Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Análisis de Técnicas de Deep Learning en la Detección de Nódulos Pulmonares

Estudante: Miguel Escarda Fernández
Dirección: Alfonso Castro Martínez

A Coruña, xuño de 2021.

Para mis padres, mi familia y mis amigos.

Agradecimientos

Agradecer a mi tutor Alfonso tanto la ayuda brindada, como la oportunidad de poder realizar este trabajo en un campo tan emocionante y recientemente relevante como es el deep learning. A toda mi familia y amigos por acompañarme en este proceso.

Resumen

Además de ser uno de los tipos de cáncer más común en la población, el cáncer de pulmón es también uno de los más mortales. Los investigadores indican que una rápida detección mejora en gran medida las posibilidades de superar la enfermedad. Los radiólogos, con la ayuda de las tomografías axiales computarizadas (TACs), pueden detectar nódulos peligrosos en etapas iniciales. Sin embargo, la detección de los nódulos pulmonares en los TACs es una tarea muy dura y que consume mucho tiempo, por esta razón se han creado sistemas CAD (Computer Aided Diagnosis), cuyo fin es facilitar la detección de nódulos. Con el reciente auge de las técnicas de deep learning para la clasificación de imágenes, estas técnicas también se han aplicado para la tarea de detección de nódulos, debido a que sus resultados superan por mucho a los de las técnicas clásicas de detección. En este trabajo se revisarán y evaluarán arquitecturas y algoritmos de Deep Learning que son propuestos como sistemas CAD para la detección de nódulos pulmonares.

Abstract

In addition to being one of the most common types of cancer in the population, lung cancer is also one of the most deadly. Researchers indicate that rapid detection greatly improves the chances of overcoming the disease. Radiologists with the help of computed axial tomography (CT) scans can detect dangerous nodules in early stages. However, the detection of pulmonary nodules in CT scans is a very difficult and time-consuming task, for this reason CAD (Computer Aided Diagnosis) systems have been created, the purpose of which is to facilitate the detection of nodules. With the recent rise of deep learning techniques for image classification, these techniques have also been applied for the nodule detection task, because their results far exceed those of classical detection techniques. In this work, Deep Learning architectures and algorithms that are proposed as CAD systems for the detection of pulmonary nodules will be reviewed and evaluated.

Palabras clave:

- Aprendizaje Profundo
- Tomografía computarizada
- Detección de nódulos pulmonares
- Diagnóstico Asistido Ordenador

Keywords:

- Deep Learning
- CT scan
- Lung nodule detection
- CAD

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introdución

En este capítulo se comienza explicando el motivo de la realización de este proyecto, el por qué de la utilización de técnicas de deep learning, los objetivos específicos, la estructura de esta memoria, y la metodología y planificación seguidas.

1.1 Motivación

Se conoce como cáncer al conjunto de enfermedades relacionadas con un proceso descontrolado en la división celular. Este puede empezar en cualquier parte del cuerpo humano, y ocurre cuando se descontrola el proceso del crecimiento y división de las células. Cuando una célula se daña, envejece o muere es sustituida por una nueva célula, que surge del crecimiento y división de otra preexistente. Sin embargo, en el cáncer, este proceso ordenado se descontrola. Cuanto más anormales se hacen las células viejas o dañadas, estas sobreviven cuando deberían morir, y células nuevas se forman cuando no son necesarias. Estas células adicionales pueden dividirse sin interrupción y formar masas llamadas tumores.

Los tumores que forman estas nuevas células pueden ser benignos o malignos. Los tumores malignos son cancerosos, y tienen la capacidad de extenderse a tejidos cercanos, pudiendo llegar a invadirlos. Cuanto más crece un tumor, más posibilidades hay de que una de sus células se desprenda y se desplace a lugares distantes del cuerpo por medio del sistema circulatorio o del sistema linfático, lo que conlleva la formación de nuevos tumores lejos del tumor original. Aún cuando estos tumores son extirpados, pueden volver a crecer y desarrollarse dentro del organismo.

Al contrario que los malignos, los tumores benignos no se extienden a los tejidos cercanos y no los invaden. Sin embargo, a veces pueden ser bastante grandes. Al extirparse, generalmente no vuelven a crecer. A pesar de la no peligrosidad de los tumores benignos en otras partes del cuerpo, pueden poner la vida en peligro, por ejemplo, cuando se encuentran en el cerebro.

Los últimos datos disponibles a nivel mundial, estimados dentro del proyecto GLOBOCAN [1], indican que en el año 2018 hubo aproximadamente 18,1 millones de casos nuevos de cáncer en el mundo y se estima que esta cifra crezca hasta los 29,5 millones en el 2040, tal como se indica en la Figura 1.1. Los tumores más frecuentes diagnosticados en el mundo en el año 2018 (Figura 1.3) fueron los de pulmón (11,6%), mama (11,6%), colon y recto (10,2%), próstata (7,1%) y estómago (5,7%). El número de muertes relacionadas con tumores a nivel mundial en el año 2018 fue de 9.555.027 personas y se estima que para el año 2040 este número llegue hasta las 16.388.459 personas. En la Figura 1.2 se puede ver la evolución del número de muertes anuales prevista hasta el 2040. Los tumores responsables del mayor número de fallecimientos a nivel mundial en el 2018 (Figura 1.4) fueron el cáncer de pulmón (18,4% del total de muertes por cáncer), el cáncer colorrectal (9,2%), el cáncer de estómago (8,2%) y el cáncer de hígado (8,2%).

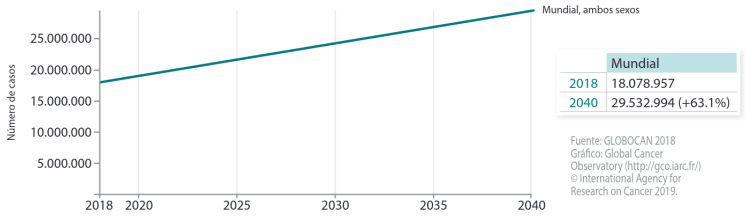


Figura 1.1: Incidencia estimada de tumores en la población mundial para el periodo 2018-2040, ambos sexos.

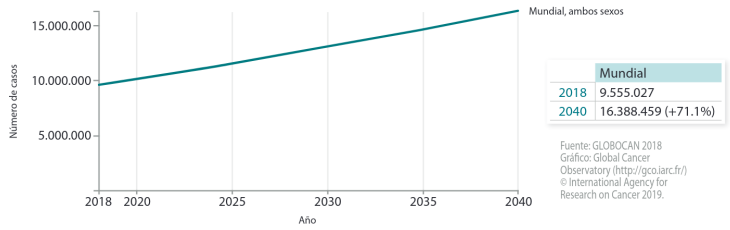


Figura 1.2: Número de fallecimientos por tumores en la población mundial para el periodo 2018-2040, ambos sexos.

Se estimó que en España para el año 2020 se diagnostiquen alrededor de 277.394 casos (de los cuales el 61% se da en mayores de 65 años), según los cálculos de REDECAN [2], siendo los más frecuentes los de colon y recto (44.231 nuevos casos), próstata (35.126), mama (32.953), pulmón (29.638) y vejiga urinaria (22.350). Además, se indicó que la principal causa de muerte por tumores en España en el 2020 será el cáncer de pulmón, mucho más frecuente en hombres que en mujeres (por cada cuatro hombres que lo padecen, hay una mujer afectada). La tasa de supervivencia en los primeros 5 años es del 17,4%. Cabe destacar que su rápida detección en

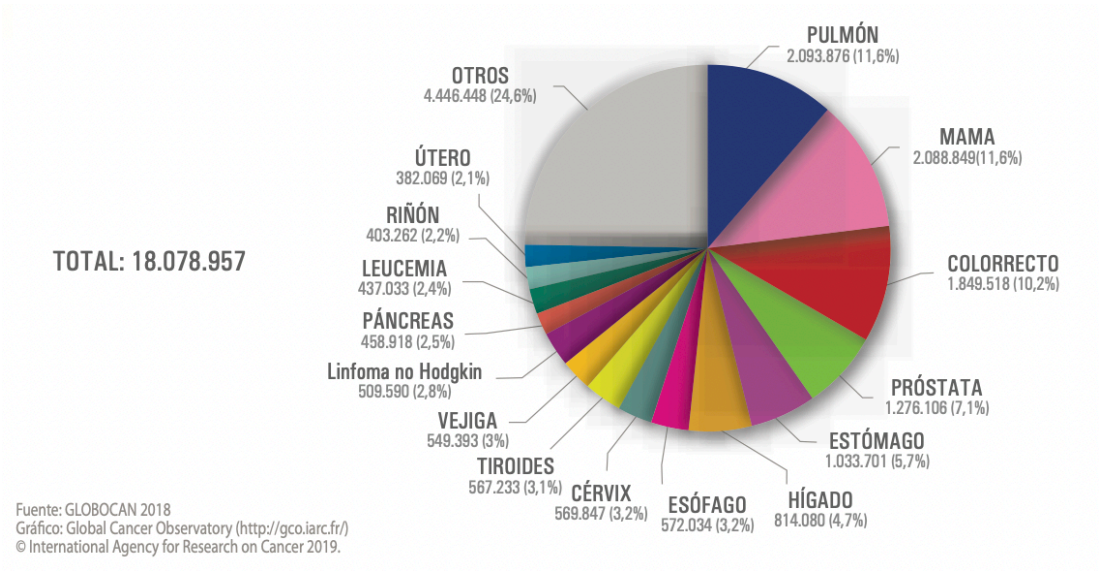


Figura 1.3: Tumores más frecuentemente diagnosticados en el mundo. Estimación para el año 2018, ambos sexos.

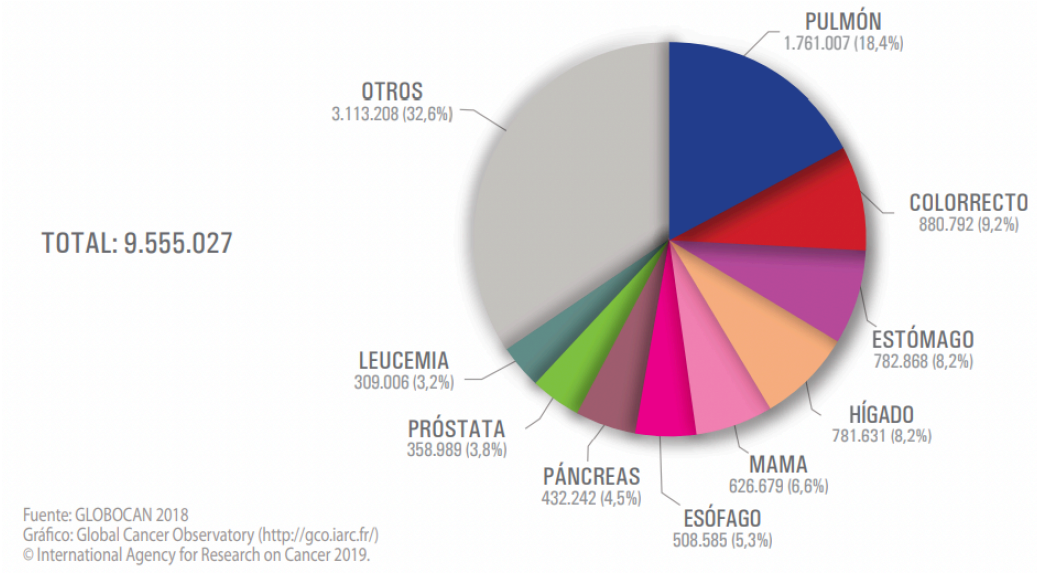


Figura 1.4: Estimación del número de fallecimientos por tumores en el mundo en el año 2018, ambos sexos.

los estadios iniciales aumenta las posibilidades de supervivencia a 5 años hasta el 54,8%.

El 2021 es un caso algo diferente ya que trae consigo las consecuencias todavía desconocidas de todo un año de pandemia mundial. Con la llegada del Covid-19 muchos países vieron afectada su capacidad de diagnóstico de cáncer; entre ellos España, que sufrió y sufre enormes dificultades derivadas de la pandemia.

Es probable que, con motivo de esta situación, el número de cánceres detectados en el año 2020 fuese menor que el que se estimaba en un principio. Esto es debido a las tensiones sufridas en el sistema sanitario, que hicieron que empeorase la efectividad de los programas de cribado y de su propia capacidad diagnóstica. Además, la vigencia de sucesivos estados de alarma y el miedo de los ciudadanos a acceder a centros sanitarios, provocan que tumores que podían haber sido detectados a tiempo lo sean tarde o incluso que no lleguen a hacerlo. [3] Por episodios como este es todavía más importante que los sistemas de diagnóstico de tumores sean cada vez más precisos en su tarea de ayudar a los radiólogos en su detección.

Todo esto hizo que se tambaleasen las estimaciones para el año 2020 y aún se desconoce cómo afectará a las del 2021. Aún así, REDECAN hace sus estimaciones del año 2021. Se espera que haya un total de 276.239 cánceres incidentes (un 61,1% de ellos en mayores de 65 años). El más frecuente seguirá siendo el de colon y recto (43.581 casos), seguido del de próstata (35.764), mama femenina (33.375) y pulmón (29.549), y más lejos el de vejiga urinaria (20.613).

Observando los datos anteriores se puede constatar que, aunque el cáncer de pulmón no es el cáncer más común (tiene una elevada incidencia), sí que es el que tiene una mayor tasa de mortalidad. Debido a esto y a la alta complejidad de la estructura pulmonar, que dificulta su análisis, la detección de nódulos pulmonares ha sido una de las áreas más estudiadas dentro del análisis de imágenes médicas. La investigación se centra principalmente en la detección de los nódulos en los primeros estadios de la enfermedad, cuando se presentan en regiones con un diámetro menor a 3mm. Además, no sólo se busca identificarlos, sino que su correcta clasificación es fundamental. Aquí es notorio el aporte de los radiólogos, identificando los factores clínicos (forma, densidad, localización, evolución,...) que definen su grado de malignidad y benignidad.

El método más utilizado para la detección de nódulos es la TC (tomagrafía computarizada). En la Figura 1.5 se observan cortes de TC con ejemplos de distintos tipos de nódulos. En cada estudio TC puede llegar a haber más de 500 imágenes, lo que hace que su inspección sea una tarea muy dura y que consume mucho tiempo, ya que cada una de estas imágenes le lleva de media a un radiólogo entre 2 y 3,5 minutos examinarla. Cuando un sólo radiólogo examina el escáner, solamente el 68% de las veces los nódulos pulmonares cancerígenos son correctamente diagnosticados, este porcentaje aumenta al 82% cuando son dos los radiólogos que lo examinan [4]. Con el fin de ayudar y reducir la carga de trabajo de los radiólogos, se diseñaron los sistemas CAD (Computer aided detection), que en sus versiones actuales intentan discernir entre nódulos malignos y benignos. Se observó que para la clasificación de imágenes, el uso de técnicas de deep learning produjo resultados que eran muy superiores con respecto a las técnicas tradicionales y además el nivel de procesado necesario para las imágenes podía ser inferior para obtener los mismos o mejores resultados.

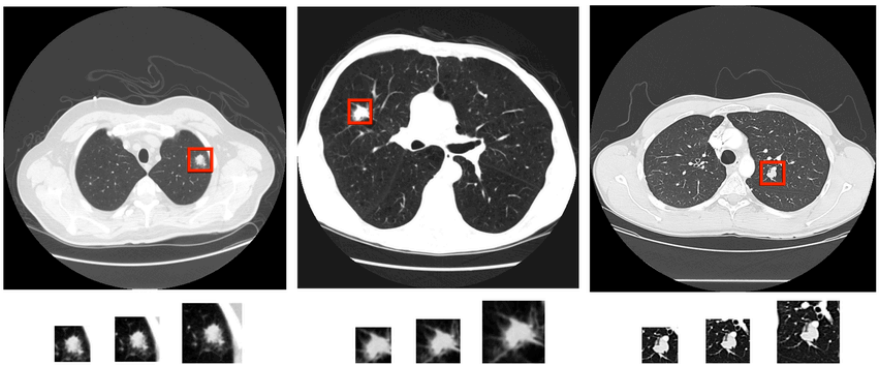


Figura 1.5: Categorías de nódulos pulmonares; benigno; maligno primario; maligno metastásico (de izquierda a derecha).

1.2 Objetivos

El objetivo concreto de este proyecto es analizar distintas técnicas de deep learning para la detección de nódulos pulmonares en estudios TC torácicos. Para conseguir este objetivo primero se necesita una base sobre el problema y sobre las técnicas utilizadas para conseguir soluciones. Esto conseguirá que el lector obtenga un conocimiento de grano grueso sobre el cáncer de pulmón y de grano fino sobre las técnicas recientes en visión artificial utilizando CNNs.

Se analizarán, revisarán y compararán las principales técnicas de detección de nódulos utilizando deep learning, finalizando con la realización de un experimento para la detección de los nódulos, utilizando alguna de las técnicas explicadas.

1.3 Organización Memoria

La organización de la memoria ha sido ideada con el objetivo de facilitar la comprensión a cualquier lector, a grandes rasgos, de todos los conceptos que se mencionan a medida que avanza en este TFG. Para ello se realizará una pequeña introducción sobre qué es el cáncer de pulmón, sus tipos, tratamientos, diagnóstico... Después se explicará el concepto de deep learning haciendo un pequeño repaso de su historia, de su uso, y de sus bases y principios. Detallando de forma minuciosa las técnicas empleadas en la actualidad, explicando su funcionamiento y aplicaciones. A continuación, se realizará la revisión del estado del arte centrándose en explicar algunas de las técnicas de deep learning utilizadas hasta la fecha para la detección de nódulos pulmonares. Por último, se hablará sobre las pruebas realizadas y las conclusiones a las que se ha llegado.

1.4 Metodología y planificación

En todo proyecto software es necesario un proceso de planificación con el fin de establecer un conjunto de medidas con las que estimar los recursos y horas necesarias para la realización del mismo. Esta planificación debe actualizarse a medida que el proyecto avanza, dejando constancia de los hitos realizados, su duración en tiempo y su coste en recursos, para que una vez finalizado el proyecto pueda compararse con los valores estimados en un principio.

A continuación se detallará la explicación sobre la planificación del proyecto, definiendo los conceptos de la metodología seguida, y lo que se ha realizado. Destacando aquí que, si bien en muchos de los trabajos de fin de grado la redacción de la memoria sigue el orden cronológico del proceso del proyecto (explicando detalladamente el proceso seguido para cada iteración), se ha optado por distanciarse de este estándar común, y primero presentar la planificación que se ha seguido, al considerar esta memoria como el producto del proyecto por ser este un TFG de experimentación.

