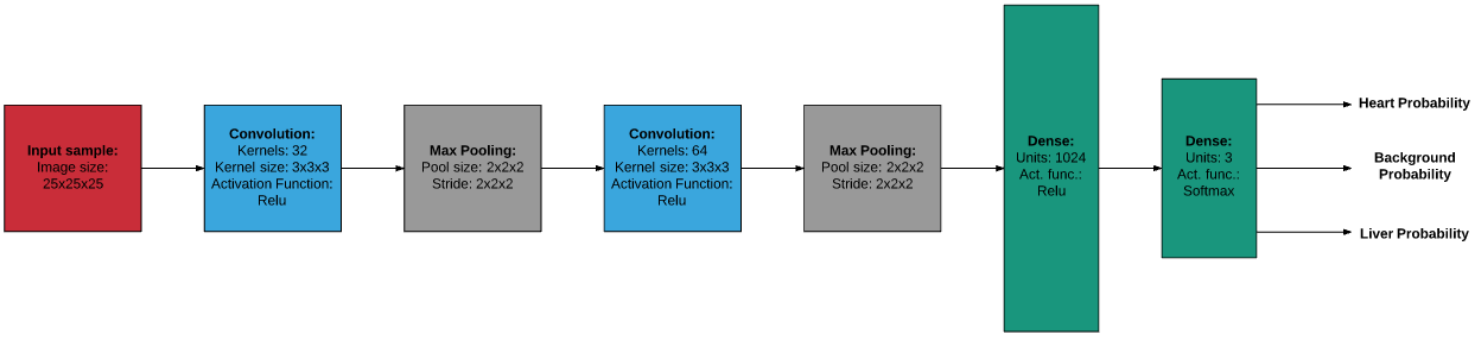

Figura 45: Arquitectura de la Red Neuronal Convolucional 3D

Como se puede observar en la Figura 45 la red neuronal que se plantea es muy poco profunda para que pueda ser entrenada con pocas muestras de entrenamiento y además se ejecute lo más rápido posible.

La red consta de 2 capas convoluciones, 2 capas de max pooling y 2 capas dense (también conocidas como fully connected). La función de las capas convolucionales de la red es extraer las características más relevantes de las imágenes de entrada. Las capas de max pooling reducen cada una de las dimensiones de los mapas de entrada a la mitad quedándose con los puntos de máxima intensidad dentro de las ventanas 2x2x2 en las que se dividen los mapas. Y, por último, las capas fully connected tienen la función de combinar las características extraídas por los filtros convolucionales para obtener la mejor clasificación posible.

También cabe destacar que en todas las capas de la red se ha utilizado como función de activación la función relu para evitar el desvanecimiento del gradiente durante la fase de entrenamiento, salvo en la capa de salida de la red (última dense), donde se ha utilizado

la función softmax, debido a que esta función presenta una salida probabilística, lo que permite estimar la probabilidad que le asigna el clasificador entrenado a cada una de las diferentes clases con que se ha entrenado el sistema (corazón, background o hígado).

