ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

TRABAJO DE FIN DE GRADO

EVALUACIÓN DE ALGORITMOS DE CLASIFICACIÓN PARA EL DIAGNÓSTICO DE CÁNCER A PARTIR DE IMÁGENES

Autor: Ricardo Moreno Alonso

Directores:

Jaime Boal Martín-Larrauri

Santiago Moreno Carbonell

MADRID

Julio 2018

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. RICARDO MORENO ALONSO

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: "Evaluación de algoritmos de clasificación para el diagnóstico de cáncer a partir de imágenes", que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que

pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a ....17..... de ....julio..... de .....2018.....

ACEPTA Ricardo Moreno Alonso

Fdo.....

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título: "Evaluación de algoritmos de clasificación para el diagnóstico de cáncer a partir de imágenes" en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2017-2018 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Ricardo Moreno Alonso Fecha: 17/ 07/ 2018

Autorizada la entrega del proyecto

LOS DIRECTORES DEL PROYECTO

Fdo.: Jaime Boal Martín-Larrauri Fecha: 16/ 07/ 2018

Fdo.: Santiago Moreno Carbonell Fecha: 16/ 07/ 2018

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

TRABAJO DE FIN DE GRADO

EVALUACIÓN DE ALGORITMOS DE CLASIFICACIÓN PARA EL DIAGNÓSTICO DE CÁNCER A PARTIR DE IMÁGENES

Autor: Ricardo Moreno Alonso

Directores:

Jaime Boal Martín-Larrauri

Santiago Moreno Carbonell

MADRID

Julio 2018

EVALUACIÓN DE ALGORITMOS DE CLASIFICACIÓN PARA EL DIAGNÓSTICO DE CÁNCER A PARTIR DE IMÁGENES

Autor: Moreno Alonso, Ricardo

Directores: Boal Martín-Larrauri, Jaime.; Moreno Carbonell, Santiago

Entidad colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

La motivación del proyecto surge del importante crecimiento reciente del campo del aprendizaje automático aplicado a la medicina. El interés del proyecto reside en la gran incertidumbre que existe en el aprendizaje automático sobre las mejores prácticas en el diseño de los modelos. Esto se debe, en parte a la naturaleza del campo, y en parte debido a que es difícil comparar resultados de la literatura existente debido a que se utilizan datos de entrenamiento y parámetros diferentes en cada caso.

El objetivo de este proyecto es realizar experimentos con distintas técnicas de aprendizaje automático aplicadas al diagnóstico asistido por ordenador, todos ellos con una base de datos única para poder hacer comparaciones. En particular, se aplican éstas técnicas en el diagnóstico automático de cáncer de pulmón a partir de imágenes tridimensionales de tomografía computarizada. En la siguiente figura se muestra una sección de una de éstas imágenes:

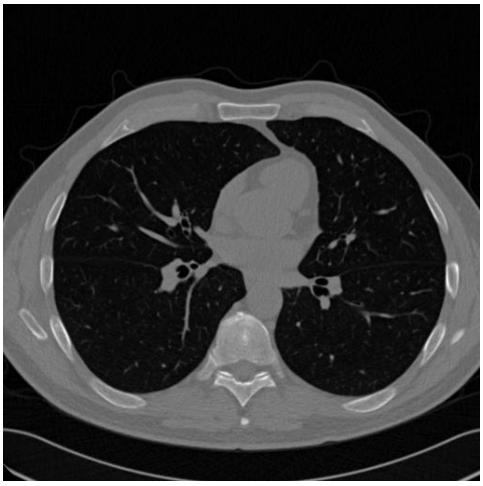

Figura 1. Ejemplo de sección de una imagen 3D de pulmón.

En la literatura existente sobre el diagnóstico automático a partir de imágenes médicas, las siguientes técnicas son algunas de las más populares: las redes neuronales artificiales (ANN) [1], las redes neuronales convolucionales (CNN) [2], las máquinas de vectores de soporte (SVM) [3] y el algoritmo de K vecinos más cercanos (KNN) [4]. En este proyecto se experimenta principalmente con distintas modalidades de redes neuronales convolucionales.

Las redes neuronales convolucionales se utilizan frecuentemente en tareas de visión artificial y de clasificación de imágenes debido a que su estructura está bien adaptada para explotar la naturaleza de la información contenida en una imagen (abundante correlación espacial, por ejemplo). En este proyecto se realizan experimentos con cuatro modelos de aprendizaje automático basados en las redes neuronales convolucionales:

1. **Modelo global:** Red neuronal convolucional que extrae un diagnóstico para el paciente a partir de la imagen 3D entera.

2. **Modelo local:** Combinación de dos sistemas. Se trata de un modelo basado en análisis local por zonas de la imagen 3D. Por un lado se diseña una red neuronal convolucional 3D cuya entrada es un recorte de la imagen global de lado aproximadamente 2 veces el diámetro de un nódulo pulmonar medio y cuya salida es una predicción de si el "cubo" en cuestión contiene o no un nódulo. Se desplaza el detector de nódulos por la imagen completa y se genera un tensor de predicciones. Finalmente, se utiliza un algoritmo de *clustering* aglomerativo jerárquico para agrupar las predicciones positivas del modelo y dar un diagnóstico global del paciente.

3. **Modelo *transfer learning*:** Basado en la técnica del *transfer learning*. El procedimiento se basa en tomar de internet una red neuronal clasificadora de imágenes entrenada con un base de datos muy grande y adapatarla al problema en cuestión. Se toma el modelo de base y se le añade un clasificador a la salida. Se adaptan también las últimas capas del modelo base. Este modelo aplica sobre imagenes 2D obtenidas, de nuevo, a partir de secciones de las imágenes 3D.

4. **Estimador de malignidad:** A diferencia de los precedentes, se trata de un modelo de regresión, no de clasificación. En este caso se aplica una red neuronal convolucional para predecir la malignidad de un tumor. La entrada del modelo es un recorte de la imagen completa del mismo tamaño que en el modelo local. En este caso el modelo aplica únicamente sobre recortes que contienen un nódulo. La salida es una predicción de su malignidad (número entre 0 y 5).

Para el entrenamiento de los modelos se utiliza la base de datos del concurso LUNA16, que a su vez proviene de la base de datos pública de imágenes médicas LIDC/IDRI. Se tiene un conjunto de 888 imágenes 3D de tomografía computarizada de pulmón. Además de dispone de anotaciones de 4 radiólogos de éstas imágenes. Se tiene una lista de 1186 nódulos cancerígenos encontrados en este conjunto de imágenes. Para cada nódulo se da el identificador único de la imagen a la que pertenece (identificador del paciente), sus coordenadas dentro de ésta imagen, su malignidad (número entre 0 y 5 resultado de la valoración de los radiólogos) y otras características del mismo.

Todos los modelos del proyecto se diseñan, entrenan y validan en Python, utilizando principalmente la librería Keras con la libería Tensorflow como programa de soporte. Se lanzan los entrenamientos y las pruebas en una máquina virtual de Google Compute Engine con GPU integrada.

Para cada modelo utilizado se prueban distintas combinaciones de parámetros y se comparan resultados. Las conclusiones son las siguientes:

Los malos resultados del **modelo global** parecen indicar que se trata de un problema demasiado complejo y sutil como para atacarlo con una red convolucional de escasa profunidad

(6 capas) y complejidad (entorno a 270000 parámetros), dado el tamaño del conjunto de imágenes.

La red neuronal del **modelo local** da los mejores resultados de clasificación (79 % de precisión y 91 % de sensibilidad en su mejor versión). Esto demuestra que un diagnóstico por zonas de la imagen facilita enormemente la tarea de clasificación. El algoritmo de *clustering* utilizado en este modelo no funcionó. Se cree que esto se debe a la tendencia del conjunto de predicciones positivas locales a ser conexo.

Los resultados del **modelo de *transfer learning*** son algo peores en clasificación que en el modelo local pero mucho mejores que en el modelo global. La comparación de los resultados para los distintos modelos de base sugiere que un incremento del número de parámetros (y no del número de capas) de la red mejora el diagnóstico. El modelo base de mejores resultados es el VGG19 con una precisión media del 63 % y una sensibilidad del 40 % (casi el doble que los otros).

El **estimador de malignidad** obtiene un error cuadrático medio de 0.89 en la mejor versión del modelo. La malignidad se mide de 0 a 5 y tiene un valor medio de 3. Se ha de tener en cuenta que las etiquetas de malignidad son una media de las valoraciones subjetivas de 4 radiólogos. Se hace la hipótesis de que esta es la principal razón de que el error sea tan elevado.

Como futuros desarrollos se propone, en primer lugar, una combinación del modelo local y del de *transfer learning*. Se propone también el uso de otras características nodulares como la espiculación y la lobulación para el diagnóstico. Por último, se proponen dos vías de desarrollo del algoritmo de *clustering* del modelo local.

O

EVALUATION OF CLASSIFICATION ALGORITHMS FOR CANCER DIAGNOSIS FROM IMAGES

Author: Moreno Alonso, Ricardo

Directors: Boal Martín-Larrauri, Jaime.; Moreno Carbonell, Santiago

Collaborating Entity: ICAI – Universidad Pontificia Comillas

SUMMARY OF THE PROJECT

The motivation behind this project comes from the significant recent growth of the field of machine learning applied to medical sciences. The interest of the project lies in the huge uncertainty that exists in machine learning regarding what the best practices are when designing a model. This is in part due to the nature of the subject and partly due to the fact that it is difficult to compare experimental results from the literature because of differences in training data and parameters in each case.

The goal of this project is to carry out experiments in the application of several machine learning techniques to computer aided diagnostis, all with the same data set so that comparisons can be made. In particular, these techniques are applied to giving an automatic diagnostis of lung cancer from 3D computerized tomography lung images. The following figure shows an example of a cross-section of one of such images:

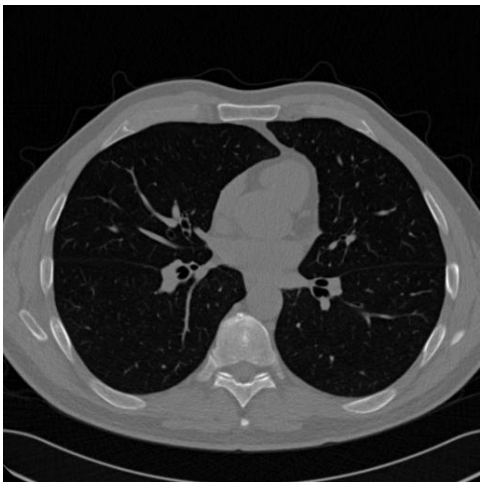

Figure 2. Example of a cross-section of a 3D lung image.

The existing literature on the subject of computer aided diagnostic from medical images is composed mainly of different applications of the following machine learning techniques: artificial neural networks (ANN) [1], convolutional neural networks (CNN) [2], support vector machines (SVM) [3] and K nearest neighbours algorithm (KNN) [4]. This project focuses mainly on applying different variations of convolutional neural networks.

Convolutional neural networks are frequently used in artificial vision and image classification tasks due to the fact that their structure is well suited to exploit the nature of the information held in an image (abundant spatial correlation for example). In this project experiments are carried out with the following four models based on convolutional neural networks:

1. **Global model:** Convolutional neural network which outputs a cancer diagnostis for a given patient from their whole lungs image.

2. **Local model:** Made up of the combination of two systems. This model is based on a local analysis of the 3D images approach. First, a 3D convolutional neural network is designed. The input of this network is a cubic subregion of the global 3D image. The side of such portion of the image is taken to be approximately twice the average nodule diameter. The network gives a prediction on whether the cube contains a nodule or not. The nodule detector is moved around the global image, giving a prediction on every position and thus generating a tensor of predictions. Finally, a clustering algorithm is used to gather together the positive predictions from the model and give a global diagnosis for the patient.

3. **Transfer learning model:** Based on transfer learning. The procedure consists of taking a pretrained model on a large image database and adapting it to the problem. An image classificator is added at the end of the base model. The last layers of the base model are also fine tuned. This model applies on 2D images, again obtained as cross-sections of 3D images.

4. **Malignancy estimator:** Unlike the previous ones, this one is a regression model, not a classification one. In this case, a 3D convolutional neural network is used to predict the malignancy of a tumor. The input of the model is a cubic subregion of the same size as that of the local model cut out from a global 3D image. In thi case, the model applies only on cubes containing a nodule. The model outputs a prediction on the nodule's malignancy (a number between 0 and 5).

All models are trained on the LUNA16 database, an online competition for lung nodule detection. This database comes from a public medical image database called LIDC/IDRI. This dataset consists of 888 3D computerized tomography lung images. Annotations from 4 radiologists are also provided on these images. The dataset contains a list of 1186 cancer nodules found on such images. The following information is given on each nodule: the ID of the image where it was found (patient identifier), its coordinates inside that image, its malignancy (a number between 0 and 5 resulting from the evaluation of the radiologists) and other characteristics.

All models are designed, trained and validated in Pyhton, using mainly Keras with Tensorflow as backend (two Pyhton libraries). All training sessions and tests are run on a Google Compute Engine virtual machine with an integrated GPU.

Each is model is tested using different parameter combinations and results are compared. The following conclusions are drawn:

The bad results obtained by the **global model** seem to indicate that the problem is too complex and subtle for such a shallow network (6 layers) and with so few parameters (around 270000 parameters), given the size of the data set.

The convolutional network used in the **local model** gives the best classification results (79% accuracy and 91% sensitivity in its best version). This proves that giving a local region-based diagnostic on the image makes the classification task much easier. The clustering algorithm used in this model did not work. It is believed that this is due to the tendency of the local positive prediction set to be connected.

Results from the **transfer learning model** are somewhat worse classification-wise than those from the local model but much better than those of the global model. Comparison of results for the different base models suggests that an increase in the number of parameters (and not the number of layers) of the model results in better diagnosis. The best results were obtained by VGG19 which achieved an accuracy of 63% and a sensitivity of 40% (almost twice as much as the others).

The **malignancy estimator** gives a mean squared error of 0.89 in its best version. Malignancy is measured on a scale of 0 to 5 and average nodule malignancy is approximately 3. It must be noted that malignancy labels are obtained through the subjetive evaluation of four radiologists, which often do not agree. It is hypothesized that this is probably the main reason for such a high error.

Three paths for future development of the project are suggested: A combination of the local model and the transfer learning model, the use of other nodule characteristics such as spiculation and lobulation in diagnosis and finally two possible developments of the clustering algorithm used in the local model.

A mis padres

Agradecimientos

Muchas gracias a mis padres por darme la oportunidad de poder estudiar esta carrera y por todo lo que me han enseñado.

También me gustaría agradecer a mi amigo Marc Bataillou su ayuda y esfuerzo con la instalación de la máquina virtual.

Por último, quisiera dar las gracias a mis directores de proyecto Jaime Boal y Santiago Moreno por haber estado siempre dispuestos a ayudar. Ambos me han guiado a distancia en este trabajo de fin de grado, lo cual incrementa su mérito. Por ello les estoy enormemente agradecido.

Índice general

Índice de figuras

1 Introducción

En este primer capítulo se presenta, en primer lugar, una introducción al aprendizaje automático y el diagnóstico asistido por ordenador. Además, se ofrece una descripción del estado del arte de la aplicación del aprendizaje automático para el diagnóstico a partir de imágenes médicas.

1.1. Introducción

Los datos recogidos por la Organización Mundial de la Salud (OMS) sobre las causas mas importantes de defunción en el año 2015 sitúan al cáncer de pulmón, junto con el de tráquea y de bronquios, como la quinta de esta lista. Estos tres tipos de cáncer se cobraron en 2015 la vida de 1,7 millones de personas. Además, esta cifra subió respecto a la registrada en el año 2000, que rondaba los 1,3 millones de defunciones [5].

El cáncer de pulmón es además difícil de diagnosticar. Un problema importante de las técnicas utilizadas para el diagnóstico de cáncer de pulmón es que tienen una tasa de falsos positivos muy grande, lo cual produce ansiedad al paciente y lleva en muchos casos a practicar otras pruebas y tratamientos de forma innecesaria. En [6] se obtuvo una tasa de falsos positivos de 96,4 % en pacientes a los que se practicó una exploración por tomografía computarizada de dosis baja. Se clasificó como positivo a aquellos pacientes sospechosos de cáncer de pulmón. Esto constituye un buen ejemplo de la necesidad de mejorar las técnicas de diagnóstico de este tipo de cáncer.

El objetivo de este proyecto es realizar distintos experimentos con varias técnicas de aprendizaje automático aplicadas al diagnóstico de cáncer de pulmón a partir de imágenes médicas.

1.2. Estado del arte

1.2.1. Diagnóstico asistido por ordenador (CAD)

El término diagnóstico asistido por ordenador (CAD) comprende el conjunto de técnicas y procedimientos médicos que ayudan a los doctores en la interpretación de contenidos

multimedia obtenidos de un paciente. Está tecnología permite al doctor interpretar toda la información disponible, ya que detecta detalles que podrían escapar al ojo humano. Otra gran ventaja del diagnóstico asistido por ordenador es que permite procesar grandes cantidades de pruebas médicas en poco tiempo. Las técnicas de CAD se aplican frecuentemente en el diagnóstico del cáncer de pulmón, cáncer de mama, cáncer colorrectal y en la cardiopatía congénita [7].