Se optó por seguir una metodología Scrum para el desarrollo del proyecto, siendo el alumno el único desarrollador en él. Antes de comenzar el proyecto se llevó a cabo un proceso de conceptualización del mismo. Al no ser este un proyecto al uso, la metodología fue adaptada al problema a abordar. El tutor asumió el rol de product owner y se fijaron sprints mensuales. Para cada sprint se realizó una reunión donde se establecían los requisitos de dicho sprint y se planificaba el avance del mismo hasta la siguiente reunión. Al haber un sólo desarrollador que también asumió el puesto de Scrum Master no se hicieron reuniones diarias.

A continuación se hará una breve introducción sobre lo que es Scrum.

1.4.1 Scrum

Scrum es un marco de trabajo en el que se aplican un conjunto de buenas prácticas o reglas para trabajar en equipo, con el objetivo de obtener el mejor resultado posible en un proyecto software. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

Los proyectos con metodología Scrum siguen una ejecución cíclica, con iteraciones cortas y de longitud fija. Estas iteraciones suelen ser de 2, 3 o 4 semanas, y cada una de ellas debe proporcionar un resultado completo al final, que recibe el nombre de incremento. Este incremento debe ser susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

El plan del proyecto se realiza a partir de la lista de requisitos del producto (product backlog). El cliente (Product Owner) prioriza los requisitos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas.

Roles

- **Product Owner:** Es el cliente o su portavoz, trabajan juntos, encargado de gestionar el product backlog.
- **Scrum Master:** Líder a nivel Scrum del equipo, aunque sin poder jerárquico. Su función es ayudar al equipo y gestionar la correcta implementación de la metodología en el equipo.
- **Equipo:** Grupo de personas que tienen como objetivo el desarrollo del producto.

Eventos

- **Sprint:** Nombre de la iteración en Scrum, se busca que todas tengan una dificultad similar y que al final de cada una de ellas se entregue un producto funcional y testeable.
- **Sprint Planning:** Reunión para la planificación del siguiente Sprint basándose en lo ocurrido en el anterior. La duración está preestablecida.
- **Daily:** Reunión diaria del equipo, llevada a cabo por el Scrum master. Su función es recibir feedback del equipo y en caso de que se encuentren problemas, buscar formas de solucionarlos ayundándose entre los distintos miembros del equipo.
- **Revision Sprint:** Reunión para analizar los requisitos implementados y hacer cambios en los siguientes si es necesario. Útil para saber la evolución del producto por parte del product owner.

Artefactos scrum

- **Product backlog:** Documento de alto nivel para todo el proyecto. Es el conjunto de todos los requisitos de proyecto, el cual contiene descripciones genéricas de funcionalidades deseables, priorizadas según su retorno sobre la inversión. Representa el qué va a ser construido en su totalidad. Es abierto y solo puede ser modificado por el product owner. Contiene estimaciones realizadas a grandes rasgos, tanto del valor para el negocio, como del esfuerzo de desarrollo requerido.
- **Sprint backlog:** Subconjunto de requisitos que serán desarrollados durante el siguiente sprint. Al definir el sprint backlog, se describe el cómo el equipo va a implementar los requisitos durante el sprint. Las tareas en el sprint backlog nunca son asignadas, son tomadas por los miembros del equipo del modo que les parezca adecuado.
- **Incremento:** Producto final utilizable de un sprint.

1.4.2 Scrum en el proyecto

Para el desarrollo del proyecto lo primero que se hizo fue la realización del product backlog, que como en este caso no existe cliente, el rol lo asumió el tutor. La duración de los sprints se estableció en 4 semanas para poder compatibilizar las tareas de docencia del tutor como la de trabajo del alumno, además el tiempo de trabajo mínimo por semana fue de 20 horas. El primer día de cada sprint se realiza la selección de requisitos y la planificación del sprint. Al sólo contar con una persona como equipo de desarrollo no se han realizado dailys. El último día de cada sprint se realiza la revisión del sprint (para ver todo lo logrado hasta ese momento) y la retrospectiva (para ver cómo ha sido el proceso de trabajo del desarrollador), siendo el tutor la persona encargada de evaluar el trabajo. Para el total del proyecto se establecieron 4 sprints o iteraciones. Al ser la memoria en sí el producto de este proyecto, la misma era entregada al final de cada sprint para su revisión como incremento.

- **Sprint 1:** Ideado por el tutor como una primera toma de contacto con el problema, lo principal fue buscar información del problema e idear una forma de exponerlo. Por lo que la idea de cómo exponer este trabajo surgió en esta iteración tras observar otros trabajos y ver cómo se organizaban. La revisión consistió en una reunión con el tutor para ver si se habían cumplido los objetivos y el conocimiento del problema era el suficiente como para pasar a la siguiente iteración. La idea básica de la revisión fue extendida en los demás sprints.

- **Sprint 2:** De la misma forma, el sprint planning lo realizó el tutor. Esta vez el objetivo era conocer la historia de la inteligencia artificial y del deep learning, además buscar información sobre los principales métodos para la detección de objetos utilizando CNNs. Se buscó obtener un conocimiento profundo del funcionamiento de las CNNs y demás arquitecturas mencionadas en el trabajo.

- **Sprint 3:** Se hizo una recopilación de trabajos para la detección de nódulos pulmonares utilizando deep learning. Los que se muestran en el proyecto buscan mostrar que en el estado del arte actual existen trabajos con distinto nivel de dificultad, además de diferenciarse al usar algunos técnicas 2D, otros 3D y otros mixtas.

- **Sprint 4:** Los requisitos aquí fueron encontrar dos arquitecturas: una 2D y otra 3D, y buscar cómo adaptar la información de la que se disponía a las arquitecturas, para acabar probando dos sistemas distintos para la detección de los nódulos. Por tanto hubo que trabajar en dos ramas distintas cuyos procesos si bien eran parecidos no eran iguales.

Capítulo 2

Contexto

En este capítulo se exponen los diferentes conceptos usados en el proyecto. Se habla del cáncer de pulmón, sus tipos, diagnóstico y tratamientos.

2.1 El cáncer de pulmón

El cáncer de pulmón es una enfermedad provocada por el desorden de los mecanismos de control de división celular, por el que se produce un crecimiento incontrolado del tejido pulmonar llegando a formar tumores o nódulos. En función de la capacidad de invasión y afectación de las células tumorales a órganos y tejidos periféricos se pueden clasificar según su malignidad en: tumores benignos (sin capacidad de afectación ni alteración de otros órganos), y tumores malignos (con capacidad para infiltrarse en tejido adyacente (infiltración) e invadir, penetrar y destruir otros tejidos del organismo (metástasis)). Además tal como podemos observar en la Figura 2.2, también se pueden clasificar según su morfología en: sólidos y subsólidos (Figura 2.1).

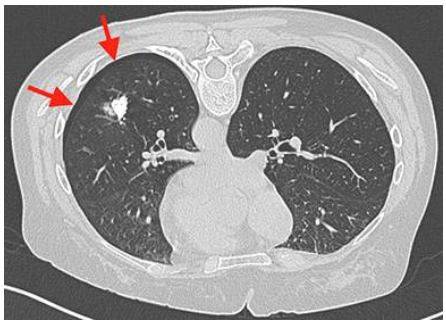


Figura 2.1: Nódulo subsólido.

Dependiendo del tipo de célula donde se origine el tumor, se puede hablar de un tipo u otro de cáncer. El 90% de ellos son generados por células epitaliales y reciben el nombre de

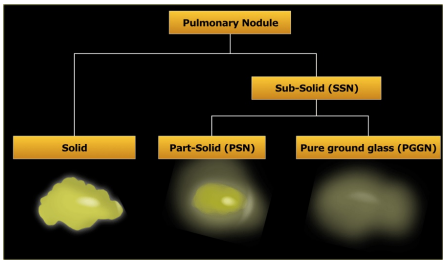


Figura 2.2: Clasificación de los nódulos en función de su morfología.

carcinomas. Los sarcomas son los derivados de células del tejido conectivo o muscular. Las leucemias, linfomas, y mielomas son originados por células de la sangre. Los neuroblastomas y gliomas son los derivados de células del sistema nervioso [5].

2.1.1 Carcinomas

Los carcinomas son el tipo de cáncer de pulmón más común, se originan en células malignas de tipo epitelial o glandular. Abarcan el 90-95% de los casos de tumores malignos primarios de pulmón y se pueden dividir en dos categorías histológicas, categorizadas por tamaño y apariencia:

Carcinoma de células pequeñas (microcítico)

En inglés Small Cell Lung Cancer, SCLC. Relacionados con el consumo de tabaco, suponen el 20% total de casos. Originado normalmente en los bronquios centrales, es la forma más agresiva y de peor diagnóstico en cuanto a supervivencia, ya que, aunque las células cancerosas son pequeñas, se produce un crecimiento rápido llegando a formar tumores grandes provocando metástasis en otros órganos.

Carcinoma de células no-pequeñas (no microcítico)

En inglés Non-Small Cell Lung Cancer, NSCLC. Suponen el 80% del total de casos, siendo el más común. Existen tres principales clases:

- **Adenocarcinoma**. Asociado al tabaco pero presente en no fumadores con enfermedades pulmonares previas. La mayoría de los casos se presentan como una lesión periférica sobre todo en el lóbulo superior.

- **Carcinoma epidermoide o escamoso**. Originado normalmente en los bronquios centrales y presenta una capacidad de metastatizar inferior al resto de tipos, por lo que si es tratado a tiempo el pronóstico es relativamente bueno.

- **Carcinoma de células grandes**. También asociado al tabaco, presenta un comportamiento más agresivo que el adenocarcinoma, con mayor rapidez de crecimiento y de metastatizar. Su apariencia es similar al adenocarcinoma con presencia en zonas periféricas.

Los restantes tumores suponen del 10% al 5%. Comprenden carcinomas de baja malignidad (carcinoides y neoplasias de las glándulas bronquiales), otros procedentes de estructuras mixtas, endodérmicas y mesodérmica.

2.1.2 Causas

El tabaco es el agente causante del cáncer de pulmón en más del 80% [6] de los casos. En los últimos años se ha dado un claro descenso en varones, debido a la reducción del hábito en ellos, mientras que se ha dado un aumento en mujeres por su incorporación al mismo.

Las probabilidades que tiene un fumador crónico de padecer un cáncer de pulmón a lo largo de su vida puede alcanzar el 30%, mientras que en no fumadores es del 1%. El riesgo depende del número de cigarrillos fumados al día, así como del número de años que se fuma.

Al dejar de fumar, el riesgo de desarrollar un cáncer de pulmón disminuye conforme pasan los años, aunque se mantienen los índices de riesgo varios años después de abandonar el hábito.

Otros factores, además del tabaquismo pasivo, incluyen la exposición al amianto, hidrocarburos aromáticos policíclicos, arsénico y níquel, así como padecer otras enfermedades pulmonares, tales como la enfermedad pulmonar obstructiva crónica (EPOC) y/o fibrosis pulmonar.

2.1.3 Etapas del cáncer de pulmón

Se pueden diferenciar dos tipos de clasificaciones en función del tipo de cáncer de pulmón [7].

- **Cáncer no microcítico**
  Su clasificación sigue el sistema TNM. La estadificación del tumor permite distinguir entre los pacientes con posibilidades de curación de los pacientes sin posibilidad de curación, la estadificación consiste en determinar el tamaño del tumor y si se ha diseminado o no. Se determina en base a los síntomas del paciente, los estudios de imágenes (escáneres TC) y los resultados de las biopsias. Además permite calcular la probabilidad de curarse. La letra T hace referencia al tamaño del tumor. Se clasifica entre T1 y T4, según el tumor sea más voluminoso o afecte estructuras cercanas importantes como los bronquios principales, las arterias o el propio corazón. La N indica si están o no afectados los ganglios linfáticos cercanos. N0 significa que no lo están. La afectación

de los ganglios es un factor pronóstico muy importante que se gradúa de N1 a N3. En particular es vital conocer si están o no invadidos los ganglios más centrales del tórax, una región conocida como mediastino. Por regla general, la afectación del mediastino significa que el tumor es inoperable. La M indica si no hay metástasis (M0) o, por el contrario, si el cáncer ya se ha extendido a otros órganos (M1). Las distintas combinaciones de estos parámetros se agrupan en lo que se conoce como estadios de la enfermedad, que van desde el 0 al 4 según se encuentre el cáncer en fase temprana, estén o no afectados los ganglios linfáticos, haya invasión del mediastino, o esté ya diseminado afectando a otros órganos.

- **Cáncer microcítico**
  Para los tumores microcíticos se habla de etapa limitada y capa extensa. La etapa limitada significa que el tumor se encuentra limitado en el hemitórax de origen, el mediastino y los ganglios supraclaviculares. Esto sería un campo tolerable para el empleo de radioterapia. La etapa extendida es aquella en la que el cáncer está demasiado diseminado para ser incluido dentro de la definición de etapa limitada, es decir, el cáncer se ha extendido al otro pulmón, a los ganglios linfáticos del otro pecho, a órganos distantes, etc.

2.1.4 Diagnóstico

Para diagnosticar a un paciente con cáncer de pulmón es necesaria la realización de pruebas diagnósticas de imagen, como los TC torácicos, en los que se profundizará más adelante, pero además en la mayoría de casos también es necesario realizar estudios endoscópicos, como las broncoscopias, para tomar una muestra de tejido y conocer el tipo de tumor, su pronóstico y el tratamiento más adecuado.

Análisis de sangre y orina

Son las primeras pruebas que se realizan. Con ellas se busca conocer el estado general del paciente, si tiene o no alteraciones de la función renal o hepática.

Citología de esputo

Consiste en analizar con el microscopio el tipo de células que existen en el esputo del paciente. El esputo del paciente es el moco que se expulsa de los pulmones al toser.

Broncoscopia

Se utiliza para examinar la tráquea y los bronquios directamente desde el interior de los mismos. Se realiza mediante la introducción de un tubo flexible llamado broncoscopio en

las vías respiratorias. El interior del tubo contiene fibra óptica que permite a los especialistas visualizar todo el recorrido desde un monitor. Tiene además un mecanismo que permite tomar muestras de las lesiones sospechosas, para posteriormente analizarlas con el microscopio.

En algunas ocasiones, cuando las lesiones no son visibles, a través del broncoscopio se introducen líquidos para lavar la zona y una vez aspirados se analizan con el microscopio las células que contienen.

Para la realización de esta prueba el paciente debe estar en ayunas. El neumólogo, médico especialista en pulmón, antes de introducir el broncoscopio, debe anestesiar la zona de la garganta, laringe, tráquea y bronquios.

La broncoscopia también aporta información sobre el lugar donde se asienta el tumor y las estructuras a las que afecta.

Punción de aguja fina

Consiste en obtener células de la lesión mediante la realización de una punción torácica con aguja fina, generalmente bajo control de una tomografía computarizada o de una ecografía.

Se realiza un corte en el tórax del paciente para introducir la aguja y extraer la muestra que se envía a analizar. Esta técnica se utiliza cuando el tumor se halla en una zona periférica del tórax a la que es difícil acceder mediante la broncoscopia.

Al finalizar el procedimiento, se realiza una radiografía del tórax para asegurarse de que no haya aire que se escapa de los pulmones al pecho (lo que podría provocar un neumotórax).

2.1.5 Tratamientos

Existen varios tipos de tratamiento en función del tumor detectado y su localización [8].

Tratamiento quirúrgico

Hay que valorar la resecabilidad de la lesión y la operabilidad del paciente para saber si puede tratarse mediante cirugía y qué procedimiento utilizar: la lobectomía consiste en la extirpación de un lóbulo del pulmón y puede utilizarse si el paciente conserva una capacidad respiratoria adecuada, la segmentectomía es la extirpación de parte de un lóbulo y la neumonectomía es la extirpación del pulmón entero.

Tratamiento con quimioterapia

La quimioterapia es un tratamiento para el cáncer que usa medicamentos para interrumpir la formación de células cancerosas, ya sea mediante su destrucción de las células o al impedir

su multiplicación. Dependiendo del caso se puede ingerir de forma oral, inyectar en vena o en un músculo.

Distintos tipos: tratamiento adyuvante para pacientes tras una cirugía completa; tratamiento neoadyuvante previo a una cirugía para la disminución del tamaño del tumor; tratamiento paliativo en el contexto de la enfermedad diseminada.

Esencial un diagnóstico histológico y/o molecular lo más preciso posible para que el tipo de tratamiento de quimioterapia sea el más recomendado a ese caso.

Tratamiento con radioterapia

La radioterapia es un tratamiento para el cáncer que usa rayos X de alta energía u otros tipos de radiación para destruir las células cancerosas o impedir que crezcan.

Puede realizarse como tratamiento combinado con la quimioterapia, se utiliza para pacientes con tumores no operables por localización y/o tamaño, si el paciente no es candidato para un tratamiento quirúrgico, o también en pacientes con metástasis.

Tratamiento con inmunoterapia

Uno de los mayores avances en el tratamiento de cáncer de pulmón, al no tener que utilizarse acompañada con quimioterapia o radioterapia. La inmunoterapia es un tratamiento en el que se usa el sistema inmunitario del paciente para combatir el cáncer. Se usan sustancias elaboradas por el cuerpo o en el laboratorio para impulsar, dirigir o restaurar las defensas naturales del cuerpo contra el cáncer. Este tipo de tratamiento para el cáncer también se llama bioterapia o terapia biológica.

2.2 Tomografía computarizada, TC

La tomografía computarizada, TC (Figura 2.3), es una técnica de diagnóstico no invasiva, basada en el uso de rayos X, mediante el cual se obtienen múltiples cortes axiales del cuerpo del paciente. En la actualidad se refiere a ella como TC y no TAC (tomografía axial computarizada) porque es posible obtener imágenes de cortes tomagráficos reconstruidas en planos no transversales.

La medición de la absorción y atenuación de los rayos X a través del paciente mediante un gran número de proyecciones es el principio básico de esta prueba. Estas proyecciones se efectúan mediante el uso de una o dos fuentes de rayos X que rotan alrededor del paciente y un conjunto de receptores a lo largo del arco que mide la atenuación de la radiación en diferentes ángulos. La representación final de la TC se obtiene tras aplicar algoritmos de reconstrucción a lo capturados por los receptores.

Los valores de los píxeles de las imágenes generadas se calculan en base de la ”Ley de Beer-Lamber”, que establece la relación entre la intensidad del haz inicial de rayos, el coeficiente de atenuación lineal, el espesor del material y la intensidad del haz atenuador de rayos X. En base a estos parámetros la imagen resultante se considera como una matriz de diferentes coeficientes de atenuación lineal. La resolución del sistema de ecuaciones resultante se realiza mediante el uso del algoritmo de retro-proyección filtrada, que constituye el estándar para la reconstrucción de la imagen y es realizado por ordenador.

La matriz de reconstrucción de los valores de atenuación lineales se trasforma en una matriz de números de TC (tomografía computarizada), medidos en unidades Hounsfield (HU). Tabla 2.1.