Debido a que nos enfrentamos a un problema de clasificación, el sistema ha sido entrenado con la función de pérdidas 'categorical crossentropy' con una ratio de aprendizaje de $1e^{-4}$. De las 300 muestras etiquetadas de las que se dispone se han utilizado 240 para entrenar el sistema y las 60 restantes para la evaluación del modelo.

```
Epoch 1/10
237/237 [==============================] - 232s - loss: 1.5001 - acc: 0.4515 - val_loss: 0.7390 - val_acc: 0.7167
Epoch 2/10
237/237 [==============================] - 278s - loss: 0.5619 - acc: 0.7932 - val_loss: 0.5046 - val_acc: 0.7667
Epoch 3/10
237/237 [==============================] - 301s - loss: 0.5188 - acc: 0.8186 - val_loss: 0.2824 - val_acc: 0.9667
Epoch 4/10
237/237 [==============================] - 303s - loss: 0.2688 - acc: 0.9283 - val_loss: 0.2460 - val_acc: 0.9167
Epoch 5/10
237/237 [==============================] - 302s - loss: 0.3106 - acc: 0.9072 - val_loss: 0.1688 - val_acc: 0.9833
Epoch 6/10
237/237 [==============================] - 300s - loss: 0.1161 - acc: 0.9916 - val_loss: 0.1490 - val_acc: 0.9333
Epoch 7/10
237/237 [==============================] - 303s - loss: 0.1531 - acc: 0.9578 - val_loss: 0.3585 - val_acc: 0.8333
Epoch 8/10
237/237 [==============================] - 317s - loss: 0.0929 - acc: 0.9789 - val_loss: 0.1398 - val_acc: 0.9667
Epoch 9/10
237/237 [==============================] - 317s - loss: 0.0705 - acc: 0.9916 - val_loss: 0.0660 - val_acc: 0.9833
Epoch 10/10
237/237 [==============================] - 323s - loss: 0.0710 - acc: 0.9831 - val_loss: 0.0688 - val_acc: 0.9667
```

Figura 46: Reporte de entrenamiento

La red planteada ha sido entrenada en tan solo 10 epochs, y, como se puede observar en la Figura 46, a partir del tercer epoch la red ya alcanza una precisión de 96,67% sobre el test de validación. Por tanto, se han obtenido unos resultados muy satisfactorios dada la complejidad de las TC, las pocas muestras de entrenamiento y la poca profundidad de la red entrenada.

Una vez obtenido un clasificador que es capaz de detectar la presencia de órganos (hígado, corazón y background) dada una ventana concreta de una tomografía computarizada el siguiente paso para obtener un sistema de detección de órganos es implementar una ventana deslizante tridimensional que sea capaz de recorrer una TC completa para analizarla y devolver las coordenadas de los órganos detectados.

3- Implementación del clasificador en una ventana deslizante

Una ventana deslizante es, en el caso de imágenes en 3D, un cubo de unas dimensiones fijas que recorre toda la imagen. En la Figura 47 podemos ver un ejemplo de ventana deslizante en 2 dimensiones.

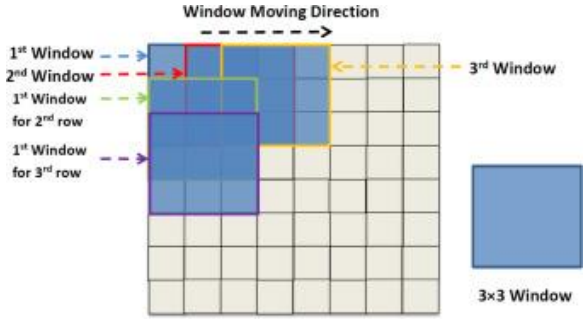

Figura 47: Ejemplo de ventana deslizante en 2D

Como se puede ver en la Figura 47 la ventana deslizante se desplaza a lo largo de toda la imagen obteniendo todas las subáreas que existen con el tamaño de la ventana deslizante dentro de la imagen original. Esto es útil para la detección de órganos debido a que el clasificador que se ha explicado en el apartado anterior pude utilizarse para distinguir si dichas subáreas contienen un órgano o no.

El factor más limitante para aplicar el método mencionado es que dentro de una tomografía computarizada en 3D se pueden seleccionar cientos de miles de subáreas de diferentes y, por tanto, se debería ejecutar el clasificador para cada una de ellas, lo que puede ser un proceso muy lento.

Para solventar el problema mencionado se ha decidido reducir la dimensionalidad de cada una de las tomografías que sean analizadas por el sistema a un tamaño de 75x75x75, debido a que después de experimentar con muchos valores diferentes se ha llegado a la conclusión de que este brinda un buen equilibrio entre detección de órganos y velocidad de ejecución del sistema.