En general un procedimiento de CAD cuenta con varias etapas: preprocesado de las pruebas, segmentación y selección de zonas de interés, extracción y estructuración de datos y, por último, clasificación y evaluación [7]. La etapa de clasificación consiste en la aplicación de algoritmos a los datos obtenidos de las pruebas médicas para localizar anomalías o patrones extraños que pudieran indicar la presencia de enfermedad. Esta es la etapa relevante para este proyecto, en concreto cuando se aplican estos algoritmos sobre imágenes.

1.2.2. Aprendizaje automático aplicado al diagnóstico a partir de imágenes médicas

El aprendizaje automático (o *machine learning*) es la rama de las ciencias de la computación y de la inteligencia artificial cuyo objetivo es crear algoritmos que sean capaces de aprender a reconocer patrones en un conjunto de datos [8]. Según el grado de intervención humana en el proceso de aprendizaje de los algoritmos, existen 2 clases de aprendizaje automático: el no supervisado y el supervisado.

1.2.2.1. Aprendizaje automático no supervisado

El aprendizaje automático no supervisado [9] comprende las técnicas de *machine learning* en las que los datos no vienen etiquetados y es el propio algoritmo el que trata de encontrar un patrón en la distribución de los mismos, sin saber a priori nada sobre ella. Esto puede ser de gran utilidad para encontrar patrones en datos de varias dimensiones ya que no pueden visualizarse de ninguna forma y por tanto no se pueden encontrar estos patrones manualmente [10].

Muchas técnicas de aprendizaje no supervisado tienen un enfoque probabilista de los datos de entrada. Analizan los casos del conjunto de datos como varias instancias de una variable aleatoria y tratan de encontrar un modelo de densidad de probabilidad para los datos (con el método de máxima verosimilitud por ejemplo) [9].

Destacan como técnicas de aprendizaje automático no supervisado el *clustering* (el algoritmo de *k-means* por ejemplo). Los métodos de aprendizaje no supervisado son menos apropiados y por tanto menos utilizados para el diagnóstico asistido por ordenador porque no permiten incorporar el conocimiento experto de radiólogos y especialistas. Por esta razón, este documento se centra principalmente en artículos de aprendizaje supervisado.

1.2.2.2. Aprendizaje automático supervisado

En estos algoritmos, el conjunto de datos se proporciona ya etiquetado. Esto quiere decir que ya se conoce para cada caso de entrada el valor real de salida [11]. El algoritmo utiliza los casos proporcionados para, en el futuro, poder predecir el valor de salida de un punto no etiquetado.

El conjunto de datos se suele separar en tres partes: el conjunto de aprendizaje (en inglés *training set*), el conjunto de validación (en inglés *validation set*) y el conjunto de prueba (en

inglés *test set*). El primero sirve como experiencia de aprendizaje del algoritmo, durante el cual se ajustan los parámetros del modelo. El segundo sirve para ajustar otras características del modelo como la regularización o el umbral de clasificación. Finalmente, el conjunto de prueba sirve para analizar la precisión del modelo con puntos que no han aparecido en su proceso de aprendizaje [12].

Algunas técnicas de aprendizaje automático supervisado son la regresión lineal, la regresión logística, la clasificación por máquinas de vectores de soporte y las redes neuronales de varios tipos. En este documento se describen varios artículos, agrupados por técnica utilizada, que aplican estos algoritmos para el diagnóstico asistido por ordenador del cáncer en imágenes médicas.

1.2.2.2.1. Máquina de vectores de soporte (Support Vector Machines, SVM)

Una máquina de vectores de soporte es un algoritmo de aprendizaje supervisado de clasificación desarrollado por Vladimir Vapnik en los laboratorios de AT&T [13]. Se trata de una técnica similar a una regresión logística. Conviene analizar primero, por tanto, la regresión logística.

La regresión logística [10] es también un algoritmo de aprendizaje supervisado de clasificación. En su forma más básica este algoritmo determina si un caso de entrada pertenece a una clase (dando como salida un 1) o no (dando un 0). La lógica de este algoritmo es la siguiente. En este tipo de clasificación, al haber solo dos posibles valores resultado, el intento de ajuste de un hiperplano a los datos de entrada como se hace en regresión es inadecuado ya que éste daría un continuo de valores de salida y no los unos y ceros que se buscan. Para resolver este problema se aplica la función sigmoide sobre el valor de salida de un hiperplano de regresión, que acota los valores de salida entre 0 y 1 como se muestra en la Figura 1.1. El valor de dicha función se interpreta como una probabilidad de pertenecer a la clase 1 y por tanto se asigna la clase 1 a los puntos cuya salida sea mayor que 0,5 y la clase 0 a los demás. El modelo de clasificación se obtiene asignando un coste a la clasificación de cada punto según su clase verdadera, que forma parte de los datos de entrada. Los parámetros del modelo son aquellos que minimizan la suma de los costes de todos los datos de entrenamiento.

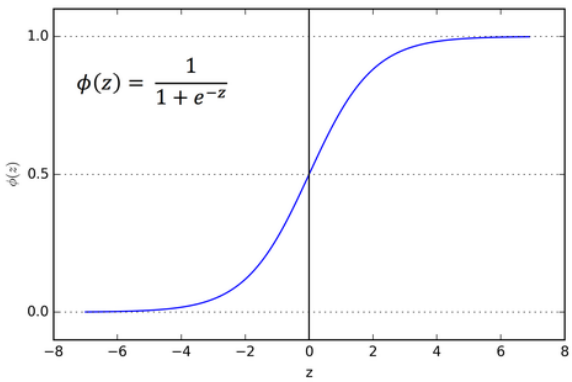

Figura 1.1. Función sigmoide [14]

La diferencia de una maquina de vectores de soporte con respecto a una regresión logística es el coste que se asigna a cada punto. En la máquina de vectores de soporte se asigna un coste nulo a aquellos puntos que están bien clasificados, lo cual quiere decir que están a una cierta distancia de la frontera de clasificación y del lado correcto de la misma. Por tanto, la maquina de vectores de soporte penaliza relativamente más a aquellos puntos que son ambiguos de

clasificar; aquellos que están cerca de la frontera de clasificación. Las ventajas de este método son, por un lado, que es computacionalmente más barato minimizar la función de coste al estar ésta linearizada, y, por otro, que se consigue una clasificación llamada de amplio margen [10]. En la Figura 1.2 se muestran diferentes barreras de clasificación para unos mismos datos. Las fronteras H2 y H3 clasifican bien los datos, sin embargo la frontera de SVM, H3, lo hace con mayor margen.

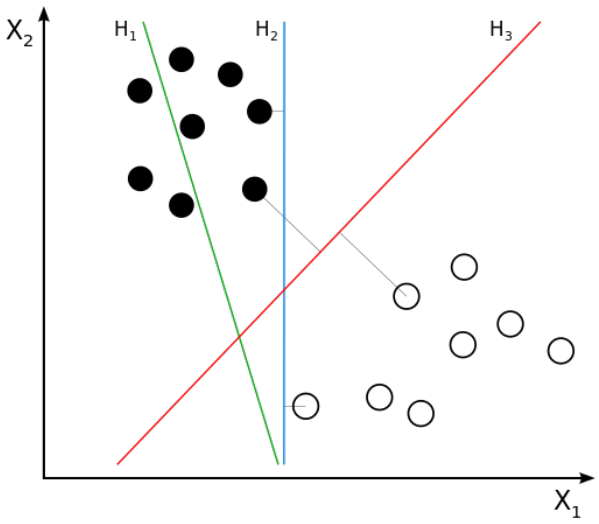

Figura 1.2. Frontera SVM (H3) frente a otras fronteras de clasificación [15]

En cuanto a la aplicación de este algoritmo para el diagnóstico automático del cáncer, destacan los artículos sobre el cáncer de pulmón y sobre el cáncer de mama, tanto los que aplican sobre mamografías como los que aplican sobre imágenes de biopsias de mama (véanse [16], [3], [17], [18]).

En [16] se aplica una SVM para clasificar tumores pulmonares en 3 clases (benigno, maligno poco desarrollado y maligno avanzado) utilizando características de imágenes de pulmón de tomografía por emisión de positrones (PET) y de CT. Las características de textura se extraen de las imágenes enteras (entropía, valor medio de captación estandarizada, nivel de grises). Por otro lado, se mide la heterogeneidad de los tumores presentes en la imagen. Para ello se extraen de forma automática volúmenes de interés que contienen un tumor a través de un algoritmo de umbralización y se calcula la heterogeneidad de las mismas a través del estadístico I de Moran. Finalmente se aplica la SVM sobre este conjunto de características. Se obtiene que la heterogeneidad del tumor es el mejor predictor de su estado de desarrollo.

En [3] se utiliza también la SVM para clasificar objetos detectados en imágenes de CT de pulmón como tumores o no. En primer lugar, se pide a un grupo de radiólogos que detecten objetos (tumores o no) sospechosos en las imágenes. Posteriormente se extraen 12 características de cada objeto detectado. Para seleccionar las mínimas características posibles que clasifiquen bien los objetos detectados, se realiza un análisis de correlaciones entre unas características y otras y se eligen las 5 más relevantes: por un lado, esfericidad y, por otro, 4 medidas de la distribución de intensidad (desviación estándar, media, mínima y máxima). Antes de clasificar los objetos se agrupan a través del algoritmo de *k-means* en varias clases y se selecciona un 20 % de cada una como representativo de la misma. Con este conjunto de datos reducido y las características extraídas, se entrena la SVM.

Las SVMs se aplican también al diagnóstico del cáncer en mamografías. Analizar estás técnicas es de interés para nuestro estudio ya que podrían extrapolarse al diagnóstico de cáncer de pulmón. En [17] los autores tratan de clasificar mamografías para la detección del cáncer. En este caso se realiza una extracción de regiones de interés (en inglés *Regions of interest*, ROI) de las mamografías en primer lugar. Después se elimina el ruido de las imágenes y se segmenta la zona afectada con tijeras inteligentes, que utilizan la técnica de detección de bordes *live-wire*. Esta técnica encuentra los bordes de un objeto en una imagen digital a través de un algoritmo de búsqueda del camino óptimo en un grafo ponderado [19]. Posteriormente, se extraen las características de cada recorte para finalmente clasificarlas. En este caso se aplica el algoritmo sobre 18 características que incluyen aspectos estadísticos (oblicuidad, curtosis, perímetro, área y desviación típica entre otras), de forma (circularidad, elongación) y de textura (entropía, correlación, contraste, momento de diferencia inversa) relevantes para el diagnóstico del cáncer. En [20] se investiga otro procedimiento de clasificación de mamografías basado en SVM. Se comparan los resultados con los de otros algoritmos de clasificación, como el de K vecinos más próximos y las redes neuronales artificiales.

Por otro lado, las SVMs también se han aplicado a la detección automática de microcalcificaciones a partir de mamografías [18]. Aunque su objetivo principal no es detectar cáncer de mama, la presencia de microcalcificaciones puede resultar un indicador importante para el diagnóstico de éste y por tanto el artículo es relevante para este estudio. Tras un filtrado paso alto de la imagen (para reducir la heterogeneidad del fondo), se recortan zonas cuadradas de la misma de una cierta dimensión, ajustada al tamaño típico de una microcalcificación. En este caso, se entrena la máquina de vectores de soporte directamente con estos recortes (sin extracción de características). Para elegir ciertos parámetros del modelo se utiliza el método de validación cruzada de 10 iteraciones.

1.2.2.2.2. Redes neuronales artificiales (Artificial neural networks, ANN)

Las redes neuronales artificiales (ANN), también conocidas como sistemas conexionistas, son un modelo computacional basado en el comportamiento de las neuronas biológicas [21]. Se trata de una técnica de aprendizaje supervisado de clasificación. Una red neuronal está formada por varias capas multineuronales. La primera capa se denomina capa de entrada y constituye el conjunto de datos de entrada del algoritmo. La última capa o capa de salida representa la estimación de nuestro algoritmo, siendo cada neurona la probabilidad de pertenencia a cada de una de las clases posibles. Entre la salida y la entrada se encuentran las llamadas capas ocultas, como se muestra en la Figura 1.3. Cada transición de una capa a la siguiente en una red neuronal viene regida por una serie de parámetros, que se van adaptando según la red aprende con los ejemplos de entrenamiento [10].

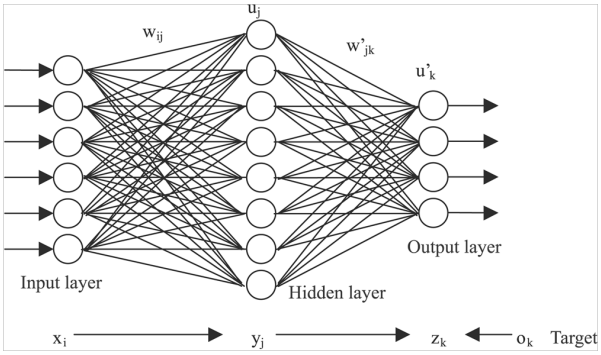

Figura 1.3. Esquema de red neuronal de 3 capas [22]

Esencialmente, el ajuste del modelo es un problema de optimización en el que se busca minimizar una función de coste que penaliza los errores de clasificación de dicho modelo comparando sus predicciones con las etiquetas de los ejemplos de entrenamiento. El objetivo es buscar en el espacio de los parámetros aquellos que minimicen la función de coste, es decir, que mejor se adapten a la experiencia de la red neuronal en el conjunto de entrenamiento.

Para lograr este objetivo, se utiliza un método iterativo llamado *gradient descent* que consiste en utilizar el gradiente de la función de coste para buscar su mínimo. La dirección del gradiente indica como ir "cuesta abajo" en la función que se busca minimizar como se observa en la Figura 1.4:

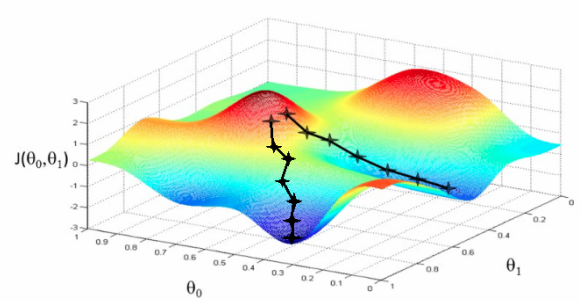

Figura 1.4. Método del gradiente [23]

Para aplicar este método se necesita la función de coste y su gradiente, que se obtienen sistemáticamente a través de los métodos de propagación hacia delante (del inglés *forward propagation*) para la función de coste y propagación hacia atrás (del inglés *backward propagation*) para el gradiente [10].

En el caso del diagnóstico asistido por ordenador, la entrada sería un conjunto de características de la imagen médica, o de una región de la misma previamente seleccionada y la salida sería la probabilidad de padecer un cáncer o no.

Las redes neuronales se aplican ampliamente en el diagnóstico del cáncer de pulmón. En [1], los autores tratan de crear una herramienta que ayude al especialista en la clasificación en benigno o maligno de nódulos pulmonares pero utilizando otro tipo de características. En primer lugar, incluye como entrada de la red neuronal varios parámetros clínicos como la edad, el sexo y el historial fumador. Se utilizan 7 parámetros clínicos y 16 radiológicos (extraídos de la imagen; concavidad y definición del borde por ejemplo). Además, las características radiológicas del nódulo no se extraen de forma automática sino que se obtienen a partir de la evaluación subjetiva de 3 radiólogos experimentados. Para comprobar el impacto positivo de la red neuronal sobre la decisión de los radiólogos se presentaron imágenes de alta resolución de CT teniendo y sin tener el diagnóstico de la red neuronal. Se muestra que la ayuda de la red neuronal mejora el diagnóstico de los médicos en la clasificación de los nódulos significativamente (pasando de 0.831 a 0.959 en el área bajo la curva ROC).

En [24], Suzuki et al. diseñan un sistema de clasificación de nódulos sacados de imágenes CT de pulmón de dosis baja basado en redes neuronales artificiales de entrenamiento masivo (MTANN). Las MTANN son filtros altamente no lineales basados en las ANN pero capaces de actuar directamente sobre imágenes. En este caso se aplican sobre subregiones de regiones de interés de las imágenes CT. Se disponen en paralelo 6 de estas MTANN, cada una especializada en la detección de un tipo de nódulo benigno. Las salidas de estas redes se introducen en una ANN de integración, cuya salida se interpreta como la probabilidad de malignidad. Por último,

se establece un umbral de clasificación y se realiza el diagnóstico. Un esquema del sistema completo puede apreciarse en la Figura 1.5.