Sustancia	**HU**
Aire	-1000
Parénquima pulmonar	-700 a -600
Grasa	-100 a -55
Agua	0
Líquido cerebroespinal	15
Hígado	30
Sangre	30 a 45
Músculo	10 a 40
Riñón	40 a 60
Tejido blando	100 a 300
Hueso	700 a 3000

Tabla 2.1: Valores para la escala HU.

En 1976, se introdujo la tomografía axial computarizada, TAC, en la que la adquisición se realizaba mediante una rotación del tubo de rayos X y el uso de una única fila de detectores con cientos de elementos. Esta técnica permitía realizar una exploración completa de TC a partir de una o más series de cortes axiales que cubrían la zona de interés, en las que después de cada corte axial la camilla se movía.

En 1989, se desarrolla la tomografía computarizada helicoidal que permite, gracias al desplazamiento simultáneo de la camilla y la rotación de la fuente de rayos X, la adquisición continua de las imágenes en una sola inspiración del paciente. Las principales ventajas que se consiguen con esta técnica son la reducción de tiempo, una mejor resolución y una mejora en la reconstrucción de imágenes multiplanares en 3D.

En años posteriores, las mejoras han venido de la mano de los dispositivos TC multicorte (TCMD). En estos escáneres se incorporan varias filas de detectores, de 4 a 64, lo que hace posible la adquisición simultánea de varias regiones en tiempos de rotación muy pequeños de entre 0,3-0,4s. De este modo, se permite adquirir el cuerpo completo de una persona en

una única inspiración con grosores de cada corte inferiores a 1 mm. Los equipos multicorte más recientes incorporan varios conjuntos de filas de detectores y doble fuente de rayos X, permitiendo reducir los tiempos de adquisición y exposición a la radiación.

Figura 2.3: Imagen de una máquina TAC.

En la actualidad, la tomografía computarizada se ha convertido en una de las modalidades de diagnóstico más empleadas en muchas de las fases del flujo de la actividad clínica como el diagnóstico, planificación de tratamientos, cribados de poblaciones con riesgo y evolución de enfermedad.

Capítulo 3

Deep learning

En este capítulo se explica el concepto de inteligencia artificial, machine learning y deep learning, también el funcionamiento de alguna de las redes neuronales más usadas en la actualidad para las tareas de detección.

3.1 Inteligencia artificial

¿Es posible que las máquinas piensen? Es la pregunta que da origen a la inteligencia artificial y cuyas ramificaciones aún se siguen explorando a día de hoy. En ”Computing Machinery and Intelligence” (1950) [9], Alan Turing, conocido como el padre de la inteligencia artificial, establece la visión y el objetivo fundamental de la misma. Una de sus definiciones formales es ”la capacidad de un sistema para interpretar correctamente datos externos, para aprender de dichos datos y emplear esos conocimientos para lograr tareas y metas concretas a través de la adaptación flexible” dada por Andreas Kaplan y Michael Haenlein.

Sus comienzos fueron en la década de 1950, aunque años antes ya se había trabajado en ella, no fue hasta que un grupo de investigadores se reunieron en Dartmouth (1956) y acuñaron el término de inteligencia artificial, el responsable del término fue el científico computacional y matemático John McCarthy.

Dentro de la inteligencia artificial originalmente existían dos paradigmas fundamentales, IA simbólica e IA subsimbólica (o conexionista), la diferencia entre ambas radica en cómo se formaliza el conocimiento. Las técnicas simbólicas buscan representar el conocimiento mediante símbolos, mientras que la subsimbólica busca conseguir el conocimiento sin explicitarlo mediante un conjunto de reglas.

Durante el congreso de Dartmouth se hicieron previsiones inalcanzables para los siguientes 10 años en la rama de la IA subsimbólica, lo que provocó un abandono casi total de los investigadores hasta los años 80. Entre los años 50 y 80 el paradigma dominante fue la IA simbólica, cuyo boom se vio en los 80 con los sistemas expertos. Durante un largo período

de tiempo se pensó que al dotar a las máquinas de un conjunto de reglas enorme, podrían llegar a alcanzar el nivel de inteligencia humana, pero con el tiempo se fue observando que las máquinas serían incapaces de reconocer ellas mismas reglas explícitas que permitiesen un mayor nivel de conocimiento. Así se llegó al machine learning.

Actualmente existen diferentes formas de clasificar la IA, una puede ser distinguir entre sistemas fuertes y débiles. Los débiles son los actuales que no poseen una inteligencia sobrehumana y se utilizan para especializarse en tareas que realizan a un nivel experto. Las fuertes, también conocidas como inteligencia artificial general, son aquellas con capacidades similares (o superiores) a las humanas y que pueden realizar diferentes tareas sin estar especializadas en ellas, estos sistemas aún no existen pero muchos expertos creen que para el año 2050 podríamos llegar a tenerlos.

3.2 Machine Lerning

Surge de la pregunta, ¿puede una máquina aprender por su cuenta a realizar un tarea?, y sus primeros orígenes datan de 1830-1840 con el primer ordenador de propósito general, ideado por Charles Babbage, aunque sin que esa fuese la razón para crearlo (para el comienzo de la IA y del ML aún había que esperar más de 100 años). Lo que pretendía era encontrar una forma de automatizar determinados cálculos en el campo del análisis matemático, y el concepto de computación de propósito general aún no se había inventado. Cabe destacar que Babbage nunca vio su máquina construida (que era totalmente analógica) aunque su diseño era absolutamente válido, en la actualidad se puede ver en el museo de ciencia de Londres.

En 1959, el científico de IBM Arthur Samuel acuñó el término de machine learning tras realizar un programa para jugar a las damas que era capaz de aprender mientras jugaba contra sí mismo. El machine learning implica que los ordenadores sean capaces de descubrir cómo realizar tareas para las que no están específicamente programados. El objetivo es que los ordenadores aprendan de los datos proporcionados, aplicando técnicas estadísticas para llevar a cabo determinadas tareas. Cuando las tareas son simples es posible que un programador pueda explicitar el algoritmo necesario para resolver el problema, pero cuando la tarea es más compleja puede resultar imposible o muy difícil para un ser humano crear manualmente los algoritmos necesarios. En la práctica, puede resultar más eficaz ayudar a la máquina a desarrollar su propio algoritmo, en lugar de tener programadores humanos que especifiquen cada paso necesario.

El machine learning emplea varios enfoques para conseguir que una máquina aprenda, los principales son:

- **Aprendizaje supervisado**
  Infiere una función a partir de un conjunto de datos de entrenamiento etiquetados. Cada

ejemplo es un par que consta de un objeto de entrada y un valor de salida deseado. El algoritmo produce un función inferida para mapear nuevos ejemplos, modificando, en función del error producido entre la salida de la red y la salida esperada, los pesos de la red.

- **Aprendizaje no supervisado**
  A partir de un conjunto de datos que contiene sólo entradas (datos sin etiquetar), tratan de encontrar estructura en los datos, agrupándolos y buscando patrones en ellos.

- **Aprendizaje por refuerzo**
  A la red se le da un conjunto de patrones de entrada y se le indica si la salida obtenida es o no correcta. Sin embargo, no se le proporciona el valor de la salida esperada. Este tipo de aprendizaje es muy útil en aquellos casos en que se desconoce cuál es la salida exacta que debe proporcionar la red. Los agentes de software deben realizar acciones en un entorno para maximizar alguna noción de recompensa acumulativa.

3.3 Deep learning

Es una rama del machine learning que se corresponde con una nueva forma de representar el aprendizaje a partir de datos y poniendo énfasis en el aprendizaje de capas sucesivas de representaciones cada vez más significativas. La profundidad de un modelo es el número de capas que contiene. Puede llegar a implicar decenas y centenas de capas sucesivas de representaciones, y todas se aprenden de forma automática con la exposición a los datos de entrenamiento. Estas capas de representaciones suelen ser aprendidas mediante redes neuronales, que son modelos computacionales inspirados en las redes de neuronas biológicas (aunque su funcionamiento es totalmente distinto). En la Figura 3.1 se puede ver un ejemplo de red neuronal y la visualización de representaciones aprendidas para el número 4 que realiza la red.

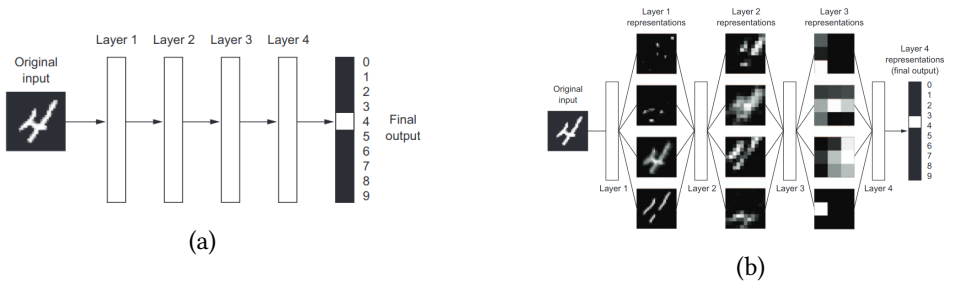


Figura 3.1: Figura (a) Deep Neural Network y (b) Representaciones aprendidas por un modelo de clasificación de dígitos.

Las Redes de Neuronas Artificiales (RNA) basan su funcionamiento en las redes de neuronas biológicas, constan de un conjunto de unidades o nodos conectados que reciben el nombre de neuronas artificiales. Cada conexión, como la sinapsis en un cerebro biológico, puede transmitir una ”señal” a otras neuronas. Una neurona artificial, que recibe una señal, luego la procesa y puede señalar a las neuronas conectadas a ella. La ”señal” en una conexión es un número real, y la salida de cada neurona se calcula mediante alguna función no lineal de la suma de sus entradas. Las conexiones se llaman bordes. Las neuronas y los bordes suelen tener un peso que se ajusta a medida que avanza el aprendizaje. El peso aumenta o disminuye la fuerza de la señal en una conexión. Las neuronas pueden tener un umbral tal que una señal se envía solo si la señal agregada cruza ese umbral. Normalmente, las neuronas se agregan en capas. Las diferentes capas pueden realizar diferentes transformaciones en sus entradas. Las señales viajan desde la primera capa (la capa de entrada) hasta la última capa (la capa de salida).

En ”A Logical Calculus of Ideas Immanent in Nervous Activity” (1943)[10], Warren McCullough y Walter Pitts, propusieron el primer modelo matemático para la construcción de una red neuronal. La universidad de Standford creó en 1960 las redes ADALINE y MADALAINE, esta última eliminaba el ruido en las llamadas telefónicas y era una red ADALINE con múltiples capas. Podemos ver los distintos hitos del deep learning en la Figura 3.2.

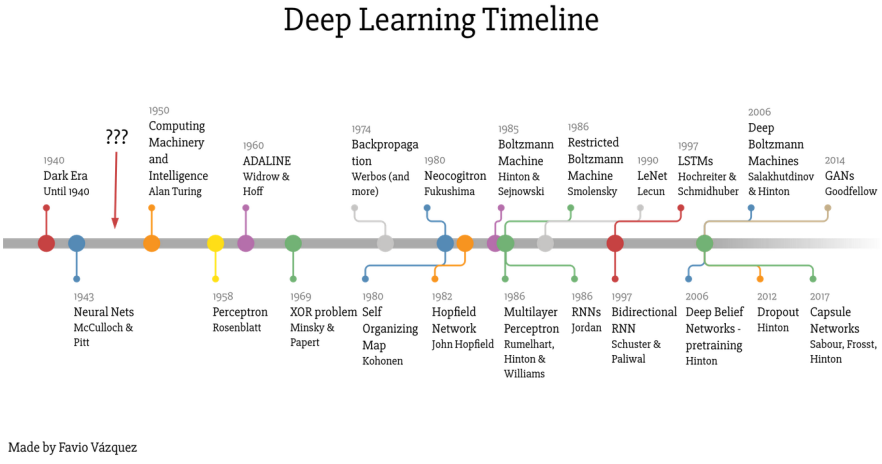


Figura 3.2: Evolución del Deep Learning.

3.4 Perceptrón

El perceptrón (1958) [11] fue creado por Frank Rosenblatt como evolución a la neurona de McCulloch-Pitts [10] y se refiere a la unidad básica de inferencia en forma de discriminador lineal, es decir, la forma más simple de una red neuronal usada para problemas de clasifica-

ción de patrones linealmente separables. Básicamente es una neurona con pesos sinápticos y umbral ajustables. Si los patrones usados para entrenar el perceptrón son sacados de dos clases linealmente separables, entonces el algoritmo del perceptrón converge y toma como superficie de decisión un hiperplano entre estas dos clases. La prueba de convergencia del algoritmo es conocida como el teorema de convergencia del perceptrón.

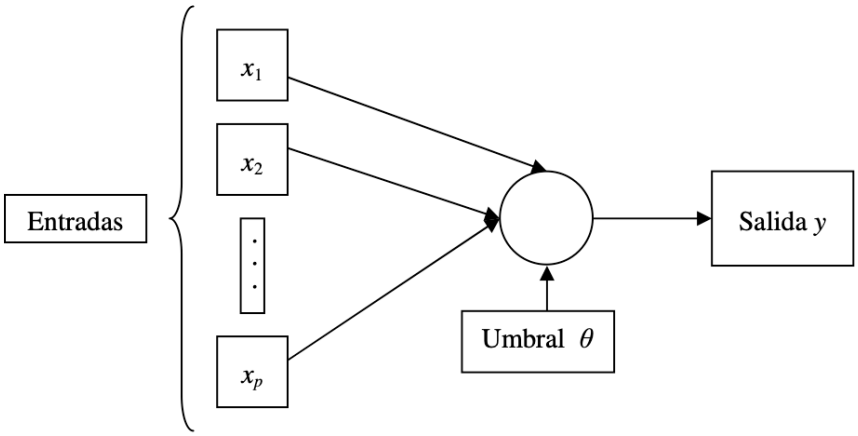


Figura 3.3: Diagrama de un Perceptron con cinco señales de entrada.

El perceptrón de una capa descrito en la Figura 3.3 tiene sólo una neurona. Si se expande la capa de salida para incluir más que una neurona, se podría realizar la clasificación en más de dos clases, pero estas tendrían que ser linealmente separables.

Dados los pesos sinápticos $w_1, w_2, ..., w_j$ y las entradas $x_1, x_2, ..., x_j$ con un bias o umbral θ, la salida del combinador lineal sería $y = \sum_{i=1}^{j} w_i x_i - \theta$. Si la salida fuese > 0 entonces pertenecería a una clase y si fuese =0 pertenecería a la otra. Diferenciamos dos tipos de aprendizaje para el perceptrón, la diferencia radica en la inclusión de un parámetro llamado tasa de aprendizaje que amortigua el cambio de los valores de peso. Primero definimos:

- $x(j)$ denota el elemento en la posición j en el vector de la entrada
- $w(j)$ el elemento en la posición j en el vector de peso
- y denota la salida de la neurona
- δ denota la salida esperada
- α es una constante tal que $0 < \alpha < 1$

En función del tipo de aprendizaje escogido utilizaremos una u otra regla de actualización de pesos, utilizando la tasa de aprendizaje:

$w(j)' = w(j) + \alpha(\delta - y)x(j)$

sin utilizar tasa de aprendizaje:

$w(j)' = w(j) + (\delta - y)x(j)$

se modela el aprendizaje como la actualización del vector de pesos después de cada iteración, esto sólo tendrá lugar si la salida y difiere de la salida deseada δ. Para considerar una neurona al interactuar en múltiples iteraciones debemos definir algunas variables más:

- x_i denota el vector de entrada para la iteración i
- w_i denota el vector de peso para la iteración i
- y_i denota la salida para la iteración i
- $D_m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ denota un periodo de aprendizaje de m iteraciones

En cada iteración el vector de peso es actualizado como sigue:

- Para cada pareja ordenada (x, y) en $D_m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
- Pasar (x_i, y_i, w_i) a la regla de actualización $w(j)' = w(j) + \alpha(\delta - y)x(j)$

El periodo de aprendizaje D_m se dice que es separable linealmente si existe un valor positivo γ y un vector de peso w tal que: $y_i \cdot (\langle w, x_i \rangle + u) > \gamma$ para todos los i. Novikoff (1962) [12] probó que el algoritmo de aprendizaje converge después de un número finito de iteraciones si los datos son separables linealmente y el número de errores está limitado a: $\left(\frac{2R}{\gamma}\right)^2$. Sin embargo si los datos no son separables linealmente, la línea de algoritmo anterior no se garantiza que converja [13].

3.5 Perceptrón multicapa

Ante la limitación del perceptrón simple de sólo poder computar funciones linealmente separables, surge el perceptrón multicapa, en el que se incorporan capas de neuronas ocultas con el objetivo de representar funciones no lineales. Minsky y Papert [14] mostraron en 1969 que de la combinación de varios perceptrones simples (inclusión de neuronas ocultas) podía resultar una solución adecuada para tratar ciertos problemas no lineales. El problema era que la regla de aprendizaje del perceptrón no podía aplicarse en este caso, y Minsky y Papert no aportaron un solución. En 1986 Rummelhart, Hinton y Wilians [15] presentaron la regla delta generalizada (Backpropagation), fue creada para generalizar la regla delta del perceptrón y aplicarla en redes de neuronas con múltiples capas.

El perceptrón multicapa está formado por una capa de entrada, una capa de salida y N capas ocultas intermedias. Las neuronas encargadas del procesamiento no lineal de los patrones

de los datos recibidos son las neuronas de las capas intermedias. Las neuronas de la capa de entrada únicamente propagan el input que reciben y las de la capa de salida proporcionan la respuesta de la red para cada uno de los patrones de entrada. Las conexiones del perceptrón multicapa siempre están dirigidas hacia adelante (las neuronas de una capa se conectan con las neuronas de la siguiente capa), de ahí que reciban también el nombre de redes alimentadas hacia adelante o redes feedforward. La arquitectura del perceptrón multicapa suele ser totalmente conectada (todas las neuronas de una capa están conectadas a todas las neuronas de la siguiente capa). Aunque hay ocasiones en que puede haber conexiones a mayores o eliminadas entre capas, y no es posible demostrar que se obtengan mejores o peores resultados sistemáticamente por este motivo. Al tener la característica de capas completamente conectadas tiene la desventaja de que el número total de parámetros crece de manera muy rápida creando redundancia e ineficiencia, otra desventaja es que no hace uso de la información espacial.