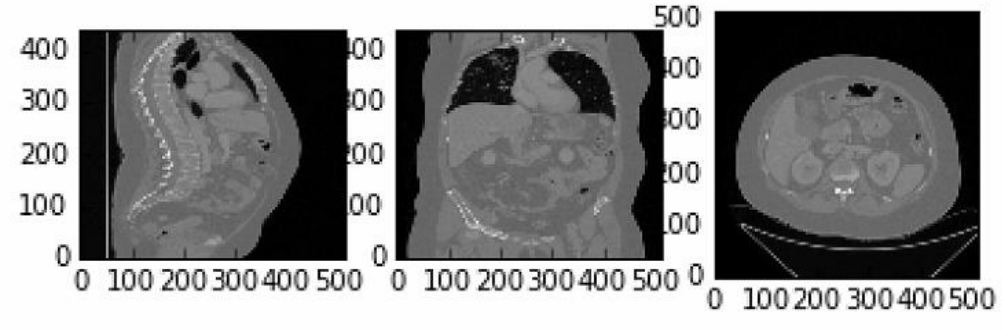

Figura 48: TC tamaño original

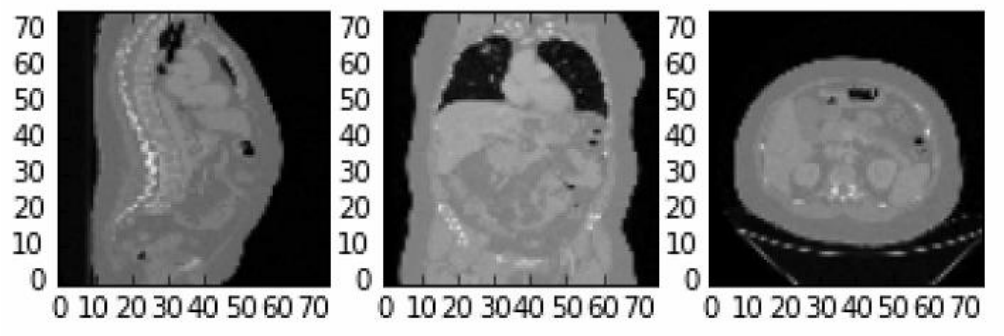

Figura 49: TC 75x75x75

Por último, se ha fijado el tamaño de la ventana deslizante a 25x25x25, debido a que se ha comprobado que es un tamaño idóneo para el problema al que nos estamos enfrentando.

En cada tomografía de tamaño 75x75x75 encontramos 17576 subáreas de tamaño 25x25x25 si utilizamos un salto de 2 vóxeles en cada desplazamiento de la ventana deslizante, por tanto, es necesario ejecutar el clasificador diseñado 17576 veces para analizar cada uno de los TC, tarea que puede realizarse en un tiempo asumible.

Al implementar el método de ventana deslizante descrito se ha observado que el clasificador entrenado presenta un elevado número de falsos positivos, en gran parte debido a que se necesitan muchas más muestras de background para modelizar de forma correcta todas las posibles estructuras que no son ni corazón ni hígado. Para solucionar este problema se ha decidido tomar como detección de corazón e hígado solo al sub-volumen al que mayor probabilidad otorgue el clasificador para cada uno de estos órganos dentro de la tomografía.

En las Figura 50 a 53 pueden verse ejemplos de detección de corazón e hígado en la TC de dimensionalidad reducida y sus respectivas extrapolaciones en la TC de tamaño original.