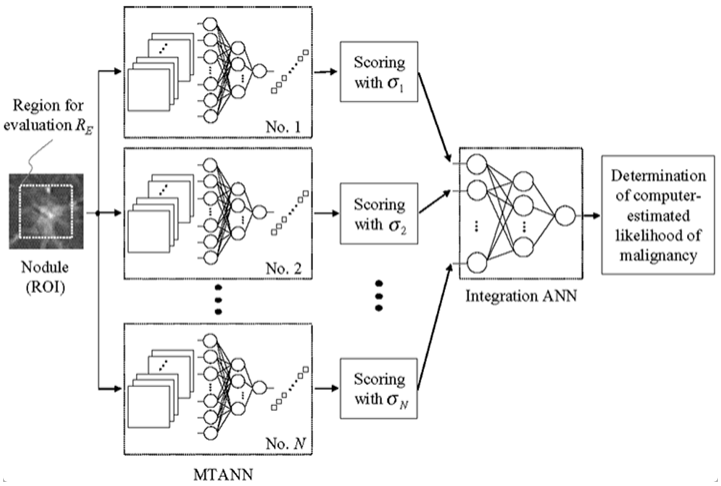

Figura 1.5. Esquema con varias MTANN y una ANN de integración [24]

Por otro lado, en [25] se utilizan parámetros estadísticos como la media, desviación típica, oblicuidad, curtosis y momentos centrales de quinto y sexto orden extraídos de zonas de interés recortadas de imágenes tomográficas de pulmón. En primer lugar se convierte la imagen en escala de grises a blanco y negro a través de un umbral fijado por el método de Otsu [26], que minimiza la varianza interna de los conjuntos de pixeles blancos y negros. Posteriormente se segmentan estas imágenes con operaciones morfológicas y se extraen las características de estos recortes. Se obtiene que la oblicuidad es el parámetro que más precisamente permite clasificar las imágenes.

En la literatura existente se pueden encontrar también numerosas aplicaciones de las ANN en el cáncer de mama, tanto en imágenes de biopsias de mama como en mamografías. En [27] se aplica una red neuronal de 2 capas intermedias para diagnosticar cáncer en imágenes de biopsias de mama. La técnica consiste, en primer lugar, en un preprocesamiento de la imagen utilizando algoritmos de Matlab Simulink que incluye pasar la imagen de RGB a escala de grises (posteriormente a blanco y negro), transformación divisoria y apertura morfológica entre otras. Posteriormente, se localizan los núcleos celulares presentes en la imagen aplicando detección de bordes y segmentación de Watershed. Una vez extraídos dichos núcleos, se extraen varias características de cada una de ellos: área, perímetro, solidez y excentricidad entre otras. Finalmente, se utilizan estas características como entrada de la red neuronal y se clasifica la biopsia.

Por otro lado, de forma similar, en [17] se aplica también una red neuronal artificial para el diagnóstico de cáncer de mama. En este caso no se procesan imágenes de biopsia de mama sino mamografías. Este artículo es de especial interés ya que utiliza y compara las ANN y las SVM. El proceso de tratamiento de las mamografías y de extracción de características ya ha sido explicado en la sección sobre SVM. La conclusión a la que se llega es que los resultados obtenidos son muy similares para ambos clasificadores (véase la tabla resumen para más detalle sobre estos resultados).

En [28], los autores clasifican mamografías en función de la densidad del seno. Esta aplicación no está directamente relacionada con el cáncer pero puede ser interesante para utilizar la densidad del seno como característica relevante en la detección del cáncer. Se prueban

distintos modelos de clasificación según número de clases y según distintas reglas de decisión entre los varios clasificadores y se comparan resultados.

1.2.2.2.3. Redes neuronales convolucionales (Convolutional Neural Networks, CNN)

Una red neuronal convolucional o CNN (*Convolutional Neuronal Network* por sus siglas en inglés) 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 [29].

La particularidad de este tipo de redes es que no requiere una definición manual de las características buscadas en la imagen (por comparación con el objeto que se busca detectar por ejemplo) sino que define de forma automática las características más relevantes para la detección de un objeto a través de su experiencia con los casos de entrenamiento [30]. Esto hace que las CNNs sean muy precisas para tareas de visión artificial como la detección de objetos en imágenes [29].

El funcionamiento de una CNN es, de nuevo, una particularización del de una red neuronal artificial en el que cada capa tiene una función específica. Se distinguen los siguientes tipos de capas:

- **Capa convolucional**: Esta capa aplica una operación llamada convolución (similar a la operación lógica NOR exclusivo) que compara varias imágenes representativas de las características buscadas con la imagen recibida para ver cómo de presentes están las mismas en dicha imagen. Esta es la etapa de extracción de características [31].

- **Capa de rectificación (ReLU)**: Esta capa activa o desactiva ciertas neuronas en función de la imagen recibida a la entrada. En cierta forma su labor es la de discretización de la imagen recibida, reduciendo los valores posibles de cada neurona. La rectificación suele seguir a la etapa de convolución [31].

- **Capa de reducción o pooling**: Esta capa reduce la dimensión de la imagen a través de una selección de los rasgos mas representativos de cada región de la imagen. Esta reducción consigue evitar el sobreaprendizaje (mejorando la capacidad de generalización) y disminuir el coste computacional para las siguientes capas. El coste de este procedimiento es evidentemente una pérdida de información. Esta capa suele ir después de la capa de rectificación [31].

- **Capa clasificadora**: Es la última capa de la red y sus neuronas dan los valores de salida del algoritmo, que se interpretan como probabilidades de pertenecer a cada una de las clases [31].

Varias iteraciones de las capas convolucional, de rectificación y de reducción preparan la información para la capa clasificadora como se ilustra en la Figura 1.6.

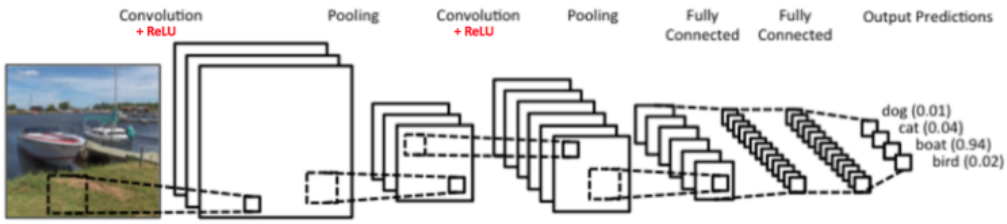

Figura 1.6. Red neuronal convolucional completa [32]

Las redes neuronales convolucionales también se aplican en la detección de cáncer de pulmón. En [2], Rossetto y Zhou utilizan la base de datos de imágenes CT de pulmón de la web Kaggle [33]. Su método aplica una red neuronal convolucional de dos formas distintas: por un lado sobre la imagen sin filtrar y por otro lado sobre la imagen pasada por un filtro gaussiano, el cual sirve para reducir el ruido y el detalle de la imagen. No se seleccionan regiones de interés. Cada red hace una predicción de la imagen y el resultado se obtiene a partir de un sistema de votación que solo da un caso como positivo si ambas redes lo clasifican como tal. En cualquier otro caso da un negativo (operación AND). Esto evidentemente reduce la tasa de falsos positivos. Se obtiene una precisión del 97,5 % con menos del 10 % de falsos positivos. En la Figura 1.7 se ilustra la estructura del sistema completo.

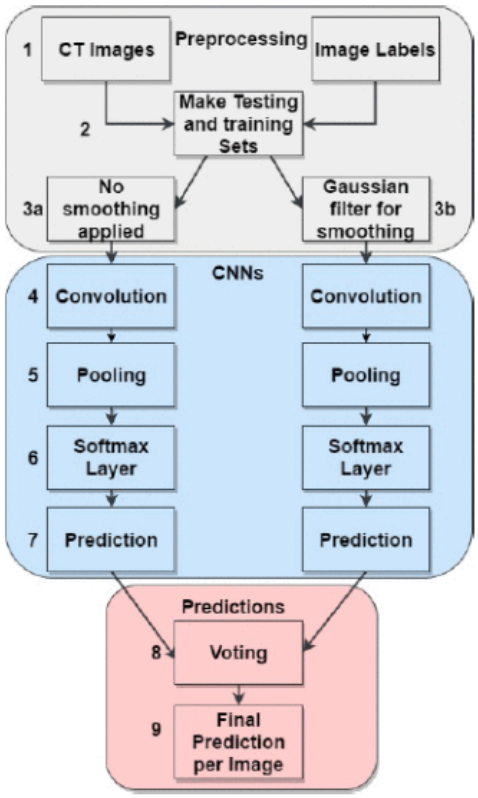

Figura 1.7. Esquema del sistema de clasificación de Rosetto y Zhou [2]

Las CNN aparecen también en el diagnóstico del cáncer de mama, tanto en imágenes histopatológicas como en mamografías. En [34] se consigue construir una red neuronal convolucional que clasifica imágenes histopatológicas de mama de cualquier nivel de magnificación. Las imágenes histopatológicas son imágenes de biopsias de distintos tejidos a escala microscópica, cuyo análisis permite el diagnóstico de enfermedades [35]. Estas imágenes

pueden tener un nivel de magnificación mayor o menor según el factor de ampliación con el que se tome la imagen. Se desarrollan 2 redes distintas: una que solo diagnostica malignidad y otra que extrae además el nivel de magnificación de la imagen.

Las CNN pueden utilizarse para la detección de microcalcificaciones en mamografías; lo cual aporta también al diagnóstico de cáncer de mama, ya que las microcalcificaciones mamarias son indicadores tempranos de cáncer. En [36], los autores investigan dos formas de preprocesamiento de las mamografías antes de aplicar la CNN en la detección de microcalcificaciones. Por un lado se estudia aplicar una transformación logarítmica; y por otro aplicar la raíz cuadrada a la intensidad de la imagen, lo cual elimina el ruido que depende de la intensidad. Ambos procedimientos mejoran los resultados de la CNN siendo los de la raíz cuadrada los mejores.

1.2.2.3. Tabla resumen

La Tabla 1.1 recoge los artículos mencionados que aplican disintas técnicas de *machine learning* para el CAD a partir de imágenes médicas.

Tabla 1.1. Resumen de los estudios analizados.

Ref.	Tipo de cáncer	Método	Entrada	Datos	Resultados
[27]	Mama	Red neuronal artificial (ANN)	Imágenes de biopsia de mama	569 imágenes	Precisión: 97,51 % (con BR) y 97,47 % (con SCG)
[34]	Mama	Red neuronal convolucional (CNN)	Imágenes histopatológicas de biopsia de mama	7909 imágenes (82 pacientes)	Tasa de reconocimiento media: 82,25 % (*single task* CNN) y 82,13 % (*multi-task* CNN)
[20]	Mama	Redes neuronales (ANN) Máquina de vectores de soporte (SVM) K vecinos más cercanos (KNN)	Mamografías	208 mamografías (166 pacientes)	Precisión: 85 % (1NN), 95 % (3NN), 65 % (SVM) y 75 % (ANN).
[17]	Mama	Máquina de vectores de soporte (SVM) Redes neuronales (ANN)	Mamografías	286 mamografías	Precisión: 96,91 % con SVM, 97,14 % con ANN
[18]	Mama (indirectamente)	Máquina de vectores de soporte (SVM)	Mamografías (perfil MLO)	76 mamografías (1120 calcificaciones)	Sensibilidad del 94 % a un ritmo de 1 falso positivo por imagen
[28]	Mama (indirectamente)	Red neuronal artificial (ANN)	Mamografías (perfil MLO)	377 Mamografías	Tasa de reconocimiento de 71,4 % (clasificación con 4 clases) y 96,9 % (clasificación con 2 clases)
[4]	Mama	K vecinos mas cercanos (KNN)	Imágenes de microcalcificaciones (MC) en mamografías	85 mamografías (100 microcalcificaciones)	Área bajo curva ROC: 0,96
[36]	Mama	Red neuronal convolucional (CNN)	Mamografías	1066 mamografías	Sensibilidad: 63,53 %
[2]	Pulmón	Red neuronal convolucional (CNN)	Imágenes de tomografía computarizada de pulmón	150.000 fragmentos de imagen de 1500 pacientes	Precisión: 97,5 %
[16]	Pulmón	Máquina de vectores de soporte(SVM)	Imágenes tomográficas de pulmón (PET/CT)	32 pacientes	Área bajo curva ROC: 0,907
[25]	Pulmón	Redes neuronales artificiales (ANN)	6 características extraídas de las regiones de interés	Imágenes de tomografía computarizada de pulmón de 155 pacientes	Precisión: 93,3 % Especificidad: 100 % Sensibilidad: 91,4 %
[1]	Pulmón	Redes neuronales artificiales (ANN)	23 características (7 parámetros clínicos y 16 radiológicos)	Imágenes de tomografía computarizada de pulmón de 155 pacientes	Área bajo curva ROC: 0,951
[3]	Pulmón	Máquina de vectores de soporte (SVM)	5 características de objetos detectados en imágenes CT de pulmón	801 objetos (81 tumores)	Sensibilidad del 100 % con 95,9 % de tasa de descubrimientos falsos
[24]	Pulmón	Redes neuronales artificiales (ANN) Redes neuronales de entrenamiento masivo (MTANN)	Subregiones de regiones de interés de imágenes CT de pulmón de dosis baja	489 nódulos (73 malignos) obtenidos de 415 pacientes	Área bajo curva ROC: 0,882 (test round robin)

2

Redes neuronales convolucionales

En este capítulo se explican en detalle el interés, el funcionamiento y los componentes de las redes neuronales convolucionales.

2.1. Motivación

Las redes neuronales convolucionales son una variante del perceptrón multicapa inspiradas en la biología de la corteza visual del cerebro animal. La investigación de Hubel y Wiesel [37] sobre la corteza visual primaria (o corteza estriada) del mono y el gato descubrió ciertos mecanismos del funcionamiento de ésta que aprovechan la correlación espacial existente en las imágenes para obtener una sensibilidad a los patrones espaciales del campo visual y permitir así una interpretación de la imagen percibida. Entre otras cosas, Hubel y Wiesel descubrieron la existencia de, por un lado, grupos de células sensibles a regiones específicas del campo visual, y, por otro, grupos de células sensibles a ciertos patrones independientemente de la posición de éstos en el campo visual [38]. Las redes neuronales convolucionales emulan el funcionamiento de la visión animal utilizando estos principios.

Los modelos con los que se ha experimentado en este proyecto se basan principalmente en las redes neuronales convolucionales por lo que es conveniente una explicación más detallada de su funcionamiento. En la sección sobre el estado del arte de este documento se ofrece una primera explicación de las redes convolucionales o CNN y sus capas más comunes. En las siguientes secciones se ofrece una explicación detallada sobre dichas capas y la estructura de las CNN.

2.2. Capas convolucionales

Para simplificar las explicaciones, en esta sección se explican las redes convolucionales en 2 dimensiones.

Las capas convolucionales toman como entrada una imagen (formalmente una matriz de dimensiones $m \times n$) y dan como salida varias imágenes nuevas correspondientes a pasar la imagen por los distintos filtros de la capa. La salida es por tanto un conjunto de imágenes llamadas mapas de características (de dimension similar a la de entrada), correspondiendo cada imagen de salida a la de entrada pasada por uno de los filtros como se muestra en la Figura 2.1. Para que una capa convolucional quede definida, simplemente se ha de especificar el numero de filtros, el tamaño de los filtros (en inglés *kernel size*) y el paso con el que se desplaza por la imagen.

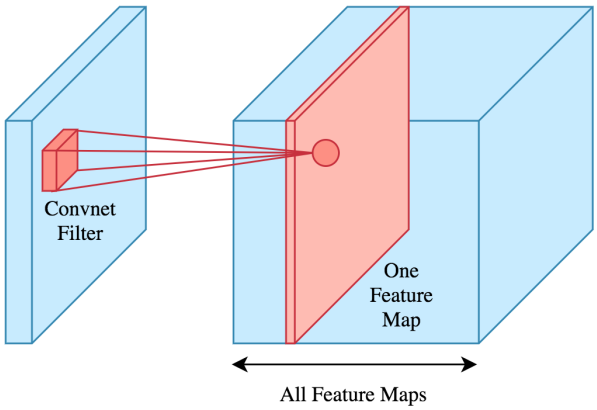

Figura 2.1. Esquema de formación de mapas de características a partir de los filtros [39]

Cada uno de los filtros constituye un conjunto de parámetros de la capa. Es importante entender que no es necesario elegir y codificar estos filtros para detectar ciertos patrones, sino que los filtros forman parte de los parámetros del modelo y se generan con el entrenamiento. Así, los filtros representan los patrones más característicos o que mejor permiten detectar aquello que se busca con la red convolucional. Un filtro se puede ver como una señal discreta donde los valores son sus parámetros. El número de parámetros del filtro es igual al número de píxeles del mismo.

Típicamente, las capas convolucionales resultantes tras el entrenamiento detectan patrones más o menos complejos según lo lejos que estén de la entrada de la imagen. Las primeras capas sirven para detectar patrones muy generales y muy simples como bordes, contornos, puntos y líneas mientras que las capas más profundas combinan la información de estas capas para detectar características más particulares del objeto a detectar [40]. En la Figura 2.2 se muestra el patrón que detecta cada filtro en una red convolucional entrenada para detectar caras humanas. Se puede apreciar que las últimas capas convolucionales son sensibles a partes de la cara mientras que la primera capa detecta bordes y sombras.

El nombre de éstas capas proviene de la operación matemática producto de convolución de dos funciones, que es la operación que permite obtener el mapa de características de la imagen a través de un filtro. Esta operación consiste en desplazar el filtro por las distintas posiciones de la imagen y realizar en cada posición la suma de los productos elemento a elemento del filtro y la región correspondiente de la imagen. En la Figura 2.3 se muestra un esquema de la operación.