3.5.1 Fase de propagación

El perceptrón multicapa define una relación entre las variables de entrada y las variables de salida de la red. Esta relación se obtiene propagando hacia adelante los valores de las variables de entrada. Para ello, cada neurona de la red procesa la información recibida por sus entradas y produce una respuesta o activación que se propaga, a través de las conexiones correspondientes, hacia las neuronas de la siguiente capa. Las neuronas de la capa de entrada simplemente propagan el valor de entrada de la red. La activación de las neuronas de las capas intermedias y de la capa de salida viene dada por la función de activación f aplicada a la suma de los productos de las entradas que recibe por sus correspondientes pesos. La función f es la llamada función de activación. Para el perceptrón multicapa, las funciones de activación más utilizadas son la función sigmoidal y la función tangente hiperbólica. Dichas funciones poseen como imagen un intervalo continuo de valores dentro de los intervalos [0,1] y [−1,1], respectivamente, y vienen dadas por las siguientes ecuaciones:

$$f_{sigm}(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

$$f_{thip}(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{3.2}$$

Ambas son funciones crecientes con dos niveles de saturación: el máximo, que proporciona salida 1, y el mínimo, salida 0 para la función sigmoidal y salida -1, para la tangente hiperbólica, como se observa en la Figura 3.4.

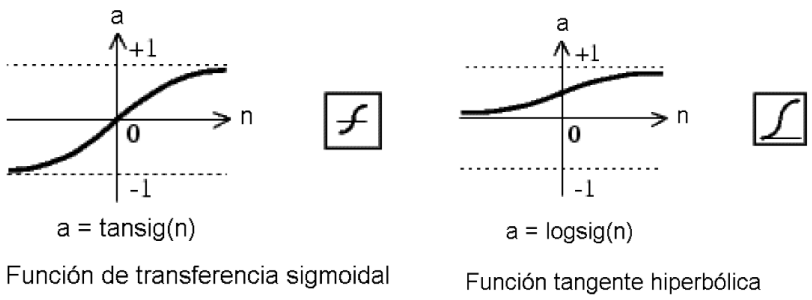


Figura 3.4: Funciones de activación sigmoidal y tangente hiperbólica.

3.5.2 Fase de aprendizaje

Los errores obtenidos a la salida del perceptrón se van propagando hacia atrás (retropropagación) con el objetivo de modificar los pesos de las conexiones para que el valor estimado de la red se asemeje cada vez más al real, esta aproximación se realiza mediante la función gradiente del error. El aprendizaje de la red se formula como un problema de minimización del siguiente modo:

$$Min_W E \tag{3.3}$$

siendo W el conjunto de parámetros de la red (pesos y umbrales) y E una función error que evalúa la diferencia entre las salidas de la red y las salidas deseadas. En la mayor parte de los casos, la función error se define como:

$$E = \frac{1}{N}\sum_{n=1}^{N} e(n) \tag{3.4}$$

donde N es el número de patrones o muestras y $e(n)$ es el error cometido por la red para el patrón n, dado por:

$$e(n) = \frac{1}{n_C}\sum_{i=1}^{n_C}(s_i(n) - y_i(n))^2 \tag{3.5}$$

siendo $Y(n) = (y_1(n), ..., y_{n_C}(n))$ y $S(n) = (s_1(n), ..., s_{n_C}(n))$ los vectores de salidas de la red y salidas deseadas para el patrón, respectivamente.

De este modo, si W^* es un mínimo de la función error E, en dicho punto el error es próximo a cero, lo cual implica que la salida de la red es próxima a la salida deseada, alcanzando así la meta de la regla de aprendizaje.

Por tanto, el aprendizaje del perceptrón multicapa es equivalente a encontrar un mínimo de la función error. La presencia de funciones de activación no lineales hace que la respuesta de la red sea no lineal respecto a los parámetros ajustables, por lo que el problema de minimización es un problema no lineal, y, como consecuencia, tienen que utilizarse técnicas de

optimazación no lineales para su resolución. Dichas técnicas están, generalmente, basadas en una adaptación de los parámetros siguiendo una cierta dirección de búsqueda. En el contexto de redes de neuronas, y en particular para el perceptrón multicapa, la dirección de búsqueda más comúnmente usada es la dirección negativa del gradiente de la función E (método de descenso del gradiente), pues conforme al cálculo de varias variables, esta es la dirección en la que la función decrece.

Aunque, estrictamente hablando, el aprendizaje de la red debe realizarse para minimizar el error total, el procedimiento más utilizado está basado en métodos del gradiente estocástico, los cuales consisten en una sucesiva minimización de los errores para cada patrón, $e(n)$, en lugar de minimizar el error total E. Por tanto, aplicando el método de descenso del gradiente estocástico, cada parámetro w de la red se modifica para cada patrón de entrada n de acuerdo con la siguiente ley de aprendizaje.

$$w(n) = w(n-1) - \alpha \frac{\partial e(n)}{\partial w} \tag{3.6}$$

donde $e(n)$ es el error para el patrón n dado por la ecuación 3.5, y α es la tasa de aprendizaje, parámetro que influye en la magnitud del desplazamiento en la superficie del error.

Debido a que las neuronas de la red están agrupadas en capas de distintos niveles, es posible aplicar el método del gradiente de forma eficiente, resultando el conocido algoritmo de retropropagación o regla delta generalizada. El término de retropapagación se utiliza debido a la forma de implementar el método del gradiente en el perceptrón multicapa, pues el error cometido en la salida de la red es propagado hacia atrás, transformándolo en un error para cada una de las neuronas ocultas de la red.

3.5.3 Funcionamiento

- Se pasa un vector de entrada a la red y se calculan las salidas.
- Se determina una medida de error.
- Se determina la dirección en la que cambiar los pesos buscando minimizar el error.
- Se determina la cantidad precisa en la que cambian los pesos.
- Se cambian los pesos
- Se repiten los pasos anteriores para todos los patrones de entrenamiento hasta que el error del conjunto de vectores de entrenamiento quede reducido a un valor aceptable.

3.6 CNNs

Aunque dentro del deep learning existen otro tipo de redes de neuronas, tales como las redes de neuronas recurrentes (Recurrente Neural Networks, RNNs), en este trabajo nos centraremos en las Convolutional Neural Networks (CNN), en español redes neuronales convolucionales, son una variación de un perceptrón multicapa y su aplicación se realiza en matrices bidimensionales, resultando muy efectivas para las tareas de visión artificial, como por ejemplo la clasificación y segmentación de imágenes.

Los fundamentos de las redes neuronales convolucionales se basan en el Neocognitron, introducido por Kunihiko Fukushima en 1980 [16]. Este modelo fue más tarde mejorado por Yann LeCun [17] en 1998 al introducir un método de aprendizaje basado en la propagación hacia atrás para poder entrenar el sistema correctamente. En el año 2012, fueron refinadas por Dan Ciresan [18], e implementadas para una unidad de procesamiento gráfico (GPU) consiguiendo así resultados impresionantes.

Una CNN es una red a la que se le pasa una imagen como entrada, le asigna una importancia (aprendida en función de pesos y sesgos) a varios aspectos u objetos en la imagen y es capaz de diferenciar unos de otros. El preprocesado requerido en una CNN es mucho menor comparado con otros algoritmos de clasificación. Además, es capaz de capturar tanto las dependencias espaciales como las temporales de una imagen, a través de la aplicación de filtros relevantes. El objetivo principal de las CNN es reducir las imágenes de una forma en la que sea más fácil procesarlas sin perder características, que son críticas a la hora de obtener una buena predicción. Esto es importante cuando debemos diseñar una arquitectura que no sea solamente buena para aprender características, sino que sea también escalable a conjuntos de datos masivos.

Su topología está dividida en múltiples etapas de aprendizaje compuestas por una combinación de: capas convolucionales, unidades de procesamiento no lineal (Relu) y capas de submuestreo. La operación de convolución ayuda en la extracción de características útiles de puntos de datos correlacionados localmente. La salida de la convolución se asigna luego a la unidad de procesamiento no lineal (función de activación), que no solo ayuda a aprender abstracciones sino que también integra la no linealidad en el espacio de características. Esta no linealidad genera diferentes patrones de activación para diferentes respuestas y, por lo tanto, facilita el aprendizaje de las diferencias semánticas en las imágenes. La salida de la función de activación no lineal suele ir seguida de un submuestreo, que ayuda a resumir los resultados y también hace que la entrada sea invariable a las distorsiones geométricas. No necesita de un extractor de carácterísticas por separado, al poseer esta capacidad de forma automática, por lo que una CNN no necesita un procesamiento exhaustivo para aprender buenas representaciones a partir de píxeles sin procesar.

Durante el entrenamiento, la CNN aprende a través del algoritmo de retropropagación, regulando el cambio de peso según el objetivo. La optimización de una función objetivo utilizando un algoritmo de retropropagación es similar al aprendizaje basado en respuestas del cerebro humano. La estructura jerárquica de múltiples capas profundas de la CNN le da la capacidad de extraer características de nivel bajo, medio y alto. Las características de alto nivel (características más abstractas) son una combinación de características de nivel medio y bajo. La capacidad de extracción de características jerárquicas de CNN emula el proceso de aprendizaje profundo y en capas del Neocórtex en el cerebro humano, que aprende características dinámicamente de los datos sin procesar. La popularidad de CNN se debe principalmente a su capacidad de extracción de características jerárquicas.

Las arquitecturas profundas a menudo tienen una ventaja sobre las arquitecturas superficiales cuando se trata de problemas de aprendizaje complejos. El apilamiento de múltiples unidades de procesamiento lineales y no lineales en forma de capas proporciona la capacidad de aprender representaciones complejas en diferentes niveles de abstracción. En consecuencia, en las tareas de reconocimiento que constan de cientos de categorías de imágenes, las CNN profundas han mostrado una mejora sustancial del rendimiento con respecto a los modelos convencionales.

Se ha demostrado que al darle suficientes datos de entrenamiento, las CNN profundas pueden aprender las representaciones invariantes y pueden lograr un rendimiento a nivel humano. Además de su uso como mecanismo de aprendizaje supervisado, el potencial de las CNN profundas se puede aprovechar para extraer representaciones útiles de una gran escala de datos sin etiquetar. También se ha demostrado que diferentes niveles de características, tanto de bajo como de alto nivel, pueden transferirse a una tarea de reconocimiento genérico explotando el concepto de aprendizaje por transferencia [19], [20].

Desde finales de la década de 1990 hasta el 2000, se realizaron varias mejoras en la metodología y arquitectura de aprendizaje de las CNN para hacerlas más escalables a problemas grandes, heterogéneos, complejos y multiclase. Las innovaciones en las CNN incluyen diferentes aspectos como la modificación de unidades de procesamiento, estrategias de optimización de parámetros e hiperparámetros, patrones de diseño y conectividad de capas, etc. Las aplicaciones basadas en las CNN se volvieron frecuentes después del increíble desempeño de AlexNet en el conjunto de datos ImageNet en 2012 [21]. Desde entonces se han propuesto importantes innovaciones en CNN y se atribuyen en gran medida a la reestructuración de las unidades de procesamiento y al diseño de nuevos bloques.

3.6.1 Componentes básicos CNNs

Capa convolucional

La capa convolucional está compuesta por un conjunto de kernels o filtros convolucionales que realizan operaciones de convolución mientras escanean la entrada (imagen) con respecto a sus dimensiones. Sus hiperparámetros incluyen el tamaño de filtro y la zancada (stride). La salida resultante se denomina mapa de características o mapa de activación. El kernel contiene un conjunto específico de pesos que multiplica con los elementos correspondientes del campo receptivo. La operación de convolución se puede expresar de la siguiente manera:

$$f_l^k(p,q) = \sum_c \sum_{x,y} i_c(x,y) \cdot e_l^k(u,v) \tag{3.7}$$

donde, $i_c(x,y)$ es un elemento del tensor de entrada de la imagen I_c, este es multiplicado por el índice $e_l^k(u,v)$ del k-ésimo kernel convolucional k_l de la l-ésima capa. Mientras que el mapa de características de salida de la k-ésima operación convolucional se puede expresar como $F_l^k = \big[f_l^k(1,1), ..., f_l^k(p,q), ..., f_l^k(P,Q)\big]$.

Debido a la capacidad de compartir pesos de la operación convolucional, se pueden extraer, diferentes conjuntos de características dentro una imagen, deslizando el kernel con el mismo conjunto de pesos en la imagen y, por lo tanto, hace que el parámetro CNN sea eficiente en comparación con las redes completamente conectadas. La operación de convolución puede clasificarse según el tipo y tamaño de los filtros, el tipo de relleno y la dirección de la convolución.

Capa de agrupamiento

Una vez que se extraen las características, su ubicación exacta se vuelve menos importante siempre que se conserve su posición aproximada con respecto a otras características. La agrupación o el submuestreo es una operación local que resume información similar en la vecindad del campo receptivo y genera la respuesta dominante dentro de esta región local.

$$Z_l^k = g_p(F_l^k) \tag{3.8}$$

La ecuación 3.8 muestra la operación de agrupamiento, donde Z_l^k representa el agrupamiento del mapa de características en la l-ésima capa para la k-ésima entrada del mapa de características F_l^k. $g_p(\cdot)$ define el tipo de operación de agrupamiento.

El uso de la operación de agrupamiento ayuda a extraer una combinación de características, que son invariantes a los cambios de traslación y pequeñas distorsiones [22], [23]. La reducción del tamaño del mapa de características al conjunto de características invariantes

no solo regula la complejidad de la red, sino que también ayuda a aumentar la generalización al reducir el sobreajuste. Se utilizan diferentes tipos agrupamiento, tales como máximo, promedio, L2, superposición, agrupación de pirámides espaciales, etc.

Función de activación

La función de activación sirve como función de decisión y ayuda a aprender patrones intrincados. La selección de una función de activación adecuada puede acelerar el proceso de aprendizaje. La función de activación para un mapa de características convolucionado se define:

$$T_l^k = g_a(F_l^k) \tag{3.9}$$

La ecuación 3.9 muestra que F_l^k es una salida de una convolución, que está asignada a la función de activación $g_a(\cdot)$. Esta función añade la no linealidad y devuelve una salida transformada F_l^k para la l-ésima capa. En la literatura, se utilizan diferentes funciones de activación como sigmoide, tanh, maxout, SWISH, ReLU y variantes de ReLU, como ReLU con fugas, ELU y PReLU. Sin embargo, se prefieren ReLU y sus variantes, ya que ayudan a superar el problema del gradiente de desaparición [24], [25]. Una de las funciones de activación propuestas recientemente es MISH, que ha mostrado un mejor rendimiento que ReLU en la mayoría de las redes profundas propuestas recientemente en conjuntos de datos de referencia[26].

Batch normalization

Ioffe y Szegedy [27] observaron que el cambio de la distribución de las activaciones de las redes debido al cambio de los hiperparámetros durante el entramiento, provocaba un mayor tiempo de entrenamiento y llamaron a esto cambio de covarianza interna. Para abordar los problemas relacionados con el cambio de covarianza interno dentro de los mapas de características, introdujeron batch normalization, que unifica la distribución de los valores del mapa de características al establecerlos en media cero y varianza unitaria. Además, suaviza el flujo de gradiente y actúa como factor regulador, lo que ayuda a mejorar la generalización de la red. El cambio de covarianza interno es un cambio en la distribución de los valores de las unidades ocultas, que ralentiza la convergencia (al forzar la tasa de aprendizaje a un valor pequeño) y requiere de una cuidadosa inicialización de parámetros. Se muestra batch normalization para un mapa de características transformado F_l^k en la ecuación 3.10.

$$N_l^k = \frac{F_l^k - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{3.10}$$

En la ecuación 3.10, N_l^k representa el mapa de características normalizado, F_l^k es el mapa de características de entrada, μ_B y σ_B^2 corresponden a media y varianza del mapa de características para un minibatch. Se añade ϵ para evitar la división entre cero.

Ioffe y Zegedy observaron en sus experimentos que la utilización de batch normalization, a parte de acortar el tiempo de convergencia de la red, también reducía la necesidad de dropout.

Dropout

El dropout introduce la regularización dentro de la red, que en última instancia mejora la generalización al omitir aleatoriamente algunas unidades o conexiones con una cierta probabilidad. En las CNNs, las conexiones múltiples que aprenden una relación no lineal a veces se coadaptan, lo que provoca un sobreajuste. Este dropout aleatorio de algunas conexiones o unidades produce varias arquitecturas de red adelgazadas y, finalmente, se selecciona una red representativa con pesos pequeños. Esta arquitectura seleccionada se considera luego como una aproximación de todas las redes propuestas.

Capa completamente conectada

La capa completamente conectada se usa principalmente al final de la red para la clasificación. A diferencia de la agrupación y la convolución, es una operación global. Toma información de las etapas de extracción de características y analiza globalmente la salida de todas las capas anteriores. En consecuencia, realiza una combinación no lineal de características seleccionadas, que se utilizan para la clasificación de datos.

Función de pérdida

La función de pérdida o también llamada función objetivo. Normalmente en las CNN se trata de minimizar el valor de esta función, ya que evalúa la diferencia entre los valores de salida de la red de los valores verdaderos. En función del tipo de problema al que nos enfrentemos utilizaremos distintos tipo de funciones de pérdida. Por lo que es la encargada de gestionar los ajustes de pesos en toda la red. Antes de que comience el entrenamiento de la red, los pesos en las capas de convolución y completamente conectadas reciben valores aleatorios. Luego, durante el entrenamiento, la capa de pérdida verifica continuamente las suposiciones de la capa completamente conectada con los valores reales con el objetivo de minimizar la diferencia entre la suposición y el valor real tanto como sea posible. La capa de pérdida hace esto ajustando los pesos tanto en la convolución como en las capas completamente conectadas.

Métodos de optimización

Los optimizadores vinculan la función de pérdida y los parámetros del modelo actualizando el modelo en respuesta a la salida de la función de pérdida. Es decir, los optimizadores dan forma y moldean el modelo para que sean lo más precisos posibles al combinar los pesos. La función de pérdida indica al optimizador cuándo se está moviendo en la dirección correcta o incorrecta.

- **Algoritmo SGD:** El optimizador de descenso de gradiente estocástico, SGD [28], también conocido como descenso de gradiente incremental, es un procedimiento iterativo para optimizar una función objetivo diferencial, una aproximación estocástica del descenso de gradiente. El algoritmo estocástico no necesita recordar qué ejemplos fueron vistos durante las iteraciones previas, por lo tanto, puede procesar ejemplos durante la etapa de entrenamiento.

- **Algoritmo SGDM:** El optimizador de descenso de gradiente estocástico con impulso, SGDM [29] , es una variación del descenso de gradiente estocástico utilizado para una convergencia más rápida de la función de pérdida. Utiliza una tasa de aprendizaje única para parámetros completos. Esto proporciona al descenso del gradiente un impulso para evitar quedar atrapado en el mínimo local. Sin embargo, el uso de un impulso (demasiado) grande junto con una tasa de aprendizaje también grande puede producir un salto de una solución adecuada (mínimo local) debido a un gran paso de actualización.