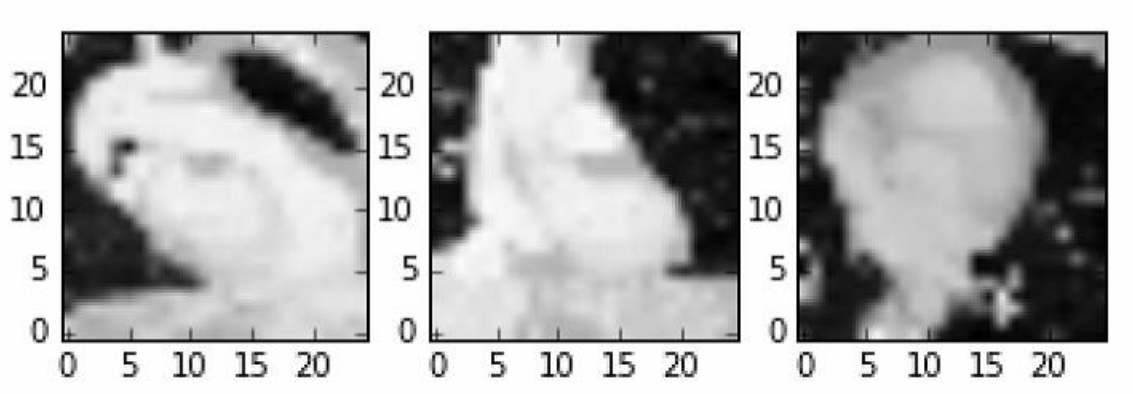

Figura 50: Corazón detectado en ventana de 25x25x25

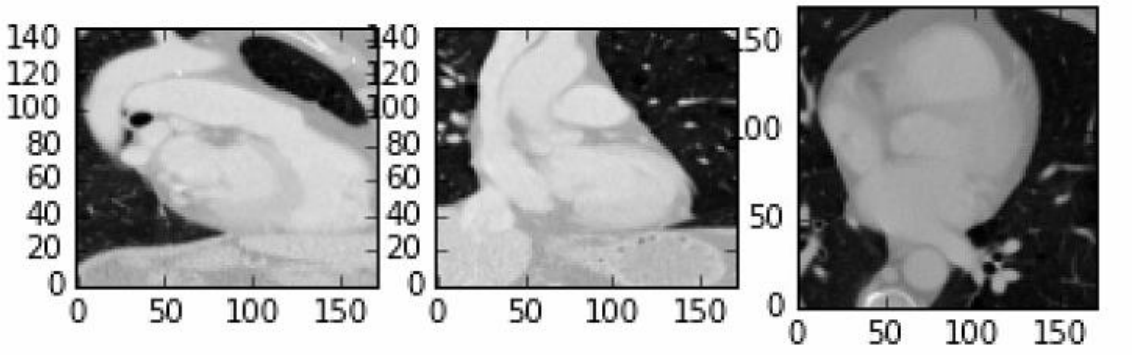

Figura 51: Corazón de la TC detectado

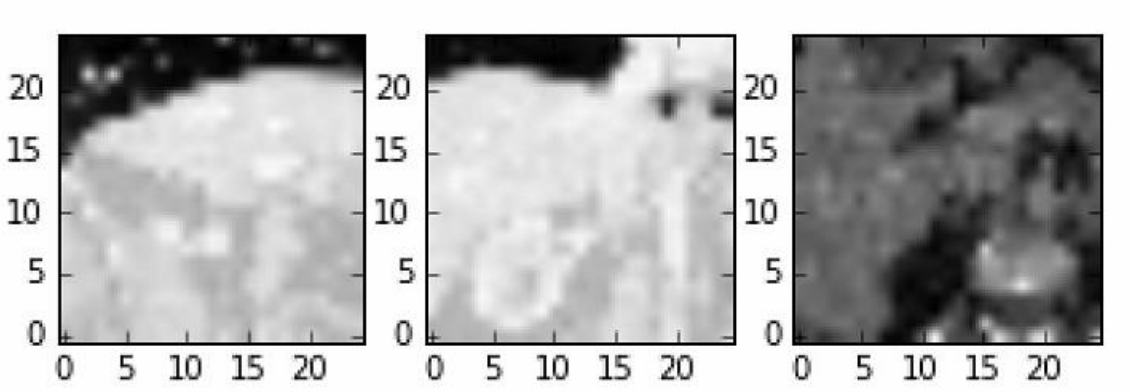

Figura 52: Hígado detectado en ventana de 25x25x25

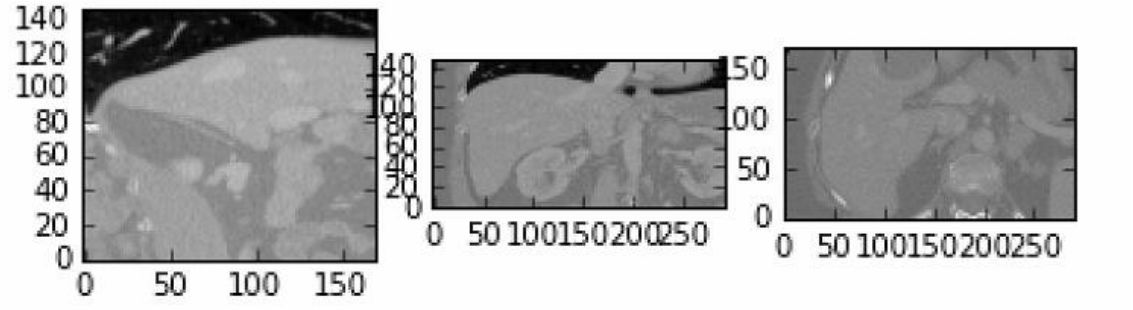

Figura 53: Hígado de la TC detectado

4- Resultados

Para evaluar el comportamiento del sistema de detección automática de órganos desarrollado, se han analizado 101 tomografías que el sistema no ha analizado durante la fase de entrenamiento, es decir, un conjunto de tomografías de test que simulan los problemas a los que el algoritmo se debe enfrentar en condiciones normales. En dichas muestras se ha medido tanto la veracidad en la detección del sistema como el tiempo de ejecución del mismo.

4.4.1 Tiempos de ejecución

Uno de los objetivos principales a conseguir era que el sistema desarrollado no fuese lento (muchos minutos o horas para analizar cada TC), por ello, como se ha explicado en apartados anteriores, muchos de los procedimientos y decisiones de diseño que se han llevado a cabo han tenido el objetivo de aligerar la computación necesaria para analizar las tomografías. En la Figura 54 pueden observarse los tiempos de ejecución obtenidos en las 101 tomografías con un re-escalado de TC a 75x75x75, una ventana deslizante de 25x25x25 y un salto de 2 vóxeles en desplazamiento de ventana.