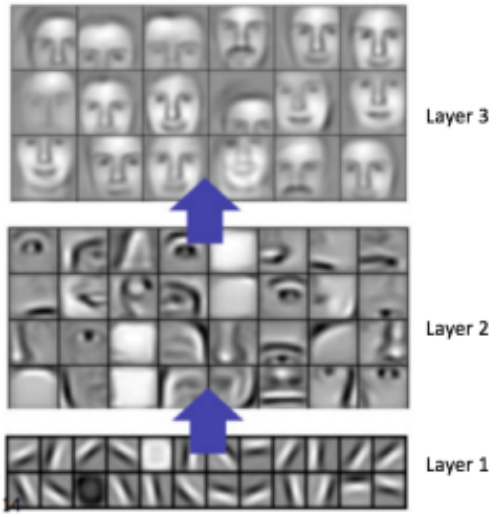

Figura 2.2. Patrones detectados por capas convolucionales más o menos profundas [40]

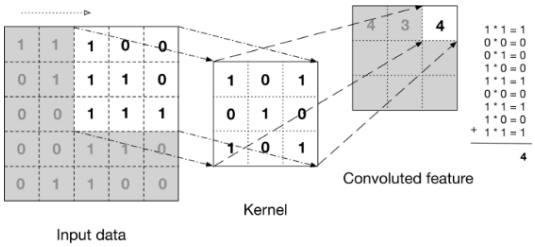

Figura 2.3. Esquema del producto de convolución [41]

Formalmente, dicha operación es el producto de convolución discreto de la imagen x y el filtro k (prolongando ambas adecuadamente hasta el infinito, siempre y cuando el paso sea de una unidad) [38]:

$$C[m,n] = (x * k)[m,n] = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} x[u,v]k[m-u,n-v] \tag{2.1}$$

Las neuronas de salida de una capa convolucional aplican posteriormente una función de activación f. Así, el mapa de características de la imagen x para el filtro k (de parámetros W^k y sesgo b^k) es una imagen h_{ij} de valores en la posición i, j:

$$h_{ij}^k = f((W^k * x)_{ij} + b^k) \tag{2.2}$$

Las capas convolucionales son interesantes en el reconocimiento de imágenes por varias razones. En primer lugar, a diferencia de otros tipos de redes neuronales, las CNN tienen en cuenta las posiciones de los datos de entrada ya que sus filtros aplican sobre subregiones de la imagen, captando así relaciones locales entre píxeles contiguos o cercanos para detectar contornos u otros patrones. Esto hace que las capas convolucionales tengan baja conectividad ya que cada neurona de la capa de salida está conectadas únicamente con las de una subregión de la imagen de entrada [38].

En segundo lugar, el hecho de que estas capas utilicen un filtro fijo que se desplaza por la imagen en la convolución hace que los filtros puedan detectar un patrón independientemente de la posición del mismo en la imagen de entrada. Además, el hecho de que las neuronas de salida solo tomen como entrada una parte de la imagen y que compartan todas los mismos parámetros (los del filtro, fijos) reduce enormemente el número de parámetros [38].

Todo lo explicado anteriormente se generaliza fácilmente a 3 dimensiones, que es el caso de aplicación de este proyecto.

2.3. ReLU

Las ReLU son neuronas cuya función de activación es la función rectificadora. Dicha función simplemente pone los valores negativos a cero y aplica la función identidad sobre los positivos. La salida de la función rectificadora viene dada por:

$$f(x) = max(0, x) \tag{2.3}$$

Tradicionalmente, en las CNN se han utilizado la función sigmoide y la tangente hiperbólica como funciones de activación de las neuronas. Sin embargo, estas funciones tienen el problema de la saturación. Estas funciones tienen un gradiente casi nulo en valores extremos, lo cual frena el proceso de entrenamiento al estar éste basado en el método del gradiente explicado en la sección anterior.

En [42], Glorot et al. demostraron que la ReLU, además de modelizar mejor las neuronas biológicas, ofrece un funcionamiento mejor y un comportamiento mejor en entrenamiento que la tangente hiperbólica (comúnmente usada también en tareas de visión artificial), sobre todo en redes de arquitectura profunda (con más de 3 capas ocultas). Por esta razón, en los modelos utilizados en este proyecto se utiliza la función rectificadora como función de activación de la salida de las capas convolucionales.

2.4. Capa de *pooling* o reducción

La capa de *pooling* o reducción es una forma de reducción de la dimensión de los datos para las capas posteriores [38]. Dichas capas simplemente dividen la imagen de entrada en subimágenes de mismo tamaño no solapadas y selecciona el valor máximo de cada una. La reducción en la dimensión de la imagen de entrada viene determinada por la dimensión de la ventana de *pooling*. Por ejemplo, si la entrada es una imagen de dimensión $n \times n$ y la ventana de *pooling* es de dimension $m \times m$ (siendo n un múltiplo entero de m), la imagen de salida tiene dimensión $(m/n) \times (m/n)$.

La utilidad de estas capas reside, por un lado, en la reducción del coste computacional como se ha explicado y, por otro, en que proporciona una invariabilidad a la traslación de los patrones. Si por ejemplo se traslada ligeramente un contorno en una imagen, el mapa de características del filtro que detecta dicho contorno aparecería trasladado también. Sin embargo, al seleccionar el máximo de esa zona del mapa de características a través de la capa de *pooling*, la salida de dicha capa sería la misma en ambos casos. Se obtiene así una independencia de la posición del patrón detectado [38].

2.5. *Dropout*

El *dropout* no es una capa propiamente dicha sino una modificación que se puede añadir sobre una capa en la fase de entrenamiento. El *dropout* no forma necesariamente parte de las redes convolucionales ni es exclusivo de éstas. Se añade esta sección debido a que es un recurso que se utiliza frecuentemente en este proyecto.

Al aplicar *dropout* sobre una capa se desconectan aleatoriamente un porcentaje de sus neuronas (su valor se pone a 0) en cada imagen procesada durante el entrenamiento. Esto mejora la capacidad de generalización de una red neuronal ya que la obliga a entrenarse ignorando parte de la información de la que se dispone. De alguna forma, el *dropout* obliga a la red a aprender patrones más robustos y generales y a no sobreaprender detalles propios del conjunto de entrenamiento [43]. El *dropout* se utiliza en las redes neuronales de este proyecto como medida contra el sobreajuste.

3

Descripción de los modelos

En este capítulo se explican en detalle los diseños de los cuatro modelos con los que se experimenta en el proyecto.

Todos los modelos de este proyecto se construyen, entrenan y validan en Python. Se utiliza principalmente la librería Keras [44] , con la librería Tensorflow [45] como programa de soporte. Keras es una librería de Python que sirve como envoltorio de Tensorflow para facilitar el diseño de redes neuronales para *deep dearning*. En este proyecto se desarrollan 4 modelos de diagnóstico automático diferentes. Los 3 primeros sirven para dar un diagnóstico global de un paciente según contenga un nódulo pulmonar o no. El modelo sobrante sirve para, una vez detectado un nódulo, hacer una estimación de su malignidad.

A continuación se describe en detalle el funcionamiento de cada uno de los modelos.

3.1. Modelo global

El principio del modelo global consiste en tratar de dar un diagnóstico basado en un análisis global de la imagen. Este primer modelo consiste simplemente en aplicar sobre la imagen 3D en cuestión una red neuronal convolucional tridimensional cuya salida directamente indica si el paciente tiene un tumor o no. En la Figura 3.1 se muestra un esquema del modelo.

La red neuronal `global_model` contiene, en primer lugar, dos bloques seguidos formados cada uno de ellos por una capa convolucional seguida de una capa de reducción o *pooling* con un cierto *dropout*. Después se añade una última capa convolucional, de nuevo con un cierto *dropout*. Antes de la capa clasificadora de la salida se utiliza una capa *flatten* que simplemente elimina la forma tensorial de los datos y los convierte en una capa de una dimensión (un vector). En todas las capas convolucionales se utiliza la función rectificadora como función de activación. En la Figura 3.2 se muestra el detalle de una de las versiones con las que se experimenta de este modelo.

El tamaño de los filtros de la primera capa se elige para detectar nódulos de diámetro real medio 8 mm en volúmenes reales de 30 cm x 30 cm x 40 cm. Como se detalla en la sección de

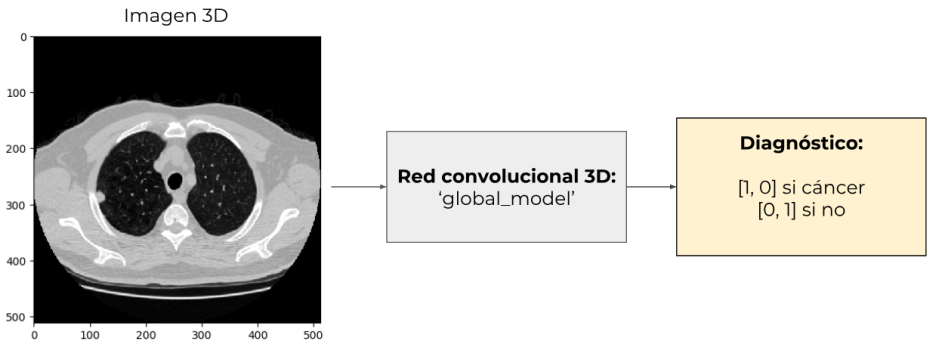

Figura 3.1. Esquema del modelo global

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv3d_1 (Conv3D)            (None, 20, 47, 191, 191)  8020
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 20, 23, 95, 95)    0
_________________________________________________________________
dropout_1 (Dropout)          (None, 20, 23, 95, 95)    0
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 17, 16, 88, 10)    243210
_________________________________________________________________
max_pooling3d_2 (MaxPooling3 (None, 17, 8, 44, 5)      0
_________________________________________________________________
dropout_2 (Dropout)          (None, 17, 8, 44, 5)      0
_________________________________________________________________
conv3d_3 (Conv3D)            (None, 14, 1, 37, 10)     12810
_________________________________________________________________
dropout_3 (Dropout)          (None, 14, 1, 37, 10)     0
_________________________________________________________________
flatten_1 (Flatten)          (None, 5180)              0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 10362
=================================================================
Total params: 274,402
Trainable params: 274,402
Non-trainable params: 0
_________________________________________________________________
```

Figura 3.2. Ejemplo de red neuronal del modelo global

los datos, cada corte de las imágenes 3D perpendicular al eje z es de dimensión 512 píxeles x 512 píxeles. Por tanto, se elige un tamaño de filtro de 8 píxeles para detectar adecuadamente los contornos e irregularidades de dichos nódulos, cuya dimensión en la imagen es de unos 13 píxeles.

3.2. Modelo local

El modelo local está basado en un análisis local de la imagen por zonas. Al estar tratando de encontrar tumores un diámetro medio de 8 mm en imágenes que representan un volumen real de 30 cm x 30 cm x 40 cm, puede parecer adecuado un sistema basado en una red neuronal que aplique sobre subregiones pequeñas de la imagen. Esto modelo consta de 2 partes: por un lado la red neuronal convolucional 3D `nodule_detector` y por otro el algoritmo `check_cluster`. En la Figura 3.3 se muestra el flujo de información del modelo, desde la imagen 3D de entrada hasta el diagnóstico.

La red neuronal `nodule_detector` toma como entrada una subregión recortada de la imagen global y devuelve una predicción sobre si existe un tumor en dicho "cubo" o no. El diámetro medio de los nódulos es de 8 mm y por tanto su dimensión media en píxeles es de aproximadamente 12 en las dimensiones x e y (menor en z debido a la menor definición de las imágenes en esta dirección). Se elige por tanto una dimensión de 30 x 30 x 30 vóxeles para

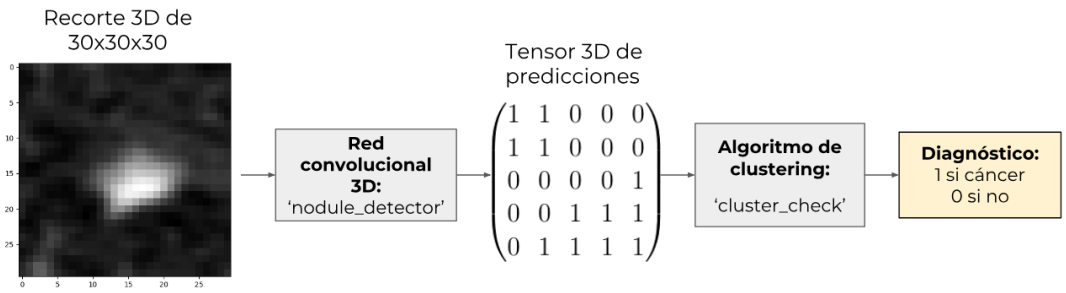

Figura 3.3. Esquema del modelo local

la dimensión de entrada de la red neuronal, de forma que se se aprecie bien el nódulo de la imagen al ser éste grande frente al tamaño del recorte de la imagen. En Figura 3.4 se detallan las capas para una de las versiones del modelo:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv3d_1 (Conv3D)            (None, 40, 27, 27, 27)    2600
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 40, 13, 13, 13)    0
_________________________________________________________________
dropout_1 (Dropout)          (None, 40, 13, 13, 13)    0
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 37, 10, 10, 20)    16660
_________________________________________________________________
max_pooling3d_2 (MaxPooling3 (None, 37, 5, 5, 10)      0
_________________________________________________________________
dropout_2 (Dropout)          (None, 37, 5, 5, 10)      0
_________________________________________________________________
conv3d_3 (Conv3D)            (None, 36, 4, 4, 10)      810
_________________________________________________________________
max_pooling3d_3 (MaxPooling3 (None, 36, 2, 2, 5)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 36, 2, 2, 5)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 720)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1442
=================================================================
Total params: 21,512
Trainable params: 21,512
Non-trainable params: 0
_________________________________________________________________
```

Figura 3.4. Detalles de las capas de la red neuronal del modelo local

El siguiente paso en el procesamiento de información es el de trasladar por toda la imagen global dicha red neuronal, dando una predicción para cada región. El detector de nódulos se desplaza por la imagen en traslaciones a lo largo de cada dimensión (por cada elemento de cada fila de cada corte). El paso de traslación se toma como 5 en todas direcciones por reducir el coste computacional con respecto a un paso menor, que aportaría mejor definición. La salida de dicho procedimiento es un tensor tridimensional de unos (si hay un tumor en la ventana de esa posición) y ceros (si no) con las predicciones para cada región de la imagen. Este paso se consigue con a la función `rolling_box` del código.

Una vez generado el tensor de predicciones, se aplica el algoritmo `cluster_check` del esquema. Dicho algoritmo trata de agrupar los unos de dicho tensor en función de la distancia entre ellos a través de un algoritmo de *clustering*. Como el paso de desplazamiento del detector de nódulos por la imagen es pequeño, un tumor en la imagen pertenece a varias ventanas de predicción y por tanto aparecerá representado en el tensor de predicciones como un conjunto de unos más o menos concentrados. Con el algoritmo de *clustering* se intenta agrupar los unos

de las mismas zonas para posteriormente aplicar un criterio de diagnóstico para dicho grupo de unos. Dicho criterio depende del número de unos del *cluster* en cuestión.

El algoritmo de *clustering* jerárquico aglomerativo [46] utilizado funciona de la siguiente manera. En cada iteración, se calculan las distancias entre todos los elementos (ya sean *clusters* o puntos individuales) dos a dos y se agrupan bajo un mismo *cluster* los dos elementos más próximos, que forman un nuevo *cluster*. La configuración de partida del algoritmo es la formada por los puntos a agrupar individualmente. Se repite el paso explicado hasta que estén todos los puntos agrupados bajo el mismo *cluster*. Para que el algoritmo quede determinado basta con dar el conjunto de puntos a agrupar y sus coordenadas, la definición de distancia entre puntos (llamada métrica del algoritmo) y la definición de distancia entre *clusters* (llamado método del algoritmo). En este caso se utiliza la **distancia euclídea** según las coordenadas en el tensor como **métrica** y la **distancia euclídea mínima** como **método**. De esta forma, al calcular la distancia entre dos *clusters* se tomaría la menor posible entre dos de sus puntos, pertenecientes cada uno de ellos a uno de los *clusters*.

La librería Scipy tiene un método que permite detener el algoritmo cuando las distancias entre los dos siguientes elementos a agrupar es mayor que una cierta distancia de truncamiento. En este modelo se utiliza esta funcionalidad para tratar de agrupar los unos pertenecientes al mismo nódulo.

El método completo es el siguiente:

1. Extracción de las coordenadas en el tensor de predicciones de todos los unos.
2. Agrupación por *clustering* aglomerativo de dichos unos en función de su distancia euclídea (métrica) y juntando *clusters* por sus distancias mínimas (método). Se agrupan los unos hasta una cierta d_{max} como se explica en el párrafo anterior.
3. Se comprueba si alguno de los *clusters* resultantes tiene más de N unos. En caso de que sí, se considera que dicho *cluster* representa un tumor y se da un diagnóstico positivo. En caso contrario se da un diagnóstico negativo.

Los resultados obtenidos a partir de esta técnica dependen en gran medida de los parámetros d_{max} y N. Para la decisión de d_{max} se toma que el algoritmo agrupe los unos que estén como mucho separados por un hueco en una dirección no diagonal. Se agrupan únicamente los unos alineados en una diagonal si son contiguos. De esta forma se intenta que los *clusters* finales no tengan muchos huecos. Así, se establece:

$$d_{max} \simeq 2 \tag{3.1}$$

Teniendo en cuenta el nivel de sensibilidad del detector de nódulos (ver sección de resultados), es poco probable que aparezcan los falsos positivos suficientes y en las posiciones necesarias como para aislar distintas zonas de las predicciones de un mismo tumor y por tanto hacer que los *clusters* resultantes no lleguen a N unos. El cálculo exacto de esta probabilidad es un problema matemático de gran complejidad.