- **Algoritmo RMSProp:** El optimizador de propagación de la raíz cuadrada media, RMSProp [30], utiliza diferentes tasas de aprendizaje para cada parámetro. Las tasas de aprendizaje pueden adaptarse automáticamente a la función de pérdida que se optimiza con el objetivo de mejorar el entrenamiento de la red. La idea central de RMSprop es mantener el promedio móvil de los gradientes al cuadrado para cada peso, y luego dividir el gradiente por la raíz cuadrada del cuadrado medio. Por eso se llama RMSprop (raíz cuadrada media).

- **Algoritmo Adam:** El algoritmo Adam [31], es un algoritmo para la optimización basada en gradientes de primer orden de funciones objetivas estocásticas, basado en estimaciones adaptativas de momentos de orden inferior. El método es sencillo de implementar, es computacionalmente eficiente, tiene pocos requisitos de memoria, es invariante al cambio de escala diagonal de los gradientes y es muy adecuado para problemas que son grandes en términos de datos y / o parámetros. El método estima las tasas de aprendizaje adaptativo individuales para diversos parámetros a partir de cálculos del primer y segundo momento de los gradientes. Algunos de los beneficios de Adam son que la cantidad de actualizaciones de parámetros no varía con el cambio de escala del gradiente,

el tamaño de sus pasos está casi limitado por el hiperparámetro de tamaño de stride.

3.6.2 Estrategias de aprendizaje

Aprendizaje desde cero

Todos los parámetros de la CNN se inicializan de manera aleatoria siguiendo distribuciones gaussianas aleatorias.

Estrategias de aprendizaje por transferencia

El avance del deep learning en los últimos años ha posibilitado un gran progreso en muchos campos, al poder abordar problemas complejos y obtener muy buenos resultados, pero el tiempo de formación y la cantidad de datos necesarios para este tipo de sistemas es mucho mayor que para los sistemas de aprendizaje automático tradicionales. Este problema se trata de solventar al compartir los detalles de las redes más usadas de deep learning entrenadas para conjuntos de datos y que de esta forma otras personas puedan usarlos para entrenar sus modelos.

- **Modelos pre-entrenados listos para usar como extractores de funciones**
  La utilización de modelos pre-entrenados como extractores de funciones consiste en utilizar los modelos sin su última capa, en donde esta puede ser sustituida por otra que se ajuste más al problema a abordar. Esto es posible ya que los sistemas de deep learning están formados por capas que aprenden características de forma jerárquica, donde a medida que avanza el modelo, las capas son más ricas semánticamente. Por tanto, se utiliza una red pre-entrenada para otro problema, pero que es útil para la extracción de características de bajo nivel, que después pueden refinarse para ajustarse al problema a abordar. Por ejemplo, si se utiliza AlexNet sin su capa de clasificación final, transformará imágenes de una nueva tarea de dominio en un vector de 4096 dimensiones basado en sus estados ocultos, lo que permitirá extraer características de una nueva tarea de dominio, utilizando el conocimiento de una tarea de dominio de origen. Este es uno de los métodos más utilizados para realizar el aprendizaje por transferencia utilizando redes neuronales profundas.

- **Ajuste fino de modelos pre-entrenados listos para usar**
  Esta es una técnica más complicada, en la que no sólo se reemplaza la capa final (para clasificación / regresión), sino que también se re-entrena selectivamente algunas de las capas anteriores. Las redes neuronales profundas son arquitecturas altamente configurables con varios hiperparámetros. Las capas iniciales capturan características genéricas, mientras que las últimas se enfocan más en la tarea específica en cuestión. Usando

esta información, se pueden congelar (fijar pesos) ciertas capas mientras se re-entrenan, o ajustar el resto de ellas para satisfacer nuestras necesidades. En este caso, se utiliza el conocimiento en términos de la arquitectura general de la red y se usan sus estados como punto de partida para el paso de re-entrenamiento. Esto, a su vez, ayuda a lograr un mejor rendimiento con menos tiempo de entrenamiento.

3.6.3 Arquitecturas

Desde que surgieron en 1989 se han hecho incontables mejoras a su arquitectura. Estas mejoras se pueden categorizar en optimización de parámetros, regularización, reformulación estructural, etc. Sin embargo, la principal mejora en el desempeño de las CNN provino de la reestructuración de unidades de procesamiento. La mayoría de las innovaciones en las arquitecturas de las CNN se han realizado en relación con la profundidad y la explotación espacial. Aunque no profundizaremos en ellas, dependiendo del tipo de modificaciones arquitectónicas, las CNNs se pueden clasificar en siete clases diferentes como se ve en la Figura 3.5.

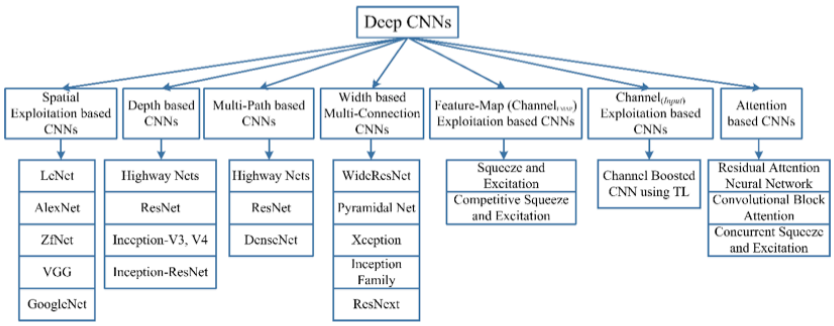


Figura 3.5: Taxonomía de las arquitecturas CNN profundas.

3.7 CNNs en la detección de nódulos pulmonares

El problema que se aborda en este proyecto consiste en detectar nódulos en TC torácicas. Esto se puede hacer de tres formas, utilizando información 2D, 2,5D o 3D. La diferencia radica en que si se sigue la aproximación 2D hay que detectar los nódulos en cortes bidimensionales del TC, 2,5D incorpora información de los cortes adyacentes,y 3D realiza un corte del volumen 3D en forma de cubo (procesar el volumen completo es muy costoso a nivel computacional). Todo esto para detectar un objeto (nódulo) dentro del corte, lo que se hará con redes para la detección de objetos en imágenes, que recibirán la posición exacta del nódulo dentro de ese corte para usarla como verdad fundamental a la hora de entrenar el sistema, esta posición se

obtiene de anotaciones en formato xml proporcionadas por la base de datos a utilizar.

3.7.1 R-CNN

R-CNN, "Region-Based Convolutional Network" [32] desarrollada en 2014 por un grupo de investigadores de la Universidad de Berkeley, es una red para la detección de objetos y su principal innovación fue la utilización de una CNN para la extracción de características.

Consta de tres módulos, el primero utiliza el algoritmo de búsqueda selectiva para la generación de propuestas de regiones, se generan sobre 2000 regiones. El segundo módulo (la CNN), extrae un vector de características de longitud 4096 de cada propuesta de región. El tercero utiliza un SVM pre-entrenado para clasificar entre fondo o una de las clases de objetos.

Las limitaciones de este modelo incluyen: la imposibilidad de entrenar de un extremo a otro, al estar formado de componentes independientes; requiere de una gran capacidad de memoria para el entrenamiento; lentitud, el algoritmo de búsqueda selectiva lleva mucho tiempo; cada propuesta de región se envía de forma independiente a la CNN para la extracción de características. Todo esto hace que sea imposible ejecutar R-CNN en tiempo real.

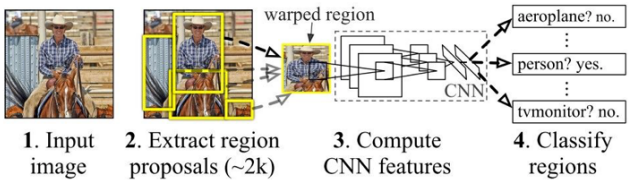


Figura 3.6: Funcionamiento R-CNN.

El funcionamiento de la R-CNN se puede ver en la Figura 3.6. Primero se pre-entrena una CNN en tareas de clasificación de imágenes. La clasificación de estas imágenes puede involucrar N clases. Una vez hecho esto obtenemos las propuestas ROI (Region Of Interest) usando el algoritmo de búsqueda selectiva. Estas ROI pueden contener objetos "target" (los objetos que buscamos detectar) y son de diferentes tamaños. Los candidatos ROI se deforman para tener el tamaño fijo requerido por la CNN. Se ajusta la CNN en ROI deformadas para K + 1 clases, la clase adicional se refiere al fondo. En la etapa de ajuste fino, se debería usar una tasa de aprendizaje mucho menor y el mini-batch sobremuestrea los casos positivos porque la mayoría de las regiones propuestas son solo fondo. Dada cada región de la imagen, una propagación hacia adelante a través de la CNN genera un vector de características. Este vector de características es consumido por una SVM binaria entrenada para cada clase de forma independiente. Las muestras positivas son regiones propuestas con umbral de superposición de IoU (intersección sobre unión)> = 0,3, y las muestras negativas son otras irrelevantes. Para reducir los errores de localización, se entrena un modelo de regresión para corregir la ventana

de detección predicha en el desplazamiento de corrección del bounding-box (cuadro delimitador) usando características CNN.

Regresión del bounding-box

Dada una coordenada predicha del bounding-box $P = (P_x, P_y, P_w, pP_h)$ (coordenada central, ancho, alto) y las correspondientes coordenadas verdaderas del bounding-box $G = (G_x, G_y, G_w, G_h)$, el regresor está configurado para aprender la transformación invariante de escala entre dos centros y la transformación de escala logarítmica entre anchos y alturas. Los objetivos de la regresión para los pares de entrenamiento (P,G) son:

$$t_x = \frac{(G_x - G_x)}{G_w} \tag{3.11}$$

$$t_y = \frac{(G_y - P_y)}{P_h} \tag{3.12}$$

$$t_w = \log\left(\frac{G_w}{P_w}\right) \tag{3.13}$$

$$t_h = \log\left(\frac{G_h}{P_h}\right) \tag{3.14}$$

En la siguientes ecuaciones aparece un nuevo término, $d(P)$ que es igual a producto escalar (w, P)

$$\widehat{G_x} = G_w d_x(P) + G_x \tag{3.15}$$

$$\widehat{G_y} = G_h d_y(P) + G_y \tag{3.16}$$

$$\widehat{G_w} = G_w \exp(d_w(P)) \tag{3.17}$$

$$\widehat{G_h} = G_h \exp(d_h(P)) \tag{3.18}$$

Entonces, la ecuación de regresión para encontrar W será

$$W_* = argmin_{\widehat{w}_*} \sum_{i}^{N} (t_*^i - \widehat{w}_*^T \phi_5(P^i))^2 + \lambda ||\widehat{w}_*|| \tag{3.19}$$

El término de regularización es fundamental aquí y el documento RCNN eligió el mejor λ mediante validación cruzada. No todos los bounding-box predichos tienen verdades fundamentales. Por ejemplo, si no hay superposición, no tiene sentido ejecutar la regresión bbox. En el artículo, solo se mantiene un bounding-box predicho con una verdad fundamental cercana con al menos 0.6 IoU para entrenar el modelo de regresión bbox.

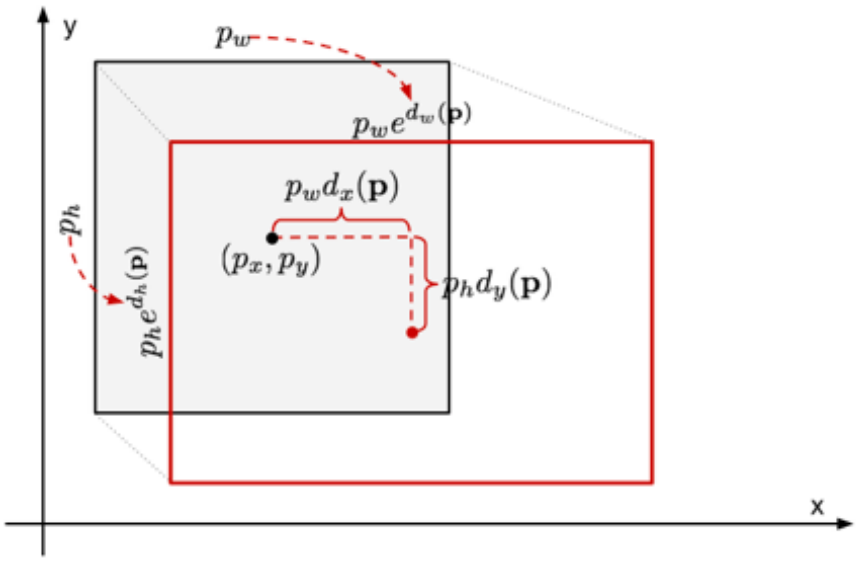


Figura 3.7: Transformación entre bounding-box predicho y verdadero.

Supresión no máxima

Es probable que el modelo pueda encontrar varios bounding-box para el mismo objeto. La supresión no máxima ayuda a evitar la detección repetida de la misma instancia. NMS es un algoritmo que recorre todas las clases y, para cada clase, comprueba si hay superposiciones (IoU - Intersección sobre Unión) entre todos los bounding-box. Si el IoU entre dos casillas de la misma clase está por encima de un cierto umbral (generalmente 0,7), el algoritmo concluye que se refieren al mismo objeto y descarta la casilla con la puntuación de confianza más baja (que es un producto de la puntuación de objetividad y la probabilidad de clase condicional).

Minería dura negativa

Se consideran los bounding-box sin objetos como ejemplos negativos. No todos los ejemplos negativos son igualmente difíciles de identificar. Por ejemplo, si tiene un fondo vacío puro, es probable que sea un ”negativo fácil”; pero si el bbox contiene una textura extraña y ruidosa o un objeto parcial, podría ser difícil de reconocer y estos son “negativos duros”.

Los ejemplos negativos difíciles se clasifican de manera errónea con facilidad. Podemos encontrar explícitamente esas muestras falsas positivas durante los ciclos de entrenamiento e incluirlas en los datos de entrenamiento para mejorar el clasificador.

3.7.2 Fast R-CNN

Fast R-CNN [33], es un detector de objetos que fue desarrollado únicamente por Ross Girshick, un investigador de inteligencia artificial de Facebook y ex investigador de Microsoft.

Fast R-CNN supera varias limitaciones de R-CNN. Como sugiere su nombre, una ventaja de Fast R-CNN sobre R-CNN es su velocidad.

Propone una nueva capa llamada "ROI Pooling", que es una capa de agrupación de ROI, que extrae vectores de características de igual longitud de todas las propuestas (ROIs) en la misma imagen.

En comparación con R-CNN, que tiene múltiples etapas (generación de propuestas de región, extracción de características y clasificación mediante SVM), Fast R-CNN construye una red que solo tiene una etapa.

Fast R-CNN comparte cálculos (es decir, cálculos de capa convolucional) en todas las propuestas (ROI) en lugar de hacerlos para cada propuesta de forma independiente. Esto se hace mediante el uso de la nueva capa de agrupación de ROI, que hace que Fast R-CNN además de ser más rápida y precisa que R-CNN, sea menos costosa a nivel de almacenamiento.

La arquitectura general de Fast R-CNN se muestra en la Figura 3.8. El modelo consta de una sola etapa, en comparación con las 3 etapas en R-CNN. Simplemente acepta una imagen como entrada y devuelve las probabilidades de clase y los bounding-box de los objetos detectados.

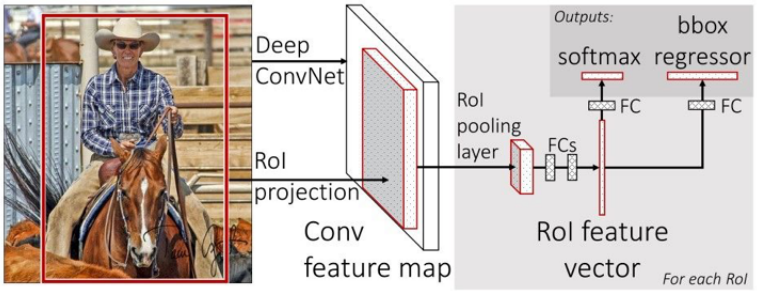


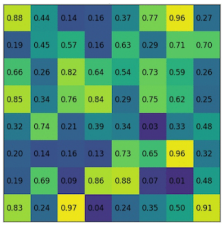
Figura 3.8: Funcionamiento Fast R-CNN.

El mapa de características de la última capa convolucional sirve de entrada a una capa de agrupación de ROI. Esto se hace para extraer de cada ROI un vector de características de una longitud fija. La razón es extraer un vector de características de longitud fija de cada propuesta de región. La Figura 3.9 muestra el funcionamiento de la capa de agrupación de ROI.

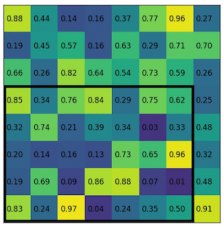
En pocas palabras, la capa de agrupación de ROI funciona dividiendo cada propuesta de región en una cuadrícula de celdas. La operación de agrupación máxima se aplica a cada celda de la cuadrícula para devolver un valor único. Todos los valores de todas las celdas representan el vector de características. Si el tamaño de la cuadrícula es 2×2, entonces la longitud del vector de características es 4.

El vector de características extraído pasa por capas completamente conectadas hasta que llega a la última capa, cuya salida se separa en 2 ramas:

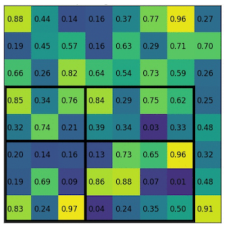
- Capa Softmax para predecir las puntuaciones de la clase.



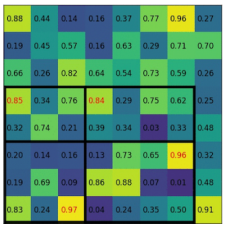
(a) Entrada



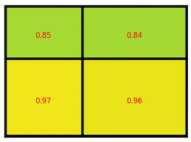
(b) Propuesta de región



(c) Secciones de agrupamiento



(d) Valores máximos de lax secciones



(e) Salida

Figura 3.9: Funcionamiento capa de agrupación ROI.

- Capa FC para predecir los bounding-box de los objetos detectados.

La mayor limitación de Fast R-CNN es el algoritmo de búsqueda selectiva, este además de ser lento también puede ser impreciso a la hora de detectar todos los objetos ”target”, ya que el método no se puede personalizar para una tarea de detección específica.

Función de pérdida

Símbolo	Explicación
u	Etiqueta de verdad fundamental; $\mu \epsilon 0, 1, ..., K$; el fondo se denota con $\mu = 0$
p	La probabilidad de cada ROI por cada K+1 clases
v	Bounding-box verdadera $v = (v_x, v_y, v_w, v_h)$
t^u	Bounding-box predicha $t^u = (v_x^u, v_y^u, v_w^u, v_h^u)$

Tabla 3.1: Tabla sobre los valores que intervienen en la función de pérdida

El modelo se optimiza buscando la minimización de la suma de dos funciones de pérdida $\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box}$. La primera se obtiene de $\mathcal{L}_{cls}(p, u) = -\log p_u$ y la segunda es $\mathcal{L}_{box}(t^u, v) = -\sum_{i\epsilon\{x,y,w,h\}} L_1^{smooth}(t_i^u - v_i)$ la función de pérdida suavizada L1 (”smooth L1 loss”) que supuestamente es menos sensible a los ”outliers”. Esta última sólo se tiene encuenta si la ROI no es fondo ($u \geq 1$). Podemos ver a qué corresponde cada parámetro en la Tabla 3.1.