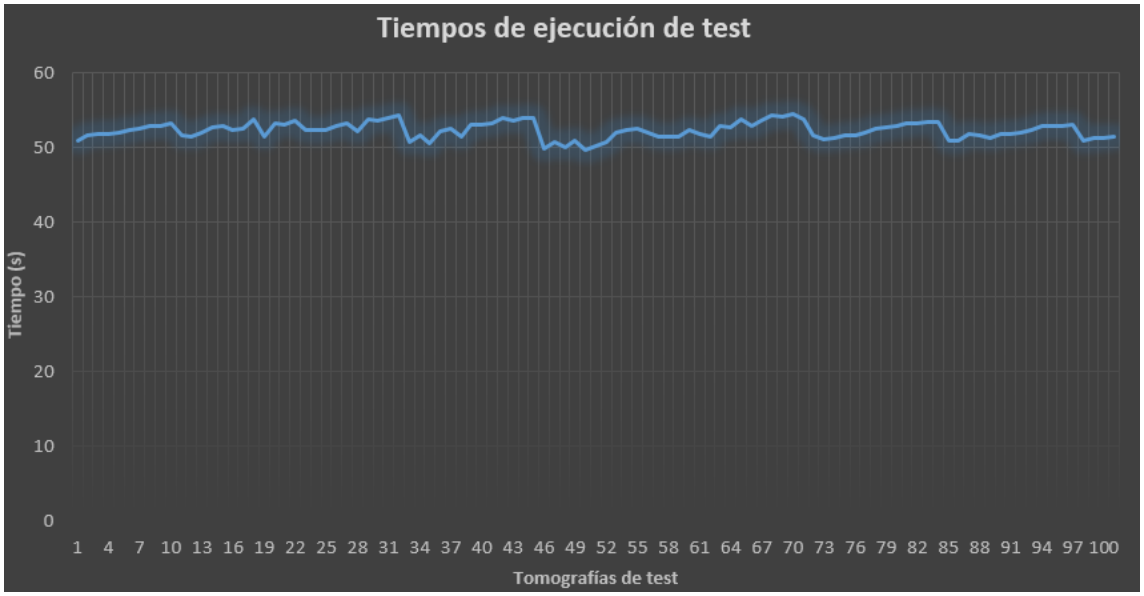

Figura 54: Tiempos de análisis de las tomografías de test

Como puede observarse en la Figura 54, en ningún caso se superan los 60 segundos de ejecución, por ello se concluye que el sistema diseñado es capaz de analizar tomografías en menos de un minuto independientemente de su tamaño, debido a que las dimensiones de las 101 tomografías de prueba varían.

El equipo con el que se han llevado a cabo las pruebas es un ordenador portátil MSI GS40 Phantom con una tarjeta gráfica dedicada NVIDIA GeForce 970M, por tanto, los tiempos de ejecución obtenidos pueden reducirse drásticamente si el sistema se ejecuta en un ordenador profesional de sobremesa.

4.4.2 Tasas de detección del sistema

Como se ha comentado anteriormente, el sistema presenta una elevada tasa de falsos positivos para cada TC. Estos falsos positivos son sub-volúmenes dentro del TC a los que la red neuronal asigna una mayor probabilidad de ser órgano (corazón o hígado) que background, cuando en realidad no lo son. En la Figura 55 se puede observar el número de falsos positivos por órgano y tomografía.