La elección de N se hace basándose en un cálculo estadístico. En primer lugar se puede calcular el número n de ventanas que contendrán un nódulo de tamaño medio, siempre y cuando esté suficientemente alejado de los bordes (podemos suponer que es el caso debido a que las imágenes contienen un contorno bastante grande alrededor de los pulmones). Se

considera que un nódulo entra dentro de una de las ventanas en una dirección si entra dentro por lo menos la mitad del nódulo. Dado que el paso de desplazamiento de la ventana es 5, se estima n como:

$$n \simeq \left(\frac{30}{5}\right)^3 = 216 \tag{3.2}$$

Para hacer una estimación de N supongamos que, estando el nódulo contenido en estas n ventanas, el número de predicciones positivas en estas n predicciones es una variable aleatoria N_p igual a la suma de n variables aleatorias X_i independientes e idénticamente distribuidas según una distribución de Bernoulli de paramétro p. Dichas variables toman el valor 1 si la predicción correspondiente detecta el nódulo y 0 si no. El parámetro p se estima a partir de la sensibilidad obtenida en la validación del detector de nódulos (ver sección de resultados). La sensibilidad es la probabilidad empírica de detectar un nódulo y se calcula como la relación entre los verdaderos positivos (nódulos detectados) y la suma de verdaderas positivos y falsos negativos (nódulos no detectados). Se obtiene una sensibilidad de 0.91 en el mejor detector de nódulos (el utilizado para la segunda fase de este modelo).

$$N_p = \sum_{i=1}^{n} X_i \qquad X_i \sim Be(p) \text{ i.i.d} \qquad \forall i = 1, 2, ..., n \tag{3.3}$$

Así la variable aleatoria N_p sigue una distribución binomial de parámetros n=216 y p=0.91. La esperanza de N_p se calcula como $n \times p$ y es aproximadamente igual a 197:

$$N_p \sim Bi(n, p) \qquad E[N_p] = np \simeq 197 \tag{3.4}$$

Ahora supongamos que esas n ventanas no contienen un nódulo. El número de predicciones positivas es de nuevo una variable aleatoria N_q igual a la suma de n variables de Bernoulli independientes e idénticamente distribuidas de parámetro p'. Análogamente, el parámetro p' se estima a partir de la probabilidad empírica de dar un positivo en un caso negativo. Se toma el cociente entre falsos positivos y la suma de falsos positivos y verdaderos negativos. Se obtiene un valor de p' de 0.33 . De nuevo:

$$N_q = \sum_{i=1}^{n} X_i \qquad X_i \sim Be(p') \text{ i.i.d} \qquad \forall i = 1, 2, ..., n \tag{3.5}$$

N_p también sigue una distribución binomial, esta vez de parámetros n y p'. La esperanza del número de predicciones positivas es aproximadamente igual a 71:

$$N_q \sim Bi(n, p') \qquad E[N_q] = np' \simeq 71 \tag{3.6}$$

La esperanza de N_p nos da una idea (el cálculo es indicativo) sobre el valor esperado del número de unos en un *cluster*, suponiendo que se agrupan todas las predicciones positivas asociadas a un nódulo. Por otra parte, la esperanza de N_q permite decir que es un muy poco probable que haya un número de unos mayor que 85 (valor correspondiente a dos desviaciones típicas por encima de la media de la variable aleatoria N_q) en el conjunto de

ventanas que contienen un punto donde no hay un nódulo. Se toma por tanto $N>170$ como criterio de diagnóstico positivo sobre los *clusters* formados. En el cálculo anterior se asume que se agruparían todas las predicciones positivas asociadas al mismo nódulo. En la práctica, los nódulos podrían segmentarse si se agrupan varios falsos negativos contiguos. Por otra parte, podrían agruparse dos zonas sospechosas pero sanas con abundancia de falsos positivos y formar un *cluster* lo suficientemente grande como provocar un falso positivo en el diagnóstico final del paciente. El razonamiento anterior sirve únicamente de estimación del parámetro N.

Estos valores dan un punto de partida para las pruebas del algoritmo. En las pruebas realizadas se hacen variar para obtener la mejor combinación.

3.3. Modelo *transfer learning*

En la práctica, cuando se quiere desarrollar una red neuronal convolucional para clasificar imágenes, rara vez se entrena una red desde cero ya que no se suele disponer de una base de datos lo suficientemente grande como para conseguir buenos resultados. La práctica más común en estas situaciones es utilizar una técnica llamada *transfer learning*. Esta técnica consiste en utilizar como modelo de base una red neuronal ya entrenada sobre una gran base de datos de imágenes y adaptarla para clasificar el tipo de imágenes concreto del problema en cuestión. Este procedimiento nace de la observación de que las primeras capas de una gran red neuronal convolucional son detectores de patrones muy simples y generales, que pueden servir para detectar casi cualquier tipo de objeto en una imagen. En un modelo de *transfer learning*, para adaptar la red neuronal de base se suelen llevar a cabo las dos o alguna de las dos siguientes modificaciones [47]:

1. Se elimina la última capa del modelo de base y **se añade un clasificador propio** con el número de clases requerido. Durante el entrenamiento se calculan los parámetros del nuevo clasificador. Este procedimiento es necesario siempre que el número de clases del nuevo conjunto de imágenes difiera del original.

2. Partiendo de los parámetros del modelo de base, se sigue entrenándolo con la base de datos del problema en cuestión, alterando típicamente solo algunas de sus capas y "congelando" las demás. Lo más común es alterar las últimas capas del modelo, ya que son las que detectan patrones más específicos de cada tipo de imagen. Con esto lo que se consigue es variar parte del modelo para que sus parámetros se calibren más adecuadamente a las imágenes del problema, alterando ligeramente el resultado de su entrenamiento anterior. A esta técnica se la llama comúnmente ***fine-tuning*** (en español "ajuste fino").

En este proyecto se experimenta con distintas combinaciones de modelos de base, con mayor o menor grado de alteración o número de capas que se "descongelan" en el entrenamiento. Además, este tercer modelo, a diferencia de los demás, no aplica sobre imágenes tridimensionales sino sobre imágenes **bidimensionales**. La razón principal por la que se aplica dicha técnica sobre imágenes 2D es debido a que existen más modelos disponibles en internet. Los problemas de visión artificial con imágenes 3D son menos populares que en 2D. Para este modelo, las imágenes planas se obtienen a partir de las imágenes tridimensionales mediante secciones perpendiculares al eje z (eje que va de pies a cabeza). En la Figura 3.5 se muestra un esquema del modelo *transfer learning*.

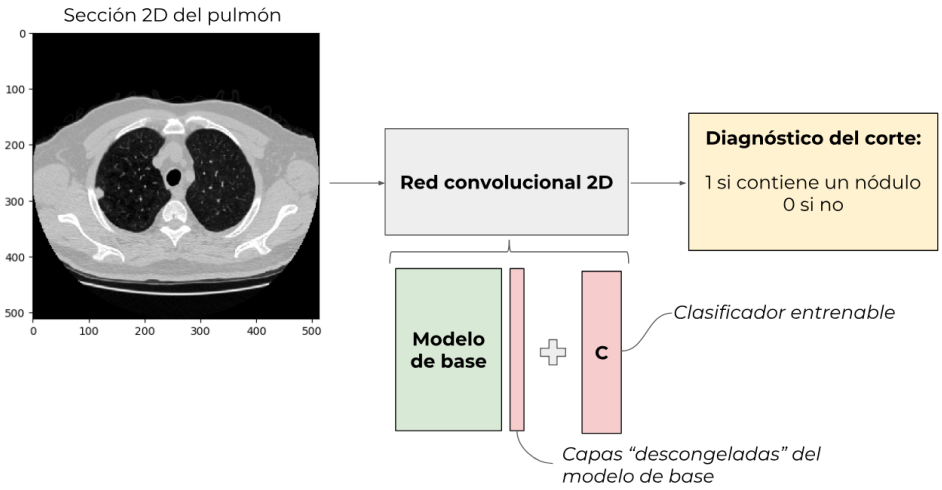

Figura 3.5. Esquema del modelo *transfer learning*

En la Figura 3.6 se muestra en detalle la estructura por capas de una de las versiones de la red neuronal de este modelo con las que se experimenta (ver secciones de entrenamiento y resultados).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
xception (Model)             (None, 10, 10, 2048)      20861480
_________________________________________________________________
flatten_1 (Flatten)          (None, 204800)            0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              209716224
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 2050
=================================================================
Total params: 230,579,754
Trainable params: 209,722,370
Non-trainable params: 20,857,384
_________________________________________________________________
```

Figura 3.6. Detalle de las capas del modelo *transfer learning*

En la Figura 3.7 se muestra una tabla con las propiedades de los 3 modelos (y otros contenidos en Keras applications) con los que se realizan pruebas para el tercer modelo: Xception, ResNet50 y VGG19.

3.4. Estimador de malignidad

El objetivo de este cuarto modelo, a diferencia de los otros tres, no es dar un diagnóstico definitivo del paciente sino estimar la malignidad de un nódulo una vez encontrado. De nuevo, se ataca el objetivo utilizando una red neuronal convolucional 3D. La entrada del estimador de malignidad es, al igual que en el segundo modelo, un recorte tridimensional cúbico de dimensión 30x30x30. En este caso, el modelo se aplica únicamente sobre "cubos" que sí contienen un nódulo. La salida es un número real entre 0 y 5 que representa la estimación de malignidad. Como se explica en la sección de datos, las etiquetas de malignidad de los tumores están comprendidas entre 0 y 5 y se obtienen a partir de la valoración de 4 radiólogos. En la Figura 3.8 se muestra un esquema del modelo.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88
DenseNet121	33 MB	0.745	0.918	8,062,504	121
DenseNet169	57 MB	0.759	0.928	14,307,880	169
DenseNet201	80 MB	0.770	0.933	20,242,984	201

Figura 3.7. Modelos de base para *transfer learning* de Keras Applications [48]

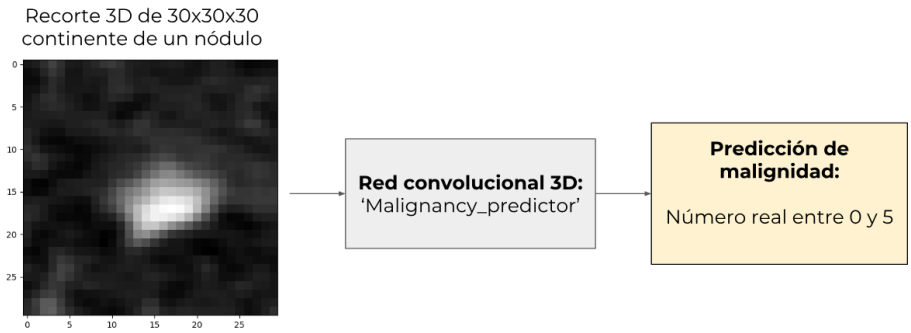

Figura 3.8. Esquema del estimador de malignidad

En la Figura 3.9 se muestra el detalle sobre las capas que forman la red neuronal de este modelo. Los parámetros se dan para una de las versiones con la que se experimenta (ver sección de resultados).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv3d_1 (Conv3D)            (None, 20, 21, 21, 21)    20020
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 10, 10, 10, 21)    0
_________________________________________________________________
dropout_1 (Dropout)          (None, 10, 10, 10, 21)    0
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 3, 3, 3, 20)       215060
_________________________________________________________________
max_pooling3d_2 (MaxPooling3 (None, 1, 1, 1, 20)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1, 1, 1, 20)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 20)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 21
=================================================================
Total params: 235,101
Trainable params: 235,101
Non-trainable params: 0
_________________________________________________________________
```

Figura 3.9. Detalle de las capas del estimador de malignidad

4

Descripción y procesamiento de los datos

En este capítulo se presenta una descripción detallada de los datos utilizados. Por otro lado, se explica el procesamiento de los mismos para pasar de imágenes médicas a información lista para entrar en los distintos modelos.

4.1. Descripción de los datos

Los datos utilizados para el entrenamiento, validación cruzada y elaboración de resultados vienen de la página web de un concurso llamado LUNA16 (*Lung Nodule Analysis 2016*), en el que los concursantes compiten en el diseño del mejor algoritmo para el diagnóstico automático de cáncer de pulmón. El concurso se divide en 2 competiciones independientes. Por un lado está el NDET, del inglés *nodule detection track*, en el que los concursantes han de desarrollar algoritmos de detección de nódulos cancerígenos directamente desde la imagen. El resultado final buscado es un algoritmo que predice la probabilidad de contener un nódulo cancerígeno de cada posición de la imagen. Por otro lado se organiza el FPRED, del inglés *false positive reduction track*, en el que el objetivo es desarrollar un algoritmo que, dada una lista de localizaciones candidatas a contener un nódulo, descarte todas aquellas que no contienen uno. Se trata, así pues, de una competición de reducción de falsos positivos [49].

En la página web de LUNA16 se ofrecen los siguientes datos a los concursantes:

- Base de datos con **888 imágenes** CT de pulmón en 3 dimensiones en formato Meta (MHD/RAW). Estás imágenes provienen a su vez de la base de datos pública de imágenes de pulmón LIDC/IDRI [50]. La dimensión de estas imágenes varía en la dimensión z (la vertical, de los pies a la cabeza del paciente) ya que el número de secciones en que se muestrea la caja torácica varía según el paciente. El tamaño de la imagen en z varía entre 100 y 700 aproximadamente. Las dimensiones x=512 e y=512 son iguales en todas las imágenes. En la Figura 4.1 se muestra una sección de una imagen 3D (unidades en vóxeles).

- Archivo **'candidates'** con 551065 candidatos a nódulo encontrados en las imágenes CT. Para cada candidato se da el identificador de la imagen en la que se encuentra, sus coordenadas absolutas en dicha imagen y la clase del candidato (1 si es nódulo, 0 si no).

- Archivo **'annotations'** con 1186 nódulos cancerígenos. Para cada nódulo se da, de nuevo, el identificador de la imagen en la que está, sus coordenadas absolutas en dicha imagen y su diámetro en milímetros.

- Archivo **'annotations enhanced'**. Este archivo es una prolongación del archivo 'annotations' que incluye además información sobre los nódulos; saber, las coordenadas del origen absoluto de coordeandas en la esa imagen, la malignidad (entre 0 y 5), el nivel de calcificación, la esfericidad, la lobulación, la espiculación y la textura entre otras.

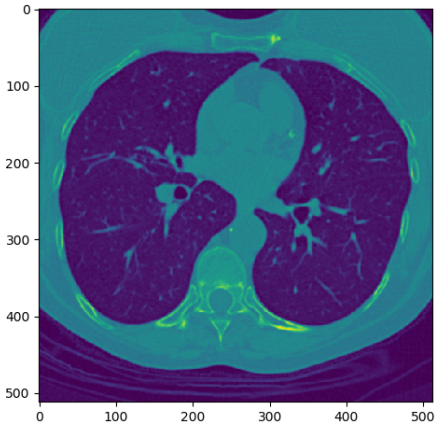

Figura 4.1. Sección perpendicular a z de una imagen 3D de muestra

La información sobre los nódulos encontrados se toma a partir de las anotaciones manuales de 4 radiólogos experimentados de la siguiente forma: cada radiólogo identifica lesiones en las imágenes aportadas y las clasifica según sean nódulos de más de 3 mm de diámetro, nódulos de menos de 3 mm de diámetro o no sean nódulos. Se consideran nódulos verdaderos (aquellos que se incluyen en 'annotations') aquellos que son identificados como nódulos por al menos 3 de los 4 radiólogos y que tienen más de 3 mm de diámetro.

La información sobre los candidatos se consigue de la siguiente forma. En primer lugar, se aplican 3 algoritmos de detección de candidatos sobre las imágenes y se marcan las posiciones sospechosas detectadas. Después, para evitar repetir candidatos encontrados por los algoritmos, se cuenta una sola vez por cada grupo de candidatos que estén a menos de 5 mm de distancia entre ellos. Este método proporciona una buena sensibilidad a los nódulos, detectándose 1120 de 1186 nódulos del archivo 'annotations'.

La información aportada en 'annotations enhanced', aunque no se encuentra en los datos de LUNA16, se puede conseguir en la base de datos LIDC/IDRI. Esta información viene también de la anotación manual de los 4 radiólogos.

Esta información permite conseguir tanto los datos de entrada de los modelos (imágenes 3D o recortes de las mismas) como las etiquetas de cada imagen, ya que se puede verificar fácilmente si un paciente tiene un tumor o no comprobando si su identificador aparece en la lista 'annotations'. A efectos de este proyecto, se utilizará la aparición en esta lista como como condición necesaria y suficiente de padecer cáncer de pulmón.

4.2. Procesamiento de los datos

Para el entrenamiento de los modelos, Keras requiere introducir las imágenes como objetos `ndarrays` de la librería numpy. En numpy, `ndarray` no es más que el nombre que se da a un tensor de n dimensiones de elementos del mismo tipo y tamaño. Todos los modelos utilizados requieren, por tanto, un procesamiento de las imágenes para transformarlas en los tensores 'ndarrays'. Esta es la fase primaria y común de procesamiento de los datos para todos los modelos.

Para leer el formato Meta '.mhd' de las imágenes se utiliza la librería de Python **SimpleITK** [51]. SimpleITK es un envoltorio que simplifica el uso de ITK (*Insight Segmentation and Registration Toolkit*), una librería para la transformación y gestión de imágenes médicas.