3.7.3 Faster R-CNN

Faster R-CNN creada por Ren et al. [34] es una extensión de Fast RCNN, a la que supera en velocidad. Esta red de detección de objetos propone una CNN llamada RPN ("Region Proposal Network") para la generación de propuestas con varias escalas y proporciones. La RPN implementa lo que se conoce como CNN con atención (indica a la Fast R-CNN dónde buscar). En lugar de utilizar pirámides de imágenes (es decir, múltiples instancias de la imagen pero a diferentes escalas) o pirámides de filtros (es decir, múltiples filtros con diferentes tamaños), este documento introdujo el concepto de "anchor boxes". Un anchor box es una caja de referencia con una escala y relación de aspecto específicas. Con múltiples anchor boxes de referencia, existen múltiples escalas y relaciones de aspecto para una región. Esto se puede considerar como una pirámide de anchor boxes de referencia. Luego, cada región se asigna a cada anchor box de referencia y, por lo tanto, se detectan objetos a diferentes escalas y relaciones de aspecto. Los cálculos convolucionales se comparten entre la RPN y la Fast R-CNN, lo que reduce el tiempo de cálculo.

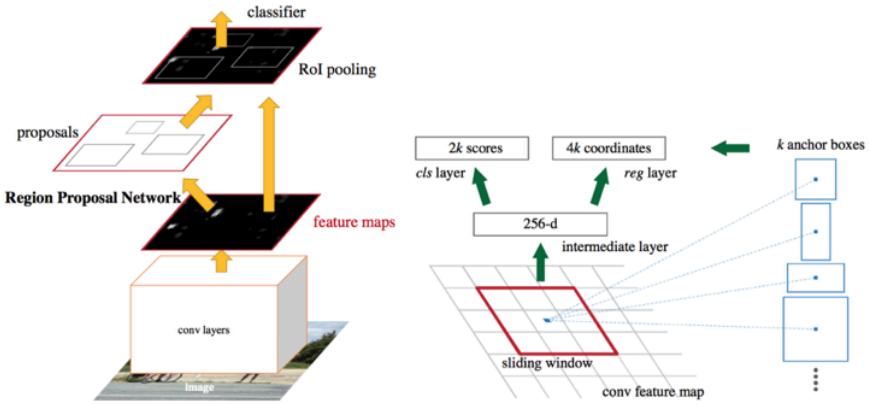


Figura 3.10: Arquitectura Faster R-CNN.

En la Figura 3.10 podemos ver su arquitectura. La RPN genera las propuestas ROI, para cada una de ellas utilizando la capa de agrupación, se extrae un vector de características de longitud fija. Estos vectores de características se clasifican usando Fast R-CNN. La salida son las puntuaciones de clase de los objetos junto a su bounding-box.

RPN

Los modelos R-CNN y Fast R-CNN dependen del algoritmo de búsqueda selectiva para generar propuestas de región. Cada propuesta se envía a una CNN previamente capacitada para su clasificación. Este artículo propuso una red denominada red de propuestas regionales (RPN) que puede producir las propuestas regionales. Esto tiene algunas ventajas:

- Las propuestas de región ahora se generan utilizando una red que puede entrenarse y personalizarse de acuerdo con la tarea de detección.
- Debido a que las propuestas se generan utilizando una red, esta se puede entrenar de un extremo a otro para personalizarla en la tarea de detección. Por lo tanto, produce mejores propuestas de región en comparación con métodos genéricos como Selective Search y EdgeBoxes.
- La RPN procesa la imagen utilizando las mismas capas convolucionales utilizadas en la red de detección Fast R-CNN. Por lo tanto, la RPN no necesita más tiempo para producir las propuestas en comparación con los algoritmos como la búsqueda selectiva.
- Debido a que comparten las mismas capas convolucionales, la RPN y el Fast R-CNN se pueden fusionar/unificar en una sola red. Por lo tanto, el entrenamiento se realiza solo una vez.

La RPN funciona en el mapa de características de salida devuelto desde la última capa convolucional compartida con Fast R-CNN. Sobre la base de una ventana rectangular de tamaño nxn, una ventana deslizante atraviesa el mapa de características. Para cada ventana, se generan varias propuestas de regiones candidatas. Estas propuestas no son las propuestas finales, ya que se filtrarán en función de su ”puntuación de objetividad”.

Anchor

El mapa de características de la última capa de convolución compartida se pasa a través de una ventana deslizante rectangular de tamaño nxn, donde n = 3 para la red VGG-16. Para cada ventana, se generan K propuestas de región. Cada propuesta se parametriza según una caja de referencia que se denomina anchor box. Los 2 parámetros de los anchor boxes son:

- Escala
- Relación de aspecto

Generalmente, hay 3 escalas y 3 relaciones de aspecto y, por lo tanto, hay un total de $K = 9$ anchor boxes. Pero K puede ser diferente de 9. En otras palabras, K regiones se producen a partir de cada propuesta de región, donde cada una de las K regiones varía en la escala o en la relación de aspecto.

Utilizando anchor boxes, se utiliza una sola imagen a una sola escala y, al mismo tiempo, se pueden ofrecer detectores de objetos invariantes en escala, ya que los anchors existen a diferentes escalas. Esto evita el uso de múltiples imágenes o filtros. Múltiples anchors son clave para compartir características en la RPN y la red de detección Fast R-CNN.

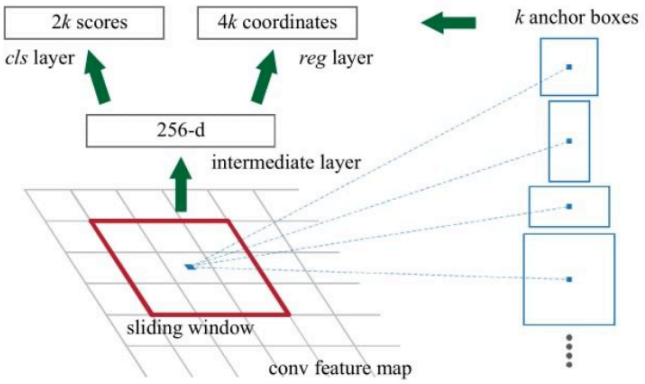


Figura 3.11: Anchor.

Para cada propuesta de región nxn, se extrae un vector de características (de longitud 256 para la red ZF y 512 para la red VGG-16). Este vector luego se alimenta a 2 capas hermanas completamente conectadas. La primera capa FC se llama cls y representa un clasificador binario que genera la puntuación de objetividad para cada propuesta de región (es decir, si la región contiene un objeto o es parte del fondo). La segunda capa FC se llama reg, que devuelve un vector 4-D que define el bounding-box de la región. La siguiente sección analiza cómo se asigna la puntuación de objetividad a cada anchor y cómo se utiliza para producir la etiqueta de clasificación.

Puntuación de objetividad

La capa cls genera un vector de 2 elementos para cada propuesta de región. Si el primer elemento es 1 y el segundo elemento es 0, la propuesta de región se clasifica como fondo. Si el segundo elemento es 1 y el primer elemento es 0, entonces la región representa un objeto.

Para entrenar la RPN, cada anchor recibe una puntuación de objetividad positiva o negativa basada en la intersección sobre unión (IoU).

El IoU es la relación entre el área de intersección entre el anchor box y el bounding-box de verdad fundamental y el área de unión de ambos. El IoU varía de 0,0 a 1,0. Cuando no hay intersección, el IoU es 0,0. A medida que las 2 cajas (bounding-box) se acercan, el IoU aumenta hasta llegar a 1,0 (cuando las 2 cajas son 100% idénticas).

Las siguientes 4 condiciones utilizan IoU para determinar si se asigna una puntuación de objetividad positiva o negativa a un anchor:

- Un anchor que tiene una superposición de IoU superior a 0,7 con cualquier bounding-box de verdad fundamental recibe una etiqueta de objetividad positiva.

- Si no hay un anchor con una superposición de IoU superior a 0,7, se asigna una etiqueta positiva a los anchors con la superposición de IoU más alta con un bounding-box de verdad fundamental.
- Se asigna una puntuación de objetividad negativa a un anchor cuando la superposición de IoU para todos los bounding-box de verdad fundamental es inferior a 0,3. Una puntuación de objetividad negativa significa que el anchor se clasifica como fondo.
- Los anchors que no son ni positivos ni negativos no contribuyen en el entrenamiento.

Función de pérdida

Símbolo	Explicación
p_i	Probabilidad predicha para que el anchor i se un objeto
p_i^*	Etiqueta binaria de verdad fundamental sobre si el anchor i es un objeto
t_i	Bounding-box predicho $v = (v_x, v_y, v_w, v_h)$
t_i^*	Bounding-box de verdad fundamental
N_{cls}	Término de normalización, en el artículo para un tamaño de mini-batch de 256
N_{box}	Término de normalización, en el artículo para el número de anchor de 2400
λ	Parámetro de balanceo, fijado a 10 en el artículo (para que tanto $\mathcal{L}_{cls}$ como $\mathcal{L}_{box}cls$ tengan el mismo peso aproximadamente)

Tabla 3.2: Tabla sobre los valores que intervienen en la función de pérdida

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box} \tag{3.20}$$

$$\mathcal{L}(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i \mathcal{L}_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{box}} \sum_i p_i^* \cdot L_1^{smooth}(t_i - t_i^*) \tag{3.21}$$

donde $\mathcal{L}_{cls}$ es la función de pérdida de clasificación en dos clases, ya que podemos traducir fácilmente una clasificación de varias clases en una clasificación binaria al predecir que una muestra es un objeto objetivo en lugar de no. L_1^{smooth} es la función de pérdida loss L1.

$$\mathcal{L}_{cls}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*) \log(1 - p_i) \tag{3.22}$$

Capítulo 4

Estado del Arte

A continuación se describe el funcionamiento de los sistemas CAD citando algunas de las técnicas actuales de deep learning para la detección de nódulos pulmonares.

4.1 Sistemas CAD

A finales de la década de 1950, con el nacimiento de las computadoras modernas, los investigadores de diversos campos comenzaron a explorar la posibilidad de construir sistemas de diagnóstico médico asistido por computadora (CAD, Computer-Aided Diagnosis). Estos primeros sistemas CAD utilizaron diagramas de flujo, comparación de patrones estadísticos, teoría de la probabilidad o bases de conocimiento para impulsar su proceso de toma de decisiones.

En la actualidad los sistemas CAD tienen como objetivo asistir en la detección y/o diagnosis de enfermedades a los radiólogos, consiguiendo que estos tengan mejores resultados a la hora de detectar y diagnosticar lesiones. Además, provoca que no tengan que estar tanto tiempo para interpretar las imágenes. Existen dos tipos de sistemas CAD, los sistemas CADe (Computer-Aided Detection) y los sistemas CADx (Computer-Aided Diagnosis). Los primeros están enfocados en detectar la localización de las lesiones, mientras que los segundos se especializan en dar un diagnóstico de las mismas. Por ejemplo, distinguir un tumor entre maligno o benigno.

Los sistemas CADe son los encargados de detectar los nódulos pulmonares y tradicionalmente sus tareas principales son la segmentación de los pulmones, detección de los nódulos candidatos, análisis de características, detección de nódulos y reducción de los falsos positivos. La segmentación de las imágenes pulmonares sirve para separar la región en estudio de otros órganos y tejidos. Con las imágenes segmentadas, se realiza una búsqueda con el objetivo de encontrar estructuras anormales presentes en los pulmones, que pueden ser nódulos. A continuación, se extraen las características de los posibles nódulos detectados. Las principa-

les características que se utilizan habitualmente para la detección de nódulos pulmonares son: valores de intensidad de píxeles, y análisis morfológico y de textura. El clasificador separa los nódulos candidatos en nódulos o no nódulos, el rendimiento final del sistema dependerá de lo bien que realice esta clasificación.

Con el paso de los años se han ido haciendo bases de pacientes tanto con nódulos pulmonares como sin ellos. El objetivo de estas bases de datos es estandarizar y facilitar la validación de sistemas CAD para la detección de nódulos. Algunas de estas bases de datos son LIDC-IRDI, ELCAP, LISS…

Etapas de un un sistema CAD tradicional.

4.1.1 Preprocesado

Se realiza para mejorar la calidad de las imágenes y así obtener mejores resultados. Los pulmones contienen muchas estructuras que pueden ser confundidas con nódulos, por lo que es muy importante mejorar la imagen, que es el objetivo principal de esta etapa. Algunos de los métodos de preprocesado son: filtro mediano adaptativo, filtro medio recortado alfa, filtro gaussiano, filtro Gabor, filtro de paso alto, filtro laplaciano y filtro bilateral. H.Kim [35]. utilizó el filtro mediano (el tamaño del kernel es 3×3) y el filtro superior para reducir el ruido de la imagen y suavizarlo. Teramoto [36], para diferenciar el nódulo del tejido pulmonar normal, utilizó un filtro de erosión en operaciones de preprocesamiento que encogen los nódulos de la imagen y los vasos sanguíneos. El filtro de erosión es un filtro morfológico que amplía la distinción entre objetos de imagen.

4.1.2 Segmentación

Consta de dos partes, extracción del pulmón y segmentación del nódulo. Se realiza la segmentación para dividir la imagen digital en múltiples elementos, con el fin de simplificar y cambiar la representación de la imagen en algo que sea más significativo y más fácil de analizar.

Extracción de pulmón

Los nódulos se clasifican en dos tipos, aquellos que están unidos a la pared pulmonar y aquellos que no. Una tarea fundamental es la separación de los nódulos que están unidos a la pared pulmonar. Existen varias metodologías: S. Shimoyama [37] usa un algoritmo de contornos activos para la segmentación pulmonar. E. M. van Rikxoort [38] usó un método híbrido que tiene 4 pasos: (1) separación del pulmón usando un algoritmo 3D con Crecimiento de Regiones y técnicas morfológicas, (2) un método de detección de errores para el paso anterior, (3) se extraen regiones de los pulmones usando un algoritmo multi-atlas, (4) finalmente los

resultados son evaluados por una algoritmo de detección de errores. M. Alilou [39] empleó un método de umbralización múltiple.

Segmentación del nódulo

Para la segmentación de los nódulos existen diferentes aproximaciones. El principal problema a la hora de la segmentación son las similitudes de los nódulos con otras estructuras como vasos sanguíneos, el tórax o tejidos circundantes. Estos son los responsables del gran número de falsos positivos que puede haber. Por lo tanto, es muy importante la precisión a la hora de segmentar los nódulos, para conocer su tamaño y su crecimiento. M. Keshani [40] identifica nódulos utilizando extracción estadística de características y la anatomía 3D con un clasificador SVM. Después, los bordes del nódulo son extraídos con un algoritmo de contornos activos. El algoritmo propuesto puede segmentar nódulos sólidos y cavitarios que están conectados a la pared pulmonar.

4.1.3 Análisis

Después de la segmentación se obtienen los nódulos candidatos, estos hay que examinarlos por la gran cantidad de falsos positivos que habrá y su estudio requiere de dos partes.

Reducción de falsos positivos

El ratio de falsos positivos es una característica fundamental que diferencia qué sistema CAD es mejor. Además, esta etapa es fundamental a la hora de detectar casos de cáncer en etapas iniciales y así poder iniciar el tratamiento lo antes posible. A. Setio [41] presentó un método de reducción de falsos positivos utilizando CNNs de múltiples vistas. Uno de los métodos de reducción de falsos positivos es eliminar las estructuras con menos similitud con los nódulos, lo que se realiza examinando las características de cada nódulo candidato.

Extracción de características

Para detectar nódulos cancerígenos, se debe realizar la extracción de características de nódulos candidatos. Algunas aproximaciones calculan múltiples parámetros geométricos para cada lesión pulmonar sospechosa, incluyendo su forma, alargamiento, tamaño, especulación, densidad y otras características, para calificar cada nódulo candidato, lo que indica su probabilidad de ser realmente un nódulo pulmonar.

4.1.4 Clasificación

Se centran en la clasificación de los nódulos en función del criterio establecido por el sistema CAD. Los métodos más comunes involucran SVM, redes de neuronas artificiales y

CNN.

4.2 Utilización del deep learning para la detección

La principal contribución del deep learning en el campo del análisis de imágenes médicas ha sido la clasificación de imágenes. El enfoque más común es introducir múltiples imágenes etiquetadas en la red y como resultado tener una única variable de diagnóstico (por ejemplo, existencia de la enfermedad o no). El conjunto de datos médicos suele ser pequeño en comparación con los conjuntos de datos utilizados en visión por computadora (por ejemplo, cientos / miles frente a millones de muestras). Para solucionar este problema se utiliza el aprendizaje por transferencia, y el data augmentation.

Debido a la complejidad de la detección de nódulos pulmonares y la importancia de intentar detectarlos todos, el marco típico se divide en dos tareas principales. El primero se centra en la detección de nódulos candidatos, intenta detectar en el volumen de las tomografías computarizadas todos los nódulos verdaderos, que generalmente incluye una gran cantidad de falsos positivos. Luego, la segunda tarea se especializa en reducir el número de falsos positivos. Algunos trabajos no utilizan esta estrategia y a partir de la TC detectan y clasifican los nódulos directamente. En la Figura 4.1, se puede ver cómo han cambiado las etapas del análisis de nódulos pulmonares con la introducción de los métodos basados en CNN.

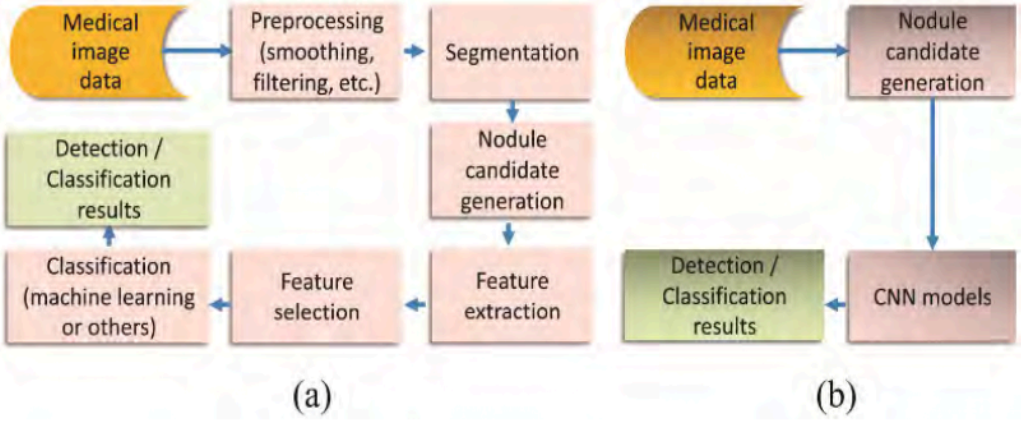


Figura 4.1: (a) Análisis de nódulos pulmonares con métodos clásicos; (b) Análisis de nódulos pulmonares con métodos CNN.