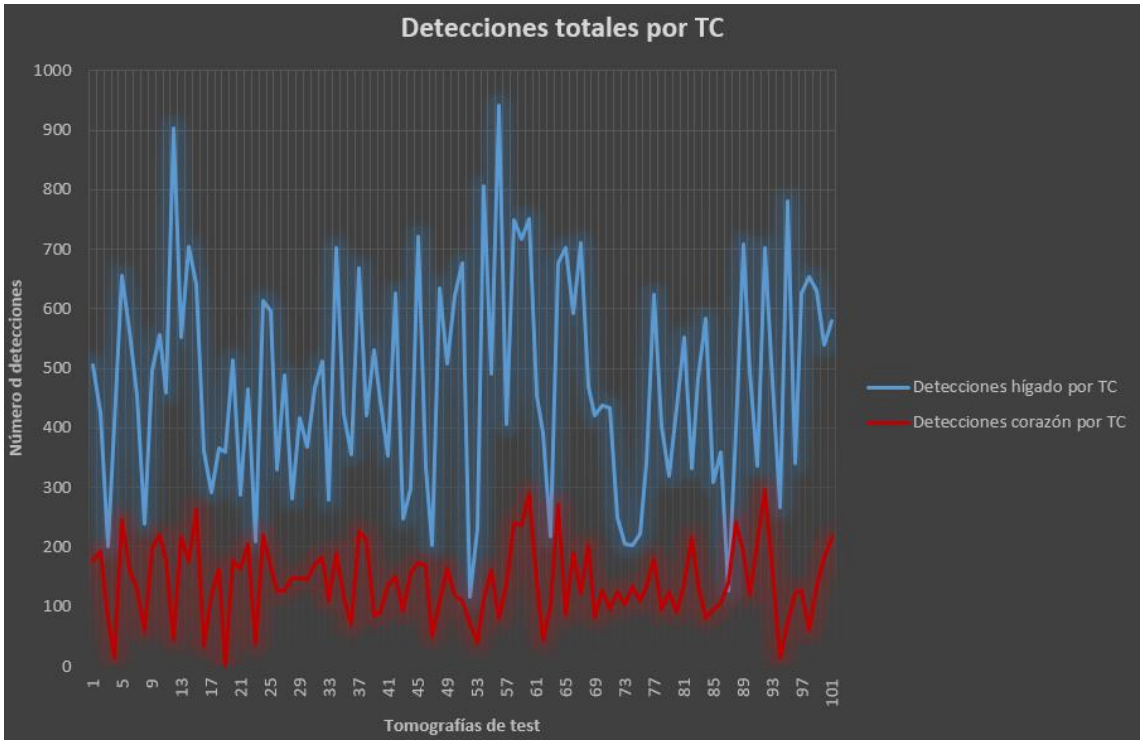

Figura 55: Detecciones por órgano y TC

Como se puede observar en la Figura 55 en cada tomografía se detectan un elevado número tanto de hígados como de corazones, cuando lo esperado sería detectar tan solo uno de cada. Sin embargo, se ha obtenido una media de 141 detecciones de corazón por TC y una media de 473 para hígado. Esto es debido a la falta de muestras de entrenamiento y la simplicidad de la red entrenada.

Para solventar este problema, como se ha comentado anteriormente, se tomarán como corazón e hígado detectados a los que mayor probabilidad les haya asignado la red de entre todas las detecciones (argmax). Una vez aplicada esta medida los resultados en detección puede verse en las Figuras 56 y 57.

Figura 56: Detección de Corazones

Figura 57: Detección de hígados

Como se puede observar en las Figuras 56 y 57 los corazones detectados correctamente por el sistema son el 98% (99 de 101), y, los hígados el 100%, por tanto, el sistema no solo cumple las expectativas, si no que las sobrepasa, debido a que ofrece unas tasas de detección excelentes pese el escaso dataset de entrenamiento del que se dispone.

Capítulo 5. Conclusiones y Trabajo futuro

1- Conclusiones

Como conclusiones a este trabajo cabe destacar que se han obtenido unos resultados muy positivos pese a las grandes limitaciones con las que se partía (especialmente el escaso dataset) y la complejidad alta del problema que se pretendía resolver. El trabajo realizado demuestras que las redes neuronales pueden ser empleadas de forma efectiva, eficaz y eficiente con imágenes tridimensionales pese a su gran dimensionalidad.

El método propuesto consigue no solo unas tasas de detección muy elevadas, si no que se ejecuta de forma rápida (menos de un minuto) en imágenes de alta dimensionalidad, es por ello que se considera que es un camino muy viable para conseguir resolver el problema de detección de órganos.

Por otra parte, es importante resaltar que el presente proyecto es una prueba de concepto y que para conseguir un sistema robusto de detección de órganos funcional y aplicable en la industria médica se necesita de mucho más trabajo y experimentación para conseguir optimizar el sistema al máximo, así como demostrar de forma irrefutable que el método propuesto puede ser de ayuda a los profesionales de la sanidad para llevar a cabo de forma más eficaz y veloz su trabajo.

2- Trabajo futuro

Como se ha comentado en las conclusiones, se debe llevar a cabo mucho más trabajo para obtener un sistema de detección de órganos robusto. Las principales áreas de mejora que se han detectado son las siguientes:

- **Recopilación y etiquetado de un dataset muy superior al que se dispone actualmente**: Este debería ser el foco principal del trabajo debido a que al disponer de un dataset más robusto y abundante se pueden entrenar modelos de redes neuronales mucho más precisos y fiables.

- **Etiquetar un mayor número de órganos**: Debido a las limitaciones temporales en el presente proyecto solo se han etiquetado corazones e hígados, pero para desarrollar un sistema de detección de órganos realmente útil en la industria se precisa de capacidad de detección de muchas más estructuras del cuerpo humano.