A continuación se explican algunas de las funciones utilizadas.

Cuando se da la posición de un elemento en una imagen 3D muchas veces se dan sus coordenadas absolutas, a saber, las que tendría en el espacio tridimensional correspondiente a la imagen respecto a algún centro de coordenadas dentro de la misma. Estas coordenadas se dan en metros, centímetros o milímetros. Sin embargo, cuando tenemos una imagen 3D en un tensor `ndarray`, nos interesa acceder a un punto de la imagen por su posición en dicho tensor. Los índices de un elemento en un tensor son sus coordenadas vóxel en la imagen. Para pasar las coordenadas absolutas de un punto a sus coordenadas vóxel en la imagen, basta con conocer la posición absoluta del origen de la imagen (esquina 0,0,0 del tensor) respecto al origen absoluto y el vector espaciado, que contiene la conversión entre distancias absolutas y distancias en píxeles en cada dirección. La fórmula utilizada es la siguiente:

$$\text{coordenadas vóxel} = \frac{\text{coordenadas absolutas} - \text{origen vóxel}}{\text{espaciado}} \tag{4.1}$$

Aquí la división por el vector espaciado es componente a componente, es decir, se divide cada coordenada por su correspondiente espaciado.

El código de la función de conversión de coordenadas, `worldToVoxelCoord` es el siguiente:

```
def worldToVoxelCoord(worldCoord, origin, spacing):
    stretchedVoxelCoord = worldCoord - origin
    voxelCoord = np.floor(stretchedVoxelCoord / spacing)
    return voxelCoord.astype(int)
```

La función `load_itk_image` lee una imagen y la devuelve como `ndarray`. Además esta función extrae también la posición del origen vóxel de la imagen respecto al origen absoluto y el vector de espaciado. El código es el siguiente:

```
def load_itk_image(filename):

    itkimage = sitk.ReadImage(filename)
    numpyImage = sitk.GetArrayFromImage(itkimage)
    numpyOrigin = np.array(list(reversed(itkimage.GetOrigin())))
    numpySpacing = np.array(list(reversed(itkimage.GetSpacing())))

    return numpyImage, numpyOrigin, numpySpacing
```

La función `normalizePlanes` se utiliza para escalar los valores de la imagen entre 0 y 1. Al tratarse de imágenes médicas de tomografía computarizada, el valor de cada píxel o vóxel representa el nivel de radiodensidad del tejido en cuestión en unidades de la escala Hounsfield [52]. El aire tiene un valor en esta escala de -1000 HU y el hueso de 400 HU. Para todas las imágenes se proyectan linealmente los valores de la escala Hounsfield al intervalo [0,1] siendo 0 la imagen de -1000 HU y 1 la imagen de 400 HU. Si existe algún valor por fuera de estos extremos se le asigna un valor igual al borde por el que sobresale. El código es el siguiente.

```
def normalizePlanes(npzarray):
    maxHU = 400.
    minHU = -1000.
    npzarray = (npzarray - minHU) / (maxHU - minHU)
    npzarray[npzarray > 1] = 1.
    npzarray[npzarray < 0] = 0.
    return npzarray
```

Por último, la función `resize_voxel` convierte una imagen 3D al tamaño requerido utilizando interpolación por splines de orden 1 (rectas) a través de la librería scipy. Esta función se utiliza sobre todo para que todas las imágenes tengan el mismo número de cortes en dirección z, ya que, como se explica en la descripción de los datos, el número de cortes varía según el paciente.

```
def resize_voxel(x, desired_shape):
    factors = np.array(x.shape).astype('float32') / np.array(desired_shape).astype('float32')
    output= ndimage.interpolation.zoom(x,1.0 / factors,order=1)
    assert output.shape == desired_shape, 'resize error'
    return output
```

5

Entrenamiento de los modelos

En este capítulo se explica de forma detallada el procedimiento seguido para el entrenamiento de los modelos con los que se experimenta en el proyecto.

5.1. Entrenamiento en aprendizaje automático

En el aprendizaje automático, la siguiente fase al diseño de un modelo es la de entrenamiento y validación para, por un lado, ajustar los parámetros del modelo a los datos y , por otro, para variar características no entrenables del mismo con el objetivo de buscar su combinación óptima. Algunos de estas características pueden ser el número de capas, el *dropout* de una capa, el grado de conectividad de una capa con la siguiente o su función de activación.

El proceso de entrenamiento consiste en buscar en el espacio de los parámetros entrenables del modelo la combinación óptima de los mismos. Esto es, el conjunto de parámetros que minimice una cierta función de coste que calcula el error de las predicciones del modelo con respecto a las etiquetas reales de los casos de entrenamiento. El entrenamiento de un modelo es un proceso de varias iteraciones llamadas épocas. Cada época constituye una barrida entera de los datos de entrenamiento, con varias actualizaciones de los parámetros durante la misma. El número de actualizaciones de los parámetros y del gradiente de la función de coste en el espacio de los parámetros durante el entrenamiento depende del llamado *batch size*, que es el número de ejemplos de entrenamiento procesados con cada actualización del estado de los parámetros. El proceso se repite durante un número prefijado de épocas. Alternativamente, se puede establecer que el entrenamiento finalice una vez el modelo ha dejado de aprender, es decir, que las siguientes iteraciones no minimizan más el error con los datos de entrenamiento. Se puede también, como se hace en este proyecto, detener el entrenamiento si una cierta métrica no mejora en el conjunto de validación durante un cierto número de épocas. En cualquier caso, se suele fijar siempre un número máximo de épocas.

Para los tres primeros modelos del proyecto, al tratarse de una clasificación, se utiliza como función de coste J la **entropía cruzada** [53]. Dicha función penaliza el modelo basándose en la probabilidad de pertenencia a cada clase c que asigna el modelo: $\hat{y_{i,c}}$. La predicción final del modelo, con la que se mide la precisión, es la clase que recibe la mayor probabilidad.

La entropía cruzada valora no sólo esta predicción, sino también la seguridad con la que el modelo clasifica la imagen. Esta función de coste crece rápidamente con la distancia entre la probabilidad predicha por el modelo y el valor real asignado a dicha clase $y_{i,c}$ (1 en caso de pertenencia, 0 en caso contrario). Para calcular la función de coste, se evalúa la entropía con las dos clases para cada ejemplo de entrenamiento y se hace una media, después de calcula la entropía media de todos los ejemplos:

$$J(\Theta) = -\frac{1}{2m}\sum_{i=1}^{m}\sum_{c=1}^{2}[(y_{i,c}\, log(\hat{y_{i,c}}) + (1 - y_{i,c})\, log(1 - \hat{y_{i,c}})] \tag{5.1}$$

Ver el apartado del estimador de malignidad de este capítulo para una explicación de su función de coste.

Debido al elevado coste computacional de entrenar redes convolucionales (más aun en tres dimensiones), todos los entrenamientos se realizan en una máquina virtual de Google Compute Engine [54]. En el Sección A.1 se puede encontrar una explicación de como conectarse a una máquina virtual de Google.

A continuación se explica el procedimiento de entrenamiento de los cuatro modelos, así como la selección de los datos de entrenamiento y validación.

5.2. Modelo global

Las imágenes de las que se dispone tienen distinto tamaño en dirección z, por tanto, es necesario transformar las imágenes a un tamaño común antes de pasarlas por el modelo. Esto se consigue con la función `resize_voxel` explicada en la sección de datos. Debido al elevado coste en computación y memoria de trabajar con las imágenes completas, se opta por reducir el tamaño de las imágenes a 50 x 200 x 200 vóxeles. El tamaño original es de Z x 512 x 512 vóxeles donde Z varía entre 100 y 700. La repartición de las imágenes en casos positivos y negativos y en entrenamiento y validación se muestra en la siguiente tabla.

	Negativos	**Positivos**
Entrenamiento	150	150
Validación	100	100

Tabla 5.1. Datos de entrenamiento y validación del modelo global

La función de coste a minimizar en este caso la entropía cruzada explicada al principio de este capítulo. Se ejecuta un entrenamiento de 50 épocas condicionado a terminar antes de tiempo si la función de coste computada en el conjunto de validación no mejora durante 10 épocas consecutivas respecto a una época, quedando como modelo final el último estado antes de dichas épocas.

Las características del modelo con las que se experimenta en este caso son las siguientes :

- El *dropout* de la última capa convolucional.
- El tamaño de los filtros de la primera capa convolucional.

5.3. Modelo local

Para el entrenamiento de la red neuronal del modelo local, correspondiente a la primera fase en el flujo de información de la estructura completa, se utilizan recortes de 30 x 30 x 30 vóxeles de las imágenes globales. Se generan "cubos" que contienen nódulos y cubos que no. La posición del nódulo dentro del cubo se elige de forma aleatoria. La posición de la que se recortan los cubos vacíos en los pacientes sanos es también aleatoria. Debido a esta componente aleatoria en la generación de los datos, los entrenamientos de cada versión del modelo no se hacen con las mismas imágenes exactamente. Sin embargo, el número de imágenes de entrenamiento y validación si permanece prácticamente constante. En la siguiente tabla pueden encontrarse la distribución de imágenes en entrenamiento y validación y en positivos y negativos.

	Negativos	**Positivos**
Entrenamiento	1326	1343
Validación	898	881

Tabla 5.2. Datos de entrenamiento y validación del modelo local

Para entrenar este modelo se toma como función de coste la entropía cruzada explicada al principio de este capítulo. Se ejecuta un entrenamiento de 50 épocas condicionado a terminar antes de tiempo si la función de coste evaluada en el conjunto de validación no mejora durante 10 épocas consecutivas el valor obtenido en la anterior a esas 10, quedando como modelo final el último estado antes de dichas épocas (la que no ha sido superada en 10 épocas).

En este caso, los parámetros que se hacen variar en los experimentos son, como en modelo global, el **tamaño de los filtros de la primera capa convolucional** y **el *dropout* del clasificador** (en la penúltima capa de la red).

Por otro lado, se realizan pruebas para calibrar los valores de los parámetros d_{max} y N del algoritmo `cluster_check`. Si bien es cierto que estas pruebas no son un entrenamiento de aprendizaje automático sino una evaluación directa de combinaciones de parámetros, se incluyen en esta sección la siguiente información sobre estas pruebas por mantener un cierto orden en el documento.

Cada prueba se realiza con 10 imágenes, de las cuales 5 son casos positivos. Todas las pruebas se realizan con la versión de la red `nodule_detector` que mejores resultados da en validación. El número de imágenes de cada prueba se reduce a 10 por una cuestión de tiempo de ejecución y plazos, dado el elevado tiempo de procesamiento de una imagen a través del algoritmo `cluster_check` (unos 10 minutos).

5.4. Modelo de *transfer learning*

Los conjuntos de entrenamiento y de validación contienen imágenes bidimensionales de secciones de distintos pacientes. Las dimensiones de las imágenes son 512 x 512 píxeles. Se trata de imágenes de un solo canal, donde el valor de cada píxel es el valor escalado de intensidad en la escala Hounsfield explicada en la sección de datos de este documento. En la Figura 5.1 se muestra un ejemplar de dichas imágenes.

Figura 5.1. Ejemplo de una imagen 2D para el modelo de *transfer learning*

Para el entrenamiento de este modelo, se utiliza además una técnica llamada *data augmentation* que consiste en aplicar transformaciones aleatorias sobre algunas de las imágenes para ampliar el conjunto de datos. La decisión sobre las imágenes sobre las que se aplica una alteración es aleatoria y distinta en cada época. Las transformaciones también son aleatorias y distintas en cada época. Las transformaciones aplicadas son rotaciones, traslaciones horizontales y verticales, deformaciones de cizallamiento, "zoom" y giro horizontal. Se rellenan los huecos resultantes en el cuadro de la imagen original con el valor del píxel en la frontera de la imagen alterada. Se utiliza el siguiente código para crear el generador de datos:

```
training_datagen = ImageDataGenerator(
            preprocessing_function=self._preprocessing_function,
            rotation_range=30,
            width_shift_range=0.2,
            height_shift_range=0.2,
            shear_range=0.2,
            zoom_range=0.2,
            horizontal_flip=True,
            fill_mode='nearest'
        )
```

En la siguiente tabla se muestra la repartición de los datos en positivos, negativos y en entrenamiento y validación.

	Negativos	**Positivos**
Entrenamiento	496	439
Validación	200	173

Tabla 5.3. Datos de entrenamiento y validación del modelo *transfer learning*

Se toma como función de coste a minimizar la entropía cruzada explicada al principio de este capítulo. De nuevo, se ejecuta un entrenamiento de 50 épocas condicionado a terminar antes

de tiempo si la función de coste computada en el conjunto de validación no mejora durante 10 épocas consecutivas (mismo principio que en los modelos anteriores).

Las características del modelo que se varían en este caso son las siguientes:

- El **modelo de base**. Se prueban tres de los modelos disponibles en Keras Applications [48]: Xception, ResNet50 y VGG19. Se eligen estos modelos para probar distintos niveles de profundidad y número de parámetros. En el Anexo 1 se puede encontrar una descripción detallada de la arquitectura de los tres modelos.
- El ***dropout*** de la última capa antes de la capa de salida del clasificador.
- El **número de capas entrenables** o "descongeladas" del modelo de base. Dichas capas se toman siempre empezando desde el final del modelo de base, es decir, cerca del clasificador y lejos de la entrada.

5.5. Estimador de malignidad

A diferencia de los modelos precedentes, la función de coste para este modelo es simplemente el error cuadrático medio (MSE del inglés *mean squared error*), que mide la media de las distancias al cuadrado entre las predicciones y los valores reales de malignidad:

$$J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{5.2}$$

Se utilizan las siglas RMSE (del inglés *root mean squared error*) para referirse a la raíz cuadrada del MSE.

Para este modelo, los conjuntos de entrenamiento y validación contienen imágenes tridimensionales de dimensión 30 x 30 x 30 recortadas de las imágenes globales. En este caso se recortan solo secciones que contienen algún nódulo. Las imágenes se toman de forma que el nódulo tenga una posición aleatoria dentro de la misma, para que el modelo sea capaz de predecir su malignidad independientemente de su posición dentro del recorte. Para construir estas imágenes se utiliza el siguiente código, que genera n imágenes aleatorias entorno a un nódulo:

```
        for i in range(n):

            x_center_raw = float(nod[1])
            y_center_raw = float(nod[2])
            z_center_raw = float(nod[3])
            (z_nod_vox, y_nod_vox, x_nod_vox) =
                worldToVoxelCoord(np.asarray([z_center_raw, y_center_raw,
                x_center_raw]), origin, spacing)

            z_start = np.random.randint(low=np.round(z_nod_vox - 0.9 * CUBE_SIZE),
                high=np.round(z_nod_vox - 0.1 * CUBE_SIZE))
            y_start = np.random.randint(low=np.round(y_nod_vox - 0.9 * CUBE_SIZE),
                high=np.round(y_nod_vox - 0.1 * CUBE_SIZE))
            x_start = np.random.randint(low=np.round(x_nod_vox - 0.9 * CUBE_SIZE),
                high=np.round(x_nod_vox - 0.1 * CUBE_SIZE))
```

```
            #z_in_cube = z_nod_vox - z_start

            z_end = z_start + CUBE_SIZE
            y_end = y_start + CUBE_SIZE
            x_end = x_start + CUBE_SIZE

            cube = img[z_start:z_end, y_start:y_end, x_start:x_end]
            cube = normalizePlanes(cube)

            if cube.shape==(CUBE_SIZE, CUBE_SIZE, CUBE_SIZE):
                cubes.append(cube)
                malignancies.append(float(nod[-1]))
```

El tamaño del conjunto de entrenamiento es de 669 imágenes. El conjunto de validación contiene 446 imágenes. La malignidad media en todo el conjunto de datos (entrenamiento y validación) es aproximadamente 3 (2.87). Como en los demás modelos, se ejecuta un entrenamiento condicionado a terminar antes de tiempo. En este caso se detiene el entrenamiento si el error cuadrático medio en el conjunto de validación no mejora durante 10 épocas consecutivas. En ese caso, queda como modelo resultante el último estado antes de las diez épocas de no mejora. El número máximo de épocas es 50.

Las características del modelo con las que se experimenta son el **tamaño de los filtros de la primera capa** y el ***dropout* de la última capa** antes de la salida.

6

Análisis de resultados

Aquí se presentan los resultados de los experimentos con los cuatro modelos del proyecto. Se concluye acerca de los mismos en relación al objetivo del proyecto.

6.1. Métricas de evaluación de resultados

Para los tres primeros modelos, al tratarse de modelos de clasificación, las métricas utilizadas son las siguientes:

- **Entropía cruzada en el conjunto de validación**: Se computa como se explica en la sección de entrenamiento de este documento. Mide la seguridad media con la que la red neuronal ha clasificado el conjunto de validación.
- En las tareas de clasificación con dos clases se definen los términos **verdaderos positivos (VP)**, **falsos positivos (FP)**, **verdaderos negativos (VN)**, **falsos negativos (FN)** para clasificar cada muestra según su etiqueta verdadera y la asignada por el clasificador. En la siguiente tabla se muestran las cuatro combinaciones posibles:

Etiqueta real \Predicción	**Negativo**	**Positivo**
Negativo	Verdadero negativo	Falso positivo
Positivo	Falso negativo	Verdadero positivo

Tabla 6.1. Nombres de los distintos casos en una clasificación de dos clases

  Para aportar la mayor información posible, en los resultados de cada modelo se muestran estos datos. Se llama **matriz de confusión** a la aportación de estos datos en una tabla como la anterior. Este concepto es general para una clasificación de cualquier número de clases.