En este apartado se presentan trabajos que se centran en la detección de nódulos y no en su clasificación según su malignidad, estos trabajos varían tanto en la complejidad de sus técnicas como en las aproximaciones utilizadas para obtener sus resultados.

4.2.1 Simultaneous Accurate Detection of Pulmonary Nodules and False Positive Reduction Using 3D CNNs

Utilizando 3D U-Net [42], 3D DenseNet [43] y RPN (Region Proposal Network) [34], Qin et al. [44] proponen un sistema CAD para la detección de nódulos candidatos y la reducción de falsos positivos de forma simultánea. El proceso general de capacitación del sistema se realizó utilizando enfoques de aprendizaje residual multitarea y minería de ejemplos negativos. Se puede ver su arquitectura en la Figura 4.2.

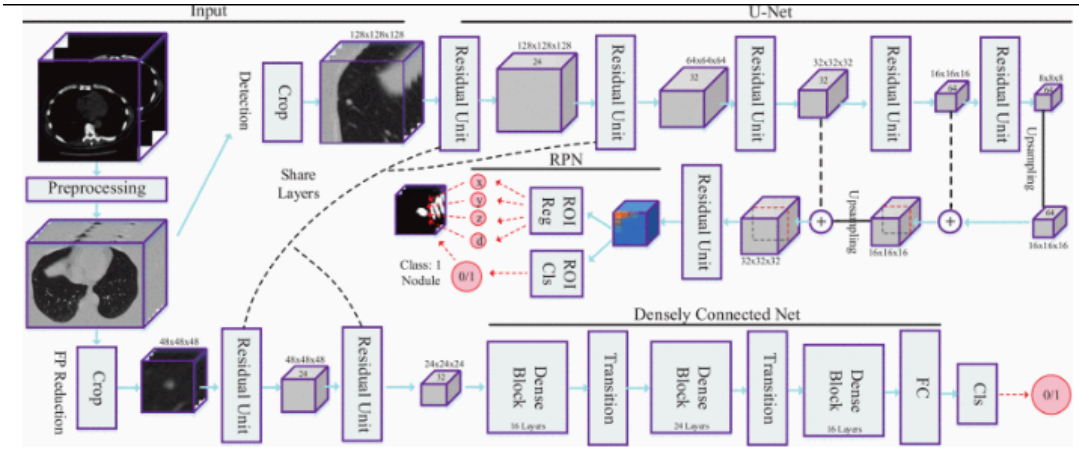


Figura 4.2: Sistema CAD propuesto por Qin.

Para la detección de nódulos candidatos utilizan como backbone una 3D Unet para construir una RPN, con esto consiguen una gran sensibilidad. Se utilizan 3 tipos de anchors para la obtención de los nódulos candidatos, cada anchor corresponde a un clasificador ROI y un regresor ROI. El clasificador ROI clasifica entre nódulo o no nódulo y el regresor ROI hace una predicción del ROI, intentando acercarse lo máximo posible a la verdad fundamental.

La red recibe un corte 3D de la TC, la parte contractora de la red consta de 5 bloques residuales, todos a excepción del primero van seguidos de una capa de agrupación máxima (2 x 2 x 2) para reducir el tamaño del cubo de características. Para la ruta expansiva (hacia atrás), convolucionan hacia arriba el cubo de características y lo concatenan con el cubo de características correspondiente de la ruta contractora, luego lo pasan a través de una unidad residual y una capa de dropout. La salida es introducida en la RPN para la regresión y clasificación ROI.

Cada vóxel en el cubo de salida de la entidad final se parametriza con respecto a los anchors de referencia. Según la distribución de tamaño de los nódulos, se diseñan 3 anchors con diferentes tamaños: $10 \times 10 \times 10$, $20 \times 20 \times 20$, $40 \times 40 \times 40$. Para cada anchor, se calcula su Intersección sobre Unión (IoU) con el boundig-box de verdad fundamental más cercano. A los anchors que tienen IoU superior a 0,5 se les asignan etiquetas positivas. Si la superposición

de IoU de un anchor es inferior a 0,02, se etiqueta como negativa. Los anchors que no sean ni positivos ni negativos serán descartados y no participarán en el entrenamiento.

La función de pérdida del aprendizaje multitarea se compone de pérdida de clasificación y pérdida de regresión. λ es un peso que equilibra los dos términos de pérdida. Dicha función de pérdida multitarea puede generar la probabilidad de ser un nódulo y su posición al mismo tiempo. También mejora la precisión de detección al considerar la relación interna entre la tarea de localizar candidatos a nódulos y la tarea de clasificación:

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda \mathcal{L}_{reg}, \tag{5}$$

Dado un conjunto de pares de entrenamiento $\{(x_i, y_i)\}_{i=1,2,...,N_{cls}}$, la pérdida de WBCE (Weighted Binary Cross-Entropy) entre la etiqueta de la entrada y_i y la salida predicha o_i es:

$$\begin{aligned} \mathcal{L}_{cls} &= -\frac{1}{N_{cls}} \sum_{i=1}^{N_{cls}} w y_i log(o_i) + (1 - y_i) log(1 - o_i), \\ w &= \frac{N_{cls} - \sum_{i=1}^{N_{cls}} y_i}{\sum_{i=1}^{N_{cls}} y_i}, \end{aligned} \tag{2}$$

donde w es el peso atribuido a la clase nódulo, que está determinado por los datos de verdades fundamentales. Obliga al modelo a centrarse en muestras positivas para mejorar la sensibilidad de detección. La pérdida de regresión de la información de ubicación viene dada por:

$$\begin{aligned} \mathcal{L}_{reg} &= \frac{1}{N_{reg}} \sum_{i=1}^{N_{reg}} \sum_{\gamma \in \{x,y,z,d\}} y_i \, \text{smooth}_{L_1}(t_\gamma - t_\gamma^*), \\ \text{smooth}_{L_1}(x) &= \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise}, \end{cases} \end{aligned} \tag{3}$$

donde se utiliza la pérdida smooth L1. t_γ y t_γ^* designan 4 parámetros del bbox predicho y el bbox de verdad fundamental (asociados con un anchor positivo), respectivamente. Los 4 parámetros se definen como:

$$\begin{aligned} t_x = (x - x_a)/d_a, t_y = (y - y_a)/d_a, \\ t_z = (z - z_a)/d_a, t_d = log(d/d_a), \end{aligned} \tag{4}$$

donde x, y, z son coordenadas y d es la longitud del borde del bounding-box predicho. Las variables x_a, y_a, z_a, d_a son parámetros del anchor box. Y $t_x^*, t_y^*, t_z^*, t_d^*$ se definen de manera similar. Esta función de pérdida penaliza la realización de una regresión de ROI desde un anchor box fijo a un bbox de verdad fundamental cercano.

Los cubos candidatos se recortan en función de las coordenadas predichas a partir de la detección de candidatos. Primero se introducen en 2 bloques de unidades residuales, que comparten los mismos parámetros con las primeras capas de convolución en la red de detección. Luego, el mapa de características se agrupa mediante un kernel de 2 × 2 × 2 con stride = 2, seguido de 3 bloques de red densamente conectada con una capa de transición entre dos bloques adyacentes. La capa de transición consta de una capa de Batch Normalization (BN), una capa ReLU, una capa de convolución de 1 × 1 × 1 y una capa de pooling average de 2 × 2 × 2. Al final del último bloque denso, se adjunta una red completamente conectada para clasificar a los candidatos como nódulos verdaderos o falsos positivos. Este framework fue implementado con 4 Nvidia GTX 1080. Logró una sensibilidad del 96,7% y una puntuación de CPM de 0,834 en el conjunto de datos LUNA16.

4.2.2 Accurate pulmonary nodule detection in computed tomography images using deep convolutional neural networks

Ding [45] utiliza una estructura de deconvolución de la Faster R-CNN para la detección de nódulos candidatos en los cortes axiales. Después, utiliza un DCNN 3D para la reducción de falsos positivos. Para la detección de candidatos, cada corte de la tomografía computarizada se concatena con sus dos vecinos y se reescalan a 600 × 600 × 3 píxeles. Para la detección de características se utilizan dos módulos, el primero es una RPN, que genera diferentes ROI; el segundo en un clasificador de ROI, donde se introduce una ROI y determina si es un nódulo o no. Para ahorrar el costo computacional de entrenar dos DCNN, las dos redes mencionadas anteriormente comparten las mismas capas de extracción de características. La red RPN toma una imagen de 3 canales como entrada y genera un conjunto de propuestas de objetos rectangulares (ROI), cada una con una puntuación de objetividad. La Faster R-CNN se entrenó en imágenes naturales y no funcionó bien en imágenes pulmonares, por ello agregaron una capa de deconvolución después del extractor de características (VGG-16). El tamaño de kernel, el tamaño de stride, el tamaño del padding y el número de kernel son 4, 4, 2 y 512 para la capa de deconvolución, respectivamente. Esta elección de diseño condujo a un mejor rendimiento. Para generar ROI, una pequeña red con una ventana de 3 × 3 se desliza a través del mapa de características de la capa deconvolucional, generando un vector de características de 512 dimensiones. Esto finalmente se alimenta en dos capas FC hermanas para hacer una regresión del bounding-box de las regiones y predecir la puntuación de objetividad, respectivamente. Para adaptarse a los diferentes tamaños de los nódulos, se diseñan seis anchor boxes diferentes para cada ventana. Los tamaños son 4 × 4, 6 × 6, 10 × 10, 16 × 16, 22 × 22 y 32 × 32. Para la clasificación de ROI con DCNN, se utiliza una capa de pooling de ROI para asignar cada ROI a un pequeño mapa de características. Funciona dividiendo el ROI en una cuadrícula de 7 × 7 y luego haciendo max-pooling a cada sub-ventana en su celda de salida correspondiente.

Esta salida alimenta a una red FC compuesta por dos capas FC de 4096 vías, que mapean el mapa de características en un vector de características. Se utilizan un regresor y un clasificador basados en el vector de características para obtener los bounding-box de los candidatos y predecir su puntuación de confianza. Se define como función de pérdida que incluye las redes RPN y ROI:

$$\mathcal{L}_t = \frac{1}{N_c}\sum_i \mathcal{L}_c(\widehat{p}_i, p_i^*) + \frac{1}{N_r}\sum_i \mathcal{L}_r(\widehat{t}_i, t_i^*) + \frac{1}{N_{c'}}\sum_j \mathcal{L}_r(\tilde{p}_j, p_j^*) + \frac{1}{N_{r'}}\sum_j \mathcal{L}_r(\tilde{t}_j, t_j^*), \quad (4.1)$$

donde N_c, N_r,$N_{c'}$ y $N_{r'}$ denotan el número total de entradas en Cls Layer, Reg Layer, BBox Cls y BBox Reg, respectivamente. $\widehat{p}_i$ y p_i^* denotan respectivamente la probabilidad predicha y verdadera de que i sea un nódulo. $\widehat{t}_i$ es un vector que representa las 4 coordenadas parametrizadas de la predicción del bounding-box de RPN y t_i^* es el del bounding-box de verdad fundamental asociado con un anchor positivo. De la misma forma, $\tilde{p}_j$, p_j^*, $\widehat{t}_j$, t_j^* denotan los conceptos correspondientes en el clasificador ROI. $\mathcal{L}_c$ es la pérdida logarítmica en dos clases (objeto vs no objeto), y $\mathcal{L}_r$ usa una función de pérdida smooth L1.

Para reducir el número de falsos positivos, eligieron un enfoque DCNN 3D. Esta red contiene seis capas convolucionales 3D, 3 capas de max-pooling, tres capas FC y una última capa de activación softmax de 2 vías para la clasificación. Todas las capas, excepto la última, utilizan activación ReLU. El Dropout se utiliza después de las capas de max-pooling y las capas FC para regularizar la red. La inicialización de los parámetros está determinada por una distribución gaussiana con media cero y la desviación estándar denota el número de conexiones de la respuesta en la l-ésima capa. Como entrada, primero normalizan cada tomografía computarizada con una media de -600 HU y una desviación estándar de -300 HU. Luego se recorta un cubo de 40 × 40 × 24 centrado en el centroide de los nódulos. Para entrenar y probar la arquitectura, se utiliza el conjunto de datos LUNA16. Como estrategia de aumento de datos, para cada 40 × 40 × 24 parches, recortan parches más pequeños de 36 × 36 × 20, aumentando 125 veces para cada candidato. Además, cada parche más pequeño se voltea en tres dimensiones ortogonales. Luego, duplican los parches positivos 8 veces, para equilibrar aún más las clases.

El modelo propuesto logró un CPM de 0,891. Además, para la tarea de reducción de falsos positivos, obtuvo una sensibilidad de 0,922 y 0,944 a 1 y 4 FP/exploración. La detección de candidatos alcanza una sensibilidad de 0,946 con 15 candidatos por exploración. Se puede ver el sistema CAD propuesto en la Figura 4.3.

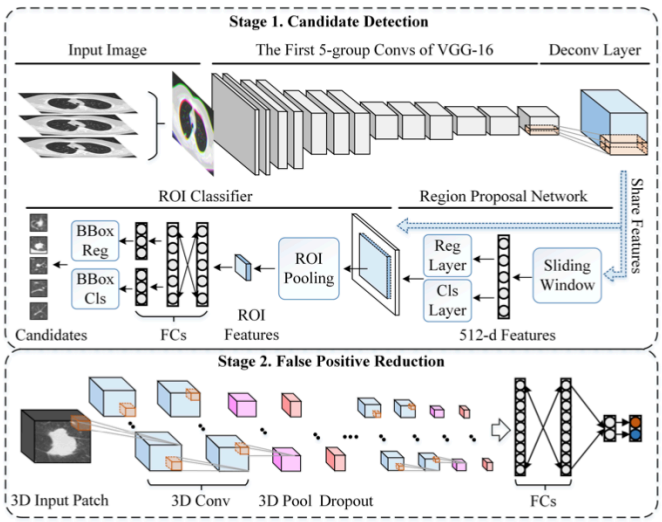


Figura 4.3: Sistema CAD propuesto.

4.2.3 Pulmonary Nodule Detection in CT Images: False Positive Reduction Using Multi-View Convolutional Networks

En "Pulmonary Nodule Detection in CT Images: False Positive Reduction Using Multi-View Convolutional Networks" [41], cuya arquitectura se muestra en la Figura 4.4, utilizan dos etapas, la primera para la detección de nódulos candidatos y la segunda para la reducción de falsos positivos. Tres sistemas CAD existentes [46] [47] [48] se utilizan para la detección de nódulos candidatos, cada uno de ellos se centra en la detección de un tipo de nódulos en particular: sólidos, subsólidos y grandes sólidos. Para cada candidato, se dan la posición y la probabilidad de nódulo. Se calculan tres conjuntos de candidatos a nódulos y se combinan para maximizar la sensibilidad del detector. Los candidatos ubicados a menos de 5 mm entre sí se fusionan. Para estos candidatos combinados, la posición y la probabilidad de nódulo se promedian.

Los nódulos sólidos se detectan siguiendo la técnica en [47], para cada vóxel en los pulmones, se calculan el índice de forma y la curvatura, y se aplica un umbral en las dos medidas para definir los puntos semilla. Se ejecuta un método de segmentación automático en los puntos de partida para obtener los clústeres de interés. Posteriormente, los clústeres ubicados cerca unos de otros se fusionan. Finalmente, se descartan los clústeres con un volumen menor de 40mm x 40mm x 40mm.

Los nódulos subsólidos se detectan siguiendo la técnica en [46], primero se realiza una máscara de densidad de doble umbral (-750, -350 HU) para obtener una máscara con vóxeles de interés. La apertura morfológica se aplica para eliminar los clústeres conectados, seguida de un análisis de componentes conectados en 3D. Los grupos cuyos centros de masa se en-

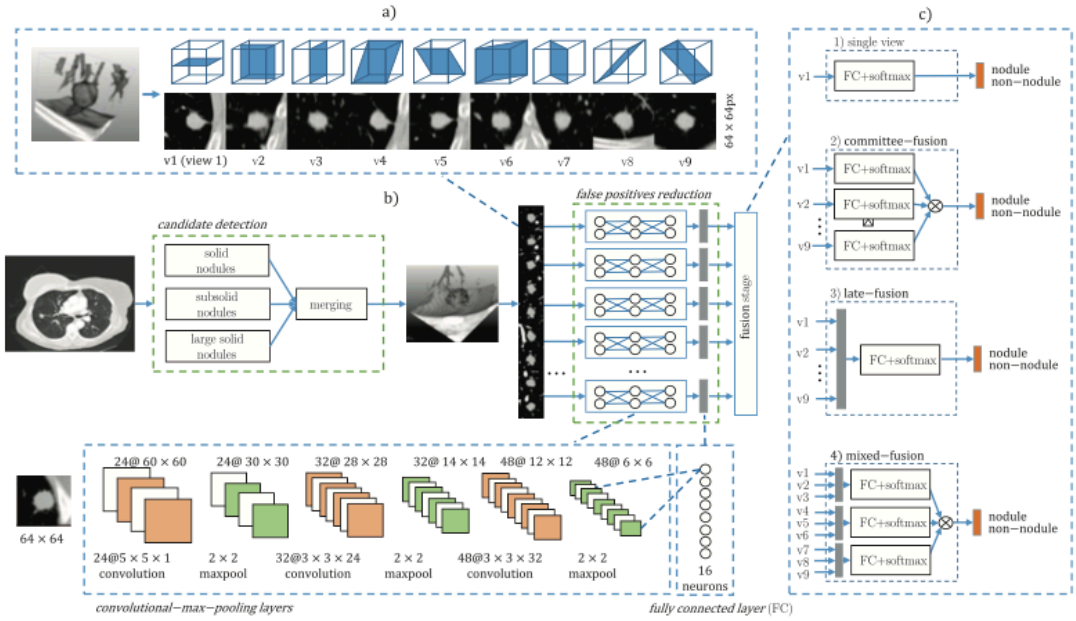


Figura 4.4: CAD propuesto.

cuentran dentro de los 5 mm se fusionan. Por último, se obtiene una segmentación precisa de los candidatos utilizando un algoritmo de segmentación de nódulos previamente publicado en [49].