- **Experimentar con una mayor variedad de arquitecturas de redes neuronales**: De nuevo, debido a las limitaciones temporales, el número de arquitecturas de redes neuronales con las que se ha trabajado no es todo lo abundante que se desearía, por ello, experimentar con diversos modelos de redes es fundamental para obtener mejoras en el sistema.

- **Implementar el sistema en una tarjeta gráfica profesional**: El tiempo de ejecución del método propuesto se reduciría sensiblemente si se implementa en una tarjeta gráfica más potente, permitiendo cálculos más rápidos o análisis más complejos en el mismo tiempo.

Capítulo 6. Bibliografía

1- Farag, A., Lu, L., Turkbey, E., Liu, J., & Summers, R. M. (2014, September). A bottom-up approach for automatic pancreas segmentation in abdominal CT scans. In *International MICCAI Workshop on Computational and Clinical Challenges in Abdominal Imaging* (pp. 103-113). Springer, Cham.

2- Criminisi, A., Shotton, J., & Bucciarelli, S. (2009, September). Decision forests with long-range spatial context for organ localization in CT volumes. In *MICCAI Workshop on Probabilistic Models for Medical Image Analysis* (Vol. 1, pp. 146-158).

3- Roth, H. R., Lu, L., Farag, A., Shin, H. C., Liu, J., Turkbey, E. B., & Summers, R. M. (2015, October). Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 556-564). Springer, Cham.

4- Criminisi, A., Shotton, J., Robertson, D. P., & Konukoglu, E. (2010). Regression Forests for Efficient Anatomy Detection and Localization in CT Studies. *MCV, 2010*, 106-117.

5- Criminisi, A., Robertson, D., Konukoglu, E., Shotton, J., Pathak, S., White, S., & Siddiqui, K. (2013). Regression forests for efficient anatomy detection and localization in computed tomography scans. *Medical image analysis*, *17*(8), 1293-1303.

6- Roth, H. R., Farag, A., Lu, L., Turkbey, E. B., & Summers, R. M. (2015). Deep convolutional networks for pancreas segmentation in CT imaging. *arXiv preprint arXiv:1504.03967*.

7- Criminisi, A., Robertson, D., Pauly, O., Glocker, B., Konukoglu, E., Shotton, J., ... & Navab, N. (2013). Anatomy detection and localization in 3D medical images. In *Decision Forests for Computer Vision and Medical Image Analysis* (pp. 193-209). Springer London.

8- Roser, M. (2015). Life Expectancy. In **https://ourworldindata.org/life-expectancy**.

9- Templeton, G. (2016). AI beats doctors at visual diagnosis, observes many times more lung cancer signals. In **https://www.extremetech.com/extreme/233746-ai-beats-doctors-at-visual-diagnosis-observes-many-times-more-lung-cancer-signals**

10- Criminisi, A., Robertson, D., Pauly, O., Glocker, B., Konukoglu, E., Shotton, J., ... & Navab, N. (2013). Introduction: The Abstract Forest Model. In *Decision Forests for Computer Vision and Medical Image Analysis* (pp. 22-38). Springer London.

11- Criminisi, A., Robertson, D., Pauly, O., Glocker, B., Konukoglu, E., Shotton, J., ... & Navab, N. (2013). Classification Forest. In *Decision Forests for Computer Vision and Medical Image Analysis* (pp. 39-59). Springer London.

12- Criminisi, A., Robertson, D., Pauly, O., Glocker, B., Konukoglu, E., Shotton, J., ... & Navab, N. (2013). Regression Forests. In *Decision Forests for Computer Vision and Medical Image Analysis* (pp. 60-72). Springer London.

13- Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees. Chapman and Hall/CRC, London.

14- Cole E, Hussein S. Automatic Detection of Multiple Organs Using Convolutional Neural Networks.

15- Paul A. Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C. Gee, and Guido Gerig (2006). User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *In Neuroimage*.