- **Precisión (P)**: Fracción de casos bien clasificados del conjunto de validación. Se calcula de la siguiente manera:

$$P = \frac{VP + VN}{VP + FP + VN + FN} \tag{6.1}$$

- **Sensibilidad (S)**: Es la fracción de casos positivos detectados por el modelo:

$$S = \frac{VP}{VP + FN} \tag{6.2}$$

- **Especificidad (E)**: Es la fracción de casos negativos correctamente identificados como tal por el modelo:

$$E = \frac{VN}{VN + FP} \tag{6.3}$$

Para el cuarto modelo se utiliza la misma métrica que la utilizada en la función de coste del modelo: el error cuadrático medio (MSE). Se muestra también el RMSE. En la sección de entrenamiento de de este documento se ofrece una explicación precisa de estas métricas.

6.2. Modelo global

A continuación se muestran los resultados en validación de los experimentos del modelo 1.

		Resultados en validación						
Tamaño de los primeros filtros	**Dropout del clasificador**	**Entropía cruzada**	**VP**	**FP**	**VN**	**FN**	**Sensibilidad**	**Especificidad**
(4, 10, 10)	0.1	1.75	16	55	45	84	0.16	0.45
(4, 10, 10)	0.2	1.61	21	57	43	79	0.21	0.43
(4, 10, 10)	0.3	1.21	18	58	42	82	0.18	0.42
(4, 14, 14)	0.1	1.2	62	83	17	38	0.62	0.17
(4, 14, 14)	0.2	1.5	28	67	33	72	0.28	0.33
(4, 14, 14)	0.3	1.41	23	65	35	77	0.23	0.35

Tabla 6.2. Resultados del modelo global

Los resultados de las pruebas son desastrosos. Resulta muy difícil establecer las razones por las que fracasa este primer modelo. Algunas causas pueden ser un sobreajuste del modelo en el entrenamiento, un tamaño demasiado pequeño del conjunto de datos o una falta de profundidad de la red neuronal (6 capas).

Dado que la evaluación de las métricas en el conjunto de entrenamiento es significativamente mejor que en el de validación (la precisión está entorno a 0.76 y la entropía cruzada a 0.4 aunque no se muestre) uno podría estar tentado a pensar que el modelo está sobreajustándose a los datos de entrenamiento. Sin embargo, se puede apreciar que un incremento importante del nivel de *dropout* del clasificador (una buena medida contra el sobreajuste) no varía prácticamente la precisión del modelo. Esto puede ser un indicativo de que el problema es otro. Además, en aprendizaje automático es normal que un modelo funcione mejor en el conjunto de entrenamiento ya que se trata de conjunto de imágenes que ya ha visto. Resulta interesante que el incremento de *dropout* si parece disminuir la entropía cruzada en validación.

Por otro lado, el aumento del tamaño del filtro en x e y parece tener un ligero efecto positivo en la precisión del modelo (el tamaño del filtro en z se mantiene constante e igual a 4).

6.3. Modelo local

A continuación se muestran los resultados en validación de la red neuronal del modelo local.

		Resultados en validación							
Tamaño de los primeros filtros	**Dropout del clasificador**	**Entropía cruzada**	**Precisión**	**VP**	**FP**	**VN**	**FN**	**Sensibilidad**	**Especificidad**
(4, 4, 4)	0.1	0.55	0.78	375	102	86	86	0.813	0.763
	0.2	0.66	0.76	344	140	328	78	0.815	0.701
	0.3	0.68	0.79	402	148	302	38	0.914	0.671
(8, 8, 8)	0.1	0.57	0.76	379	145	300	66	0.852	0.674
	0.2	0.55	0.75	298	71	369	150	0.665	0.839
	0.3	0.54	0.77	331	84	353	122	0.731	0.808
(12, 12, 12)	0.1	0.59	0.74	311	106	353	121	0.720	0.769
	0.2	0.67	0.57	426	362	79	23	0.949	0.179
	0.3	0.53	0.74	393	173	263	61	0.866	0.603

Tabla 6.3. Resultados del detector de nódulos

Los resultados de la versión de tamaño de filtro 12 píxeles de lado y *dropout* 0.2 son inesperados en comparación con los de las versiones de mismo tamaño de filtro. Viendo su sensibilidad elevada y especificidad baja, esta versión claramente da un exceso de positivos por alguna razón desconocida.

El tamaño del filtro de entrada parece ser más o menos irrelevante dentro de los rangos con los que se experimenta. Los resultados son ambiguos de interpretar. Sin embargo, está claro que la precisión conseguida por el detector de nódulos es mejor que la de los otros dos modelos de clasificación del proyecto (el modelo global y el modelo de *transfer learning*). La hipótesis más probable de por qué esto es así es que los otros dos modelos tienen que detectar un nódulo en una imagen mucho más grande mientras que este modelo lo detecta en un volumen de aproximadamente 10 veces el volumen del nódulo.

Se toma como mejor versión del detector de nódulos la correspondiente a la tercera prueba, con filtros de entrada de lado 4 y *dropout* 0.3, que da un resultado de 79 % de precision y 91 % de sensibilidad. Es con esta versión con la que se realizan las pruebas del algoritmo de *clustering*.

En la siguiente tabla se encuentran los resultados de las diferentes combinaciones de parámetros para el algoritmo de *clustering* del modelo. Todas las pruebas se realizan tomando como detector de nódulos el modelo con tamaño 4 píxeles de lado de los primeros filtros y *dropout* de 0.3, considerado el mejor a la vista de los resultados anteriores.

		Resultados			
d_{max}	N	**VP**	**FP**	**VN**	**FN**
2	170	5	5	0	0
1.5	170	5	5	0	0
1.5	197	5	5	0	0
1.5	215	5	5	0	0

Tabla 6.4. Resultados del algoritmo de clustering

Los resultados muestran un claro exceso de falsos positivos (de hecho no hay diagnósticos negativos). No se genera ningún negativo ni en el caso mas exigente para dar un diagnóstico positivo (con d_{max} igual a 1.5 y N igual a 215). Analizando los *clusters* de predicciones positivas formados se observa que únicamente existe uno, que contiene todas los positivos del tensor de predicciones. En el ejemplo que se muestra en Figura 6.1 se puede apreciar que existe un único *cluster* de 24371 elementos que agrupa todos los positivos. El tensor de predicciones contiene 94863 elementos.

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]]
Number of points in clusters:  [24371]
```

Figura 6.1. Ejemplo de *clusters* obtenidos por el algoritmo

Si se considera el caso con d_{max} igual a 1.5, el algoritmo únicamente puede agrupar positivos contiguos. La única explicación posible de que se forme un único *cluster* es que exista siempre un camino de positivos entre dos positivos cualesquiera del tensor de predicciones, es decir, que **el conjunto de predicciones positivas es conexo**. Esto se puede explicar a través de la moderada especificidad del detector de nódulos utilizado (0.67). La probabilidad de dar un falso positivo del detector es 0.33. Es posible que aparezcan falsos positivos que terminen conectando el conjunto de positivos en el tensor de predicciones. Además, las zonas más sospechosas del pulmón se encuentran cerca del centro y cerca de las paredes del pulmón y por lo tanto están conectadas. Se podría decir que la zona del contorno de los dos pulmones es un conjunto conexo. Parece razonable, por tanto, que se pueda crear un camino de positivos que recorra todas estas zonas sospechosas. En realidad, basta con que exista un fino "hilo" de positivos entre dos zonas de predicciones positivas para que éstas queden agrupadas. En la sección de futuro desarollo se propone una mejora del algoritmo.

6.4. Modelo *transfer learning*

A continuación se muestran los resultados en validación de las distintas variaciones del modelo 3.

			Resultados en validación							
Modelo base	**Número de capas descongeladas**	**Dropout**	**Entropía cruzada**	**Precisión**	**VP**	**FP**	**VN**	**FN**	**Sensibilidad**	**Especificidad**
Xception	0	0.1	4.1	0.61	60	31	169	113	0.347	0.845
		0.2	0.65	0.59	48	29	171	125	0.277	0.855
	1	0.1	0.66	0.53	1	3	197	172	0.006	0.985
		0.2	0.62	0.56	34	24	176	139	0.196	0.88
	2	0.1	0.64	0.59	54	33	167	119	0.312	0.835
		0.2	0.61	0.55	20	13	187	153	0.115	0.935
ResNet50	0	0.1	0.64	0.55	13	7	193	160	0.075	0.965
		0.2	0.61	0.54	18	10	190	155	0.104	0.95
	1	0.1	0.61	0.54	6	2	198	167	0.030	0.99
		0.2	0.59	0.62	77	47	153	96	0.445	0.765
	2	0.1	0.59	0.62	79	47	153	94	0.456	0.765
		0.2	0.6	0.61	52	23	177	121	0.3	0.885
VGG19	0	0.1	5.05	0.66	75	29	171	98	0.43	0.855
		0.2	5.21	0.61	50	23	177	123	0.289	0.885
	1	0.1	4.82	0.65	107	65	135	66	0.618	0.675
		0.2	5.08	0.65	71	27	173	102	0.410	0.865
	2	0.1	5.33	0.64	70	32	168	103	0.404	0.84
		0.2	4.82	0.6	40	15	185	133	0.231	0.925

Tabla 6.5. Resultados del modelo *transfer learning*

La entropía cruzada media es bastante mayor en el modelo VGG19. Sin embargo, su precisión media es algo mejor que las otras dos, su sensibilidad media es mucho mejor (casi el doble) y su especificidad media es prácticamente igual (algo menor). Teniendo en cuenta que las métricas más relevantes a la hora de una aplicación real en medicina son estas tres últimas, se puede decir que el modelo VGG19 es el mejor adaptado para este tipo de clasificación.

Esto probablemente se deba a su mayor número de parámetros (aproximadamente 6 veces más que los otros dos). VGG19 tiene además muchas menos capas que los otros modelos (5 veces menos que Xception y 7 veces menos que ResNet50), lo cual puede indicar que un menor número de capas pero de mayor extensión sea la estructura óptima para el problema en cuestión. De nuevo, es díficil establecer una relación causal entre los párametros de las redes y los resultados. Lo que parece probable es que aumentar el número de parámetros mejora el modelo. Se puede encontrar una tabla comparativa de las características de los modelos en la sección de modelos de este documento.

6.5. Estimador de malignidad

A continuación se muestran los resultados en validación del estimador de malignidad.

		Resultados en validación	
Tamaño del filtro de la primera capa	**Dropout de la ultima capa**	**MSE**	**RMSE**
8	0.1	1.22	1.1
	0.2	1.1	1.05
	0.3	1.08	1.04
10	0.1	0.93	0.96
	0.2	0.91	0.95
	0.3	0.89	0.94
12	0.1	0.89	0.94
	0.2	1.37	1.17
	0.3	0.92	0.96

Tabla 6.6. Resultados del estimador de malignidad

Los resultados son algo ambiguos. Parece que un amumento del *dropout* mejora los resultados (de media). Dado que el error en el conjunto de entrenamiento es significativamente menor que en el conjunto de validación es posible que el modelo esté sobreajustándose al conjunto de entrenamiento, a pesar del *dropout*. El mejor tamaño de los filtros de entrada, de media, es el de lado 10 píxeles que reduce el MSE en un 19 % respecto al de 8 píxeles y un 14 % respecto al de 12 píxeles. Los resultados del modelo de 12 píxeles de lado de primer filtro y *dropout* de 0.2 son inesperados con respecto a los otros dos de mismo tamaño de filtro.

En general, los resultados no son malos teniendo en cuenta que lo que intenta predecir el modelo es una media de la valoración subjetiva de malignidad de 4 radiólogos. Los radiólogos discrepan, a veces, incluso en su decisión de si una masa dada en un pulmón es un tumor.

7
Conclusiones

En este último capítulo se presentan las conclusiones del proyecto.

En vista de los resultados de las pruebas para los distintos modelos se sacan las siguientes conclusiones.

Vistos los malos resultados en todas las métricas posibles del **modelo global**, parece evidente que se trata de un problema demasiado complejo y sutil como para que una red neuronal convolucional sea capaz de abordarlo con buena precisión con la profundidad dada (6 capas y aproximadamente 270000 parámetros) y el conjunto de imágenes disponible.

El **modelo local** pone en evidencia que un análisis local de la imagen facilita enormemente el diagnóstico. A la hora de dar un diagnóstico global, el algoritmo de *clustering* utilizado no da los resultados esperados. Esto probablemente se deba a que el conjunto de predicciones positivas sea conexo al ser las zonas más sospechosas los contornos de los pulmones, que están conectados entre sí.

En cuanto al **modelo *transfer learning***, la conclusión es que la explicación más probable de la mejora con respecto al modelo global es el gran aumento (en varios cientos de órdenes de magnitud) del número de parámetros. Esto también es cierto si se comparan los resultados de los tres modelos de base: Xception, ResNet50 y VGG19. Viendo los resultados de los 3 modelos, VGG19 es en resultados medios el mejor de los tres (sobre todo en sensibilidad), de nuevo, probablemente por su mayor número de parámetros. El hecho de que los resultados de Xception y ResNet50 sean muy similares teniendo ambos un número similar de parámetros también parece aportar evidencia a esta hipótesis.

Los resultados del **estimador de malignidad** parecen indicar que es muy difícil estimar mediante aprendizaje automático una medición médica subjetiva desde una imagen. Esto parece razonable ya que ni siquiera los propios radiólogos que ponen las etiquetas de malignidad están siempre de acuerdo. En la sección de datos se explica que los radiólogos discrepan, a veces, incluso en la identificación de si una masa en el pulmón es un tumor o no. Las pruebas de este modelo en realidad, miden si la red neuronal es capaz de imitar un criterio medio de 4 radiólogos.

8

Futuro desarrollo

En este capítulo se sugieren posibles vías de futuro desarrollo del proyecto a raíz de los resultados y la experiencia del proyecto.

8.1. Combinación del modelo local y el de *transfer learning*

Viendo los resultados del detector de nódulos parece evidente, como se ha explicado en la sección de conclusiones en el capítulo de resultados, que la mejor forma de atacar el problema es a través de analizar la imagen desde cerca, por zonas. Además, la comparación de los resultados del modelo global y el modelo de *transfer learning* e incluso de las distintas versiones del modelo de *transfer learning* parece indicar que se trata de un problema en el que un aumento del número de parámetros de la red ayuda a conseguir una mejor clasificación.

La combinación de las fortalezas de estos dos modelos es una vía de desarrollo futuro evidente y con mucho potencial. La mejor forma de llevar a cabo esta implementación sería hacer una red neuronal cuya entrada fuera un recorte bidimensional ya que la gran mayoría de los modelos disponibles para *transfer learning* son en 2D. Sería interesante probar un modelo de base con un elevado número de parámetros, como es el caso de VGG19, alterando más o menos capas del mismo con *fine tuning*.

8.2. Uso de características de los nódulos para el diagnóstico global

Como se explica en la sección de datos de este documento, el archivo **'annotations enhanced'** contiene para cada nódulo, además del identificador de la imagen en la que se encuentra y sus coordenadas en dicha imagen, una serie de características del nódulo. Entre otras aparecen el diámetro, la malignidad, la espiculación, el nivel de calcificación, la esfericidad, la lobulación y la textura.

En este proyecto la única característica de los tumores que se ha utilizado es la malignidad y además no para dar un diagnóstico global. En [55], Hammack y de Wit, segundos clasificados del concurso de ciencia de datos Data Science Bowl 2017 organizado por Kaggle [33], utilizan las demás características nodulares en sus modelos y miden la aportación al diagnóstico global de dichas características en los nódulos encontrados. Es evidente que utilizar las demás características contribuye a la mejora de la precisión del diagnóstico. En la siguiente figura se muestra la importancia de cada característica utilizada a la hora de hacer el diagnóstico global (se hacen agrupaciones de las mismas para los nódulos contenidos en el paciente: la media, el máximo y la desviación típica entre otras). Dicha importancia se mide como el incremento en la entropía cruzada de un modelo lineal en un conjunto de imágenes cuando se hacen permutaciones aleatorias de esa característica en las muestras. De esta manera, se perturba la información que aporta dicha característica y se mide como empeora el modelo. Se añaden también características aleatorias para obtener una idea del nivel de relevancia de las características:

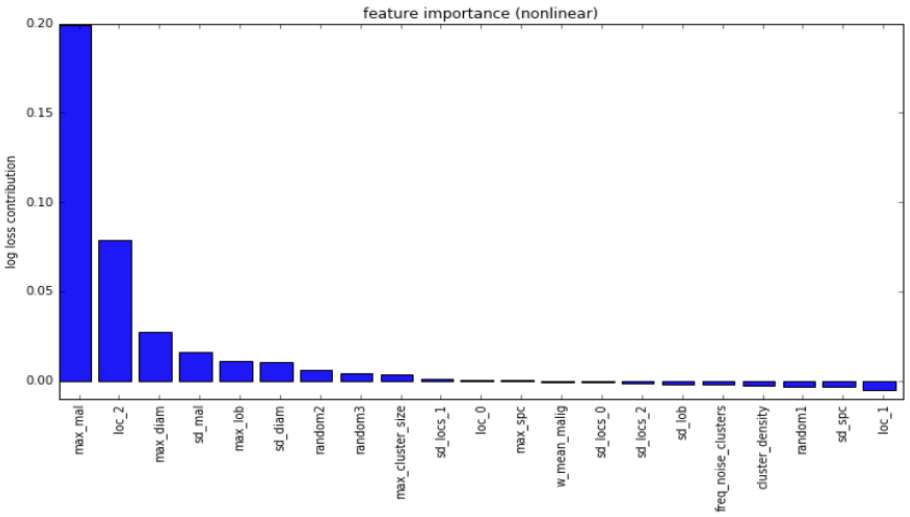

Figura 8.1. Gráfica de importancia de características [55]

Encuentran que, como sería de esperar, la malignidad máxima de los nódulos del paciente es la más relevante. La posición del nódulo más maligno en z es también una de las características más importantes, siendo más peligroso un nódulo cuanto mas cerca de la cabeza se encuentre.

Utilizar otras características nodulares puede ser una buena forma de mejorar el estimador de malignidad. Además, se podrían combinar el detector de nódulos y un estimador de malignidad mejorado para hacer un diagnóstico global como hacen los autores mencionados. Se podría también diseñar un estimador de otras características nodulares como la lobulación o la espiculación. Las posibilidades de desarollo en este aspecto son numerosas.

8.3. Desarrollo del algoritmo de clustering del modelo local

A la hora de generar el tensor de predicciones desplazando el detector de nódulos por la imagen, podría ser interesante utilizar la probabilidad de contener un nódulo asignada por el modelo en lugar de una etiqueta 1 o 0 como en este proyecto. Esto generaría un ”mapa de

calor" de la imagen y permitiría ver las zonas de mayor probabilidad de contener un tumor. Esto añadiría más información al tensor de predicciones ya que se utilizaría un continuo de valores entre 0 y 1 y no simplemente la predicción final. Al utilizar la predicción final, se pierde la información asociada al nivel de confianza del detector de nódulos en su decisión. A este "mapa de calor" se podría aplicar un algoritmo de segmentación que extraiga las zonas dentro de un cierto umbral de malignidad. Después se podría utilizar la cantidad de "masa sospechosa" en el paciente para su diagnóstico final. En [55], de Wit y Hammack utilizan esta cantidad de "masa sospechosa" para su diagnóstico con buenos resultados.

Para mejorar la capacidad del algoritmo de *clustering* de agrupar zonas con alta densidad de predicciones positivas, las cuales deberían ser indicativas de la presencia de un nódulo, se podría investigar el uso de un filtro gaussiano [56] 3D antes de la agrupación de los unos del tensor de predicciones. Con esto se conseguiría difuminarlo y quitarle importancia a las predicciones positivas aisladas. De esta forma se conseguiría dar importancia a los bloques de unos y no a los unos aislados. Con esta técnica se podría evitar que un fino "hilo" de positivos conecte dos masas sospechosas pero sanas y por tanto provoque un diagnóstico global positivo (lo sucedido en las pruebas del proyecto). Al aplciar el filtro, los valores del tensor de predicciones difuminado ya no serían unos y ceros sino un continuo de valores en el intervalo [0,1], se podría aplicar la misma lógica que la del algoritmo de *clustering* del modelo local de este proyecto pero agrupando elementos dentro de un cierto umbral de cercanía al uno.

A

Máquina virtual del proyecto

Encuentre en este Anexo una guía a cómo conectarse a una máquina virtual de Google Compute Engine. Puede encontrar también una descripción de la máquina virtual utilizada en este proyecto.

A.1. Google Compute Engine. Máquina Virtual del proyecto

Google Compute Engine es una plataforma que forma parte de Google Cloud que ofrece máquinas virtuales para realizar tareas de programación de alto coste computacional. Google permite al usuario diseñar la máquina a su gusto según las necesidades de memoria y computacionales requeridas.

En la Figura A.1 y en la Figura A.2 se pueden encontar los detallles técnicos sobre la máquina utilizada en este proyecto.

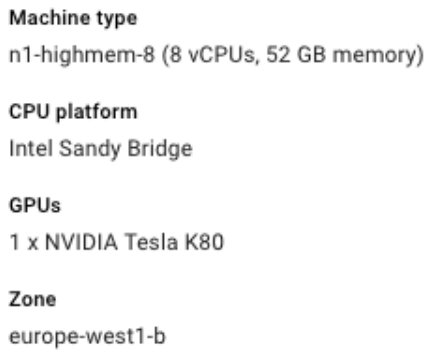

Figura A.1. Detalles técnicos de la máquina utilizada

Boot disk and local disks

Name	Size (GB)	Type	Encryption
instance	120	SSD persistent disk	Google managed

Figura A.2. Disco duro de la máquina virtual

A.2. Conectarse a la máquina virtual

En este proyecto la metodología para la ejecución de los códigos más pesados consiste en escribirlos en el ordenador local, conectarse por SSH a la máquina virtual y ejecutar los códigos ahí desde el terminal, sin necesidad de un entorno de desarrollo de Python. La máquina utilizada no tiene interfaz de usuario y por tanto solo puede controlarse a través del terminal. Para conectarse con la máquina desde el terminal es necesario descargar gcloud command-line tool [57], el paquete de comandos de Google Cloud para manejar máquinas remotas desde el terminal del ordenador local.

A continuación se explica como conectarse a la máquina para ejecutar un código.

1. Se enciende la máquina virtual desde el perfil de la plataforma Google Compute Engine. También se puede encender la máquina virtual desde el terminal con el siguiente comando:

```
gcloud compute instances start NOMBRE_DE_LA_INSTANCIA
```

El nombre de la instancia (o máquina virtual) aparece en el perfil de la plataforma Google Compute Engine.

2. Se conecta el ordenador local a la máquina virtual a través de SSH a través del siguiente comando en el terminal:

```
gcloud compute ssh NOMBRE_DE_LA_INSTANCIA
```

Una vez ejecutado este comando, la ventana del terminal pasa a ser el terminal de la máquina virtual. Desde esta ventana se controla el movimiento interno de archivos dentro de la máquina.

3. Se envía el código desde el ordenador local a la máquina virtual. Para ello se abre una nueva ventana o pestaña del terminal y se escribe el siguiente comando:

```
gcloud compute scp RUTA_LOCAL_AL_CODIGO NOMBRE_DE_LA_INSTANCIA:~/
```

En caso de querer enviar varios archivos de una carpeta se utiliza el siguiente comando en el terminal:

```
gcloud compute scp --recurse RUTA_LOCAL_A_LA_CARPETA
    NOMBRE_DE_LA_INSTANCIA:~/
```

4. Para ejecutar el código desde la máquina, en el caso de ser un código en Python, se utiliza el siguiente comando. Se han de instalar Python y las librerías necesarias de antemano.

```
python NOMBRE_DEL_ARCHIVO.py
```

Si lo que se quiere es enviar un archivo desde la máquina virtual hasta el ordenador local se ejecuta un comando similar:

```
gcloud compute scp NOMBRE_DE_LA_INSTANCIA:DIRECTORIO_REMOTO
    DIRECTORIO_LOCAL
```

O, en caso de ser varios archivos:

```
gcloud compute scp --recurse NOMBRE_DE_LA_INSTANCIA:DIRECTORIO_REMOTO
    DIRECTORIO_LOCAL
```

B

Presupuesto del proyecto

Encuentre en este documento una estimación del coste económico del proyecto.

B.1. Mediciones

En esta sección se recogen los equipos utilizados y las horas de trabajo empleados en el proyecto.

B.1.1. Equipos

Elemento	Cantidad	Horas de proyecto	Horas de uso al año
Ordenador	1	300	1000
Máquina virtual de Google Compute Engine con GPU	1	100	-

Las horas de uso anual de la máquina virtual son irrelevantes para el cálculo del coste del elemento. Se calculará el coste de la máquina virtual como un coste variable por el tiempo de uso en el proyecto, a diferencia del ordenador, cuyo coste se calcula a través de su amortización.

B.1.2. Software

Programa	Cantidad	Horas de proyecto	Horas de uso al año
PyCharm CE	1	150	200
Python	2	250	330
Keras	2	250	330
Tensorflow	2	250	330
Scipy	2	250	330
Texmaker	1	150	200

B.1.3. Mano de obra

Actividad	Horas
Programación	150
Documentación	100
Concepción de modelos	25
Análisis de resultados	25

B.2. Precios unitarios

Encuentre en esta sección los precios unitarios de los equipos y la mano de obra empleados en el proyecto.

B.2.1. Equipos

Elemento	Precio (€/ud)
Ordenador	1300
Máquina virtual de Google Compute Engine con GPU	1 / hora

B.2.2. Software

Todo el software utilizado en el proyecto es software libre. El coste del software del proyecto es por tanto nulo.

B.2.3. Mano de obra

Actividad	Precio (€/hora)
Programación	20
Documentación	40
Diseño de modelos	40
Análisis de resultados	40

B.3. Sumas parciales

En está sección se computan las contribuciones al coste total de los equipos y mano de obra empleados en el proyecto.

B.3.1. Equipos

Se cuenta por separado el coste del ordenador, correspondiente a la amortización del mismo, y el coste de la máquina virtual, que es variable.

Elemento	Unidades	Horas de proyecto	Horas anuales	Precio (€/ud)	Amortización anual	Coste (€)
Ordenador	1	300	1000	1300	25 %	100

El coste de la máquina virtual es variable en función del tiempo de uso. El precio por hora se calcula como $ 0.47 por la máquina más $ 0.45 por la GPU integrada, lo cual hace aproximadamente 0,80 €por hora.

Elemento	Uds	Precio (€/hora)	Horas de proyecto	Coste (€)
Máquina virtual de Google Compute Engine con GPU	1	0,80	100	80

B.3.2. Mano de obra

Actividad	Horas	Precio (€/hora)	Coste (€)
Programación	150	20	3000
Documentación	100	40	4000
Diseño de modelos	25	40	1000
Análisis de resultados	25	40	1000
Total			**9000**

B.4. Coste total

Sumando los costes de las herramientas y la mano de obra se obtiene el siguiente presupuesto total:

Concepto	Coste (€)
Ordenador	100
Máquina virtual de Google Compute Engine con GPU	80
Mano de obra	9000
Total	**9180**

Bibliografía

[1] Y. Matsuki, K. Nakamura, H. Watanabe, T. Aoki, H. Nakata, S. Katsuragawa, and K. Doi, "Usefulness of an artificial neural network for differentiating benign from malignant pulmonary nodules on high-resolution ct: evaluation with receiver operating characteristic analysis," *American Journal of Roentgenology*, vol. 178, no. 3, pp. 657–663, 2002.

[2] A. M. Rossetto and W. Zhou, "Deep learning for categorization of lung cancer CT images," in *IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pp. 272–273, July 2017.

[3] J. Dehmeshki, J. Chen, M. V. Casique, and M. Karakoy, "Classification of lung data by sampling and support vector machine," in *Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE*, vol. 2, pp. 3194–3197, IEEE, 2004.

[4] A. Karahaliou, S. Skiadopoulos, I. Boniatis, P. Sakellaropoulos, E. Likaki, G. Panayiotakis, and L. Costaridou, "Texture analysis of tissue surrounding microcalcifications on mammograms for breast cancer diagnosis," *The British journal of radiology*, vol. 80, no. 956, pp. 648–656, 2007.

[5] OrganizaciÃ$^3 nMundialdelaSalud$, "$Las10principalescausasdedefunci^3n$," 2017.

[6] The National Lung Screening Trial Research Team, "Reduced lung-cancer mortality with low-dose computed tomographic screening," *New England Journal of Medicine*, vol. 365, no. 5, pp. 395–409, 2011.

[7] Wikipedia, "Diagnósticos asistidos por ordenador," 2017.

[8] Wikipedia, "Aprendizaje automático," 2017.

[9] Wikipedia, "Aprendizaje no supervisado," 2017.

[10] H. Daumé III, *A Course in Machine Learning*. 2012.

[11] Wikipedia, "Supervised learning," 2018.

[12] Machinelearningtutorial.net, "Test set vs training ste vs validation set - what's the deal?," 2017.

[13] Wikipedia, "Máquinas de vectores de soporte," 2017.

[14] Towards Data Science, "Activation functions: Neural networks," 2017.

[15] Wikipedia, "Support vector machine," 2018.

[16] N. Guo, R. F. Yen, G. E. Fakhri, and Q. Li, "Svm based lung cancer diagnosis using multiple image features in pet/ct," in *2015 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, pp. 1–4, Oct. 2015.

[17] R. Ramos-Pollán, M. A. Guevara-López, C. Suárez-Ortega, G. Díaz-Herrero, J. M. Franco-Valiente, M. Rubio-del Solar, N. González-de Posada, M. A. P. Vaz, J. Loureiro, and I. Ramos, "Discovering mammography-based machine learning classifiers for breast cancer diagnosis," *Journal of Medical Systems*, vol. 36, pp. 2259–2269, Aug. 2012.

[18] I. El-Naqa, Y. Yang, M. N. Wernick, N. P. Galatsanos, and R. Nishikawa, "A support vector machine approach for detection of microcalcifications in mammograms," in *Proceedings. International Conference on Image Processing*, vol. 2, pp. 953–956, 2002.

[19] E. N. Mortensen and W. A. Barrett, "Interactive segmentation with intelligent scissors," *Graphical models and image processing*, vol. 60, no. 5, pp. 349–384, 1998.

[20] R. Biswas, A. Nath, and S. Roy, "Mammogram classification using gray-level co-occurrence matrix for diagnosis of breast cancer," in *2016 International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE)*, pp. 161–166, Sept. 2016.

[21] Wikipedia, "Artificial neural network," 2018.

[22] G. Templeton, "Artificial neural networks are changing the world. what are they?," 2015.

[23] V. Vryniotis, "Tuning the learning rate in gradient descent," 2013.

[24] K. Suzuki, F. Li, S. Sone, and K. Doi, "Computer-aided diagnostic scheme for distinction between benign and malignant nodules in thoracic low-dose ct by use of massive training artificial neural network," *IEEE Transactions on Medical Imaging*, vol. 24, no. 9, pp. 1138–1150, 2005.

[25] J. Kuruvilla and K. Gunavathi, "Lung cancer classification using neural networks for ct images," *Computer Methods and Programs in Biomedicine*, vol. 113, no. 1, pp. 202–209, 2014.

[26] A. Greensted, "Otsu thresholding," 2010.

[27] S. Singh, J. Harini, and B. R. Surabhi, "A novel neural network based automated system for diagnosis of breast cancer from real time biopsy slides," in *International Conference on Circuits, Communication, Control and Computing*, pp. 50–53, Nov. 2014.

[28] K. Bovis and S. Singh, "Classification of mammographic breast density using a combined classifier paradigm," in *4th international workshop on digital mammography*, pp. 177–180, 2002.

[29] Wikipedia, "Redes neuronales convolucionales," 2018.

[30] Mathworks, "Aprendizaje profundo: Tres cosas que es necesario saber."

[31] B. Rohrer, "How do convolutional neural networks work?," 2016.

[32] B. Rohrer, "An intuitive explanation of convolutional neural networks," 2016.

[33] Kaggle, "Data science bowl 2017: Can you improve lung cancer detection?," 2017.

[34] N. Bayramoglu, J. Kannala, and J. HeikkilÃ¤, "Deep learning for magnification independent breast cancer histopathology image classification," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2440–2445, Dec. 2016.

[35] Wikipedia, "Histopathology," 2017.

[36] A. Marchesi, A. Bria, C. Marrocco, M. Molinara, J. J. Mordang, F. Tortorella, and N. Karssemeijer, "The effect of mammogram preprocessing on microcalcification detection with convolutional neural networks," in *2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS)*, pp. 207–212, June 2017.

[37] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[38] L. lab, "Convolutional neural networks (lenet)," 2017.

[39] J. P. Nameer Hirschkind and J. Khim, "Convolutional neural networks. brilliant.org."

[40] ujjwalkarn, "An intuitive explanation of convolutional neural networks."

[41] J. P. Adam Gibson, "Chapter 4. major architectures of deep networks."

[42] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

[43] A. Budhiraja, "Dropout in (deep) machine learning," 2016.

[44] "Keras: The python deep learning libary."

[45] "Tensorflow. an open source machine learning framework for everyone."

[46] Wikipedia, "Agrupamiento jerÃ¡rquico — wikipedia, la enciclopedia libre," 2018. [Internet; descargado 9-julio-2018].

[47] "Cs231n convolutional neural networks for visual recognition. tranfer learning."

[48] "Keras applications."

[49] "Lung nodule analysis 2016," 2016.

[50] "The lung image database consortium," 2018.

[51] "Simpleitk."

[52] Wikipedia, "Escala hounsfield — wikipedia, la enciclopedia libre," 2017. [Internet; descargado 9-julio-2018].

[53] Wikipedia contributors, "Cross entropy — Wikipedia, the free encyclopedia," 2018. [Online; accessed 14-July-2018].

[54] "Google compute engine documentation."

[55] "2nd place solution to 2017 dsb."

[56] Wikipedia contributors, "Gaussian blur — Wikipedia, the free encyclopedia," 2018. [Online; accessed 15-July-2018].

[57] "gcloud overview."