Los nódulos sólidos grandes, son aquellos mayores de 10mm, tienen valores de índice de superficie/forma que son localmente diferentes de las lesiones sólidas más pequeñas y tienen un rango de intensidad específico que no es captado por los algoritmos de detección de nódulos sólidos y subsólidos. Por lo tanto, los dos algoritmos mencionados anteriormente no funcionan bien en la detección de nódulos sólidos grandes. Además, los grandes nódulos sólidos adheridos a la pared pleural pueden excluirse mediante algoritmos de segmentación pulmonar, ya que el contraste con la pleura es bajo. Por estas razones, como en [48], implementaron un tercer detector que consta de tres pasos: (1) posprocesamiento de la segmentación pulmonar mediante la aplicación de un algoritmo de bola rodante a la máscara de segmentación, que incluye grandes nódulos adheridos a la pleura en la segmentación pulmonar; (2) umbral de densidad (-300 HU), para obtener una máscara con vóxeles de interés; (3) apertura morfológica en múltiples etapas para obtener grupos candidatos, donde se comienza con grandes elementos estructurantes para extraer nódulos más grandes, y progresivamente se continúa con elementos estructurantes más pequeños para extraer nódulos más pequeños. Para evitar el sobreajuste en falsos positivos de alta prevalencia, se descartan candidatos con baja probabilidad de ser nódulos. La probabilidad fue dada por las etapas de clasificación posteriores de los algoritmos existentes [46], [47], [48] y el umbral se establece empíricamente para reducir

un gran número de falsos positivos mientras se mantiene una alta sensibilidad de detección.

Para cada candidato, se extrajeron múltiples parches 2D de 50 x 50 mm centrados. El tamaño del parche se eligió para tener todos los nódulos (<=30mm) completamente visibles en las vistas 2D e incluir suficiente información de contexto para ayudar en la clasificación del candidato. Se cambia el tamaño de cada parche de 50 x 50 mm a un tamaño de 64 x 64 px, trabajando a una resolución de 0,78 mm, que corresponde a la resolución típica de los datos de TC de cortes finos.

El rango de intensidad de píxeles se reescala de (-1000, 400 HU) a (0,1). Se recorta la intensidad fuera del rango dado. Para extraer parches, primero se considera un cubo de 50 x 50 x 50 mm, que encierra al candidato. Se extraen nueve parches en los planos correspondientes al plano de simetría de un cubo. Se utilizan tres planos de simetría que son paralelos a un par de caras del cubo. Estos planos se conocen como planos sagital, coronal y axial. Los otros seis planos son los planos de simetría que cortan dos caras opuestas de cubos en diagonales. Dicho plano contiene dos aristas opuestas del cubo y cuatro vértices.

Las CNN 2D utilizadas para la reducción de falsos positivos constan de 3 capas convolucionales consecutivas y capas de max-pooling. La entrada de la red es un parche 64 x 64. La primera capa convolucional consta de 24 kernel de tamaño 5 x 5 x 1. La segunda capa convolucional consta de 32 kernels de tamaño 3 x 3 x 24. La tercera capa convolucional consta de 48 kernels de tamaño 3 x 3 x 32. Cada kernel produce una salida de imagen 2-D. Los kernels pueden contener diferentes valores de matriz que se inicializan aleatoriamente y se actualizan durante el entrenamiento para optimizar la precisión de la clasificación. La capa de max-pooling viene dada por los valores máximos en ventanas no superpuestas de tamaño 2 x 2 (stride de 2). Esto reduce el tamaño de los parches a la mitad. La última capa es una capa completamente conectada con 16 unidades de salida. Las unidades lineales rectificadas (ReLU) se utilizan en las capas convolucionales y capas completamente conectadas, donde la activación para una entrada dada se obtiene como el máximo entre 0 y el valor de entrada.

Se investigan tres enfoques para fusionar múltiples CNN 2D:

- **Fusión de comités**
  Aplicación de un combinador basado en comités a las predicciones de salida de varias CNN. La motivación es dividir la tarea de detección de un objeto 3-D en varias tareas de detección 2-D más sencillas. Se conecta la salida de la capa completamente conectada de cada flujo a una capa de clasificación que consta de una capa adicional completamente conectada con la función de activación softmax. Cada flujo de CNN se entrena por separado usando parches de una vista específica y las predicciones de salida se combinan.

- **Fusión tardía**
  Concatena las salidas de las primeras capas completamente conectadas y conecta las salidas concatenadas directamente a la capa de clasificación. Con tal método, la capa de

clasificación puede aprender las características 3D comparando las salidas de múltiples CNN.

- **Fusión mixta**
  La fusión mixta es una combinación de los dos enfoques previos. Se implementan múltiples CNN de fusión tardía utilizando un número fijo de planos ortogonales. Aprovechando la ventaja de tener más vistas, la predicción del sistema se mejora al combinar varias CNN fusionadas tardíamente en un comité. Se dividen nueve parches en tres conjuntos independientes; cada juego contiene tres parches diferentes. Aunque se pueden utilizar otros métodos para componer estos conjuntos de parches, se intentan comparar todos los métodos de fusión de manera justa manteniendo la misma información de entrada para cada configuración.

Los resultados para la base de datos LUNA16 (888 escáneres TC y 1186 nódulos anotados) se pueden ver en la Tabla 4.1 para la tarea de detección de nódulos y en la Tabla 4.2 para la puntuación del sistema CAD.

Candidate detection	Detected Nodules	Sensitivity	FPs	FPs/scan
Solid	1016	85,7	292413	329,3
Subsolid	428	36,1	255027	287,2
Large solid	377	31,8	41816	47,1
Combined Set	1120	94,4	543160	611,7
Reduced set	1106	93,3	239941	269,2

Tabla 4.1: Tabla resultados obtenidos para la detección en LUNA16.

Configuration	Number of views	AUC	CPM
combined algorithms	-	0,969	0,573
single view	1	0,969	0,481
committee fusion	3	0,981	0,696
	9	0,987	0,780
late-fusion	3	0,987	0,742
	9	0,993	**0,827**
mixed-fusion	3*3	**0,996**	0,824

Tabla 4.2: Tabla Resultados del sistemas CAD. .

4.2.4 SCPM-Net

Recientemente, Luo et al. [50] proponen un método de free-anchor boxes (cajas de anclaje-libre), cuya arquitectura se observa en la Figura 4.5. La motivación detrás de este trabajo surge ante el problema que suponen las arquitecturas con anchor boxes, tales como Faster R-CNN o RetinaNet, cuando son aplicadas a entornos 3D. Estas arquitecturas funcionan muy bien para

entornos 2D, pero en entornos 3D suelen necesitar de un conjunto muy grande de anchor boxes, por ejemplo más de 100k en 2D RetinaNet (Lin et al.[51]), lo que resulta en un enorme consumo de memoria y computacional para imágenes médicas en 3D. Además, el uso de anchor boxes introduce muchos hiperparámetros y opciones de diseño, incluido el número, el tamaño y la relación de aspecto de los anchor boxes. El diseño manual de estos hiperparámetros no solo requiere de mucho tiempo, sino que también está sujeto a la experiencia humana, lo que puede limitar el rendimiento de detección. En [45] se indica que la detección de pequeños objetos es sensible al diseño manual de anchors. Además, la configuración de anchor predeterminada es ineficaz para detectar lesiones con un tamaño pequeño y una relación de aspecto grande (Zlocha et al. [52]), y mucho menos los nódulos pulmonares más complejos, cuyo tamaño puede variar tanto como diez veces. Por lo que este trabajo surge con el fin de detectar objetos en imágenes médicas 3D con mayor eficiencia y menos configuración manual.

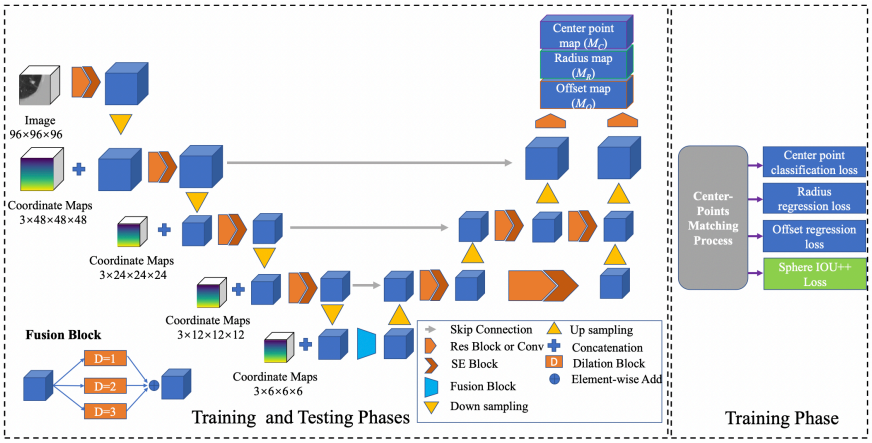


Figura 4.5: Arquitecutra SCPM-Net.

En este trabajo se propone una nueva Red de Coincidencia de Puntos Centrales basada en Representaciones Esféricas 3D para la detección de nódulos pulmonares en tomografías computarizadas. SCPM-Net consiste en un extractor de características que aprovecha los mapas de coordenadas espaciales multinivel y la atención de canal basada en Squeeze-and-Excitation (SE) [53] para una mejor extracción de características, y un cabezal que predice la existencia de un nódulo, su radio y su desplazamiento simultáneamente en dos niveles de resolución. Para ayudar al proceso de entrenamiento, proponen una estrategia de coincidencia de puntos centrales para predecir los K puntos que están más cerca del centro de un nódulo. Introducen una representación de esfera para los nódulos pulmonares en el espacio 3D, y además obtienen una función de pérdida basada en la esfera efectiva (LS IoU ++) para entrenar al detector sin anchors. En comparación con la pérdida tradicional de intersección sobre unión que se basa en bounding-boxes, LS IoU ++ está más cerca de la escena clínica, ya que la estructura geométrica de un nódulo es similar a una esfera o un elipsoide en la mayoría de

situaciones. Además, SCPM-Net es una red de extremo a extremo de una etapa sin módulo adicional de reducción de falsos positivos por lo que tiene una mayor eficiencia y un menor costo de memoria. Podemos ver los resultados de SCPM-Net con respecto a otros trabajos en la Tabla 4.3.

Método	Sensibilidad		
	FPs/Scan=1	FPs/Scan=2	FPs/Scan=8
3D CenterNet [54]	71,4%	75,8%	77,5%
DeepLung [55]	85,8%	86,9%	87,5%
3D RetinaNet [51]	83,6%	86,8%	88,7%
3D RetinaNet++ [51]	88,8%	91,3%	91,4%
CPM-Net [56]	91,2%	92,4%	92,5%
SCPM-Net [50]	92,3%	92,7%	94,8%

Tabla 4.3: Tabla sobre los valores que intervienen en la función de pérdida

La forma de los nódulos pulmonares se asemeja a un elipsoide, por lo que su representación puede ser mucho más precisa en forma de bounding sphere en vez de bounding-box. Además para representar la localización de un nódulo se suele usar sus coordenadas x,y,z junto a su diámetro. Se define $SIoU$ (Sphere Intersection over Union),

$$SIoU = \frac{|S^a \cap S^b|}{|S^a \cup S^b|}, \tag{4.2}$$

donde S^a y S^b denotan la bounding sphere predicha y la verdad fundamental respectivamente. Esto da lugar a la función de pérdida L_{SIoU},

$$L_{SIoU} = 1 - SIoU, \tag{4.3}$$

L_{SIoU} solo funciona cuando la esfera predicha y la verdad fundamental se superponen. En caso de que esto no ocurra se utiliza la relación de distancia y radio (R_{DR}) para la optimización. Dando lugar a:

$$L_{SDIoU} = 1.0 + R_{DR} - SIoU, \tag{4.4}$$

Además, el ángulo de intersección ϕ^{ab} entre las dos esferas también se puede utilizar para describir mejor la regresión de la esfera. Por lo tanto, definen una puntuación de ángulo $\eta = 0$ si $d^{AB} > (r^a r^b)$, en cualquier otro caso $\eta = \frac{arcos(\phi^{ab})}{\pi}$. Finalmente la función de pérdida se define como:

$$L_{SIoU++} = \begin{cases} R_{DR}, \text{ si } d^{AB} > (r^a + r^b) \\ 1.0 + R_{DR} - SIoU + \eta, others \end{cases} \tag{4.5}$$

Para la función de pérdida total se utilizó un método híbrido entre minería dura negativa

Algorithm 1 Pseudocode of LS IoU++ for 3D Lung Nodules Detector Training.
Input: Central coordinates $A(x_1, y_1, z_1)$ and $B(x_2, y_2, z_2)$ and radii r^a and r^b of two spheres in image space respectively.
Output: L_{SIoU++}
Initialization: $d^{AB} = ||A - B||_2$; $R_{DR} = \frac{d^{AB}}{d^{AB}+r^a+r^b}$

if $(r^a + r^b) > d^{AB}$ **then**
    Calculate $|S^a \cap S^b|$
    Calculate $|S^a \cup S^b|$
    Calculate the $SIoU$
    Calculate the angle score η
    Calculate $L_{SIoU++} = 1.0 + R_{DR} - SIoU + \eta$
else
    $L_{SIoU} = R_{DR}$
end if
return L_{SIoU++}

y re-focal loss. Se seleccionan algunas muestras de puntos negativos más difíciles para el entrenamiento, ordenándolos de manera descendente de la pérdida de clasificación del punto central, y seleccionando los N primeros puntos .

$$L_{cls} = \sum_{j=0}^{J} -w_j \alpha (1 - p_j)^{\gamma} \log(p_j), \tag{4.6}$$

El peso de la pérdida refocal se define como:

$$w_j = \begin{cases} 1, \text{ if } j \epsilon P \text{ and } p_j > t \text{ or } j \epsilon N \\ 0, \text{ if } j \epsilon I \\ w, \text{ if } j \epsilon P \text{ y } p_j < t \end{cases} \tag{4.7}$$

p_j denota la probabilidad de que el punto j sea el centro de un nódulo. $J = \frac{D}{R} x \frac{H}{R} x \frac{W}{R}$ es el número total de puntos. t es un umbral para filtrar puntos no calificados y w es un peso para equilibrar la pérdida refocal. También se utiliza smooth L1 loss para hacer una regresión del radio normalizado de la esfera delimitadora:

$$L_{radius}(r, r^*) = \begin{cases} \frac{0{,}5(r-r^*)}{\beta}, \text{ if } |r - r^*| < \beta \\ |r - r^*|, others \end{cases} \tag{4.8}$$

r denota el valor predicho de radio, r^* es la verdad fundamental del radio. Para obtener ubicaciones más precisas de objetos pequeños, se usa la pérdida L2 para hacer una regresión del desplazamiento entre la verdad fundamental de los puntos centrales y los puntos positivos

predichos.

$$L_{offset}(f, f^*) = ||f - f^*||_2, \tag{4.9}$$

donde f y f^* son vectores 3D y denotan los desplazamientos predichos y verdaderos, respectivamente. La pérdida total se expresa mediante la siguiente ecuación:

$$L_{total} = L_{cls} + \alpha(L_{radius} + L_{offset} + \lambda_s L_{SIoU}), \tag{4.10}$$

Capítulo 5

Materiales y Métodos

En este capítulo se detalla tanto el conjunto de datos utilizado como las arquitecturas utilizadas en este trabajo, el cual ha consistido en realizar dos aproximaciones: una en 2D y otra en 3D.

5.1 Base da datos: LIDC-IDRI

Sus siglas vienen de ”Lung Image Database Consortium - Image Database Resource Initiative” y es una colección de imágenes que cuentan con diagnóstico y exploraciones de tomografías computarizadas torácicas para cáncer de pulmón con lesiones marcadas con anotaciones. Para su composición colaboraron siete centros académicos y ocho compañías de imágenes médicas. La base de datos LIDC-IDRI contiene 244167 imágenes repartidas en 1010 casos, cada uno incluye imágenes de una TC torácica y un archivo XML asociado que contiene los resultados del proceso de anotación de imágenes en dos fases, realizado por cuatro radiólogos torácicos experimentados. En la fase inicial, cada radiólogo revisó de forma independiente cada TC y marcó lesiones correspondientes a tres categorías (”no-nódulo”, ”nódulo <3 mm” y ”nódulo > or = 3 mm”). En la siguiente fase, cada radiólogo de forma independiente revisó sus propias marcas junto a las marcas anónimas de los otros radiólogos para obtener una revisión final. El objetivo de este proceso era identificar los nódulos pulmonares en cada TC sin requerir un consenso forzado.

Las TC con un corte mayor de 2,5 mm se recomienda que sean excluidas para la detección de nódulos pulmonares [57] [58], lo recomendado es que su grosor de corte sea menor o igual a 1 mm. Por este motivo se ha optado por utilizar la base de datos LUNA16, para la aproximación en 3D, que es un subconjunto de las base de datos LIDC-IDRI compuesto por todas las TC con un corte menor o igual a 2,5 mm, haciendo un total de 888 escáneres TC. Además se proporcionan imágenes de segmentación pulmonar calculadas utilizando el algoritmo de segmentación automático [59]. También proporcionan un archivo de anotaciones en formato

csv que contiene un resultado por línea. Cada línea contiene el SeriesInstanceUID del escaneo, la posición x, y, z de cada hallazgo en coordenadas mundiales; y el diámetro correspondiente en mm. El archivo de anotaciones contiene 1186 nódulos.

5.2 Framework: Pytorch

Se ha optado por utilizar Pytorch, que es una biblioteca open source de machine learning basada en la biblioteca Torch, esta usaba el lenguaje script LuaJIT y estaba implementada en C, pero en 2019 cerró su desarrollo en favor de otras implementaciones (Pytorch). La decisión del uso de esta biblioteca se realizó tras observar el estado del arte para las tareas de detección en el ámbito de la investigación, y es que se tiende a favorecer más el uso de Pytorch en vez de Tensorflow en el análisis de imágenes médicas. La mayor diferencia entre ambas radica en la utilización de grafos dinámicos en vez de estáticos por parte de Pytorch, lo que posibilita que en tiempo de ejecución las funciones se vayan modificando y que el calculo de gradiente varíe con ellas. En Tensorflow lo primero que hay que hacer es definir el grafo de computación y tras ellos utilizar la session para calcular los resultados de los tensores, lo que dificulta la depuración del código y su implementación. Un ejemplo conocido por todo el mundo es Tesla Autopilot que está implementado en Pytorch.

Pytorch está desarrollado principalmente por el Laboratorio de investigación de Inteligencia Artificial de Facebook, su interfaz principal es en Python aunque también cuenta con un versión en C++. Sus dos características principales son :

- Computación de tensores (como NumPy) con una aceleración fuerte a través de unidades de procesamientos gráficos (GPU).
- Diferenciación automática para construir y entrenar redes neuronales. Evalúa numéricamente la derivada de una función. La diferenciación automática calcula los pases hacia atrás en las redes neuronales.

Además cuenta como parte de su ecosistema con la librería torchvision en la que existen multitud de modelos entrenados para las tareas de visión artificial.

5.3 Aproximación 2D

Para esta aproximación se han utilizado diferentes métodos, todos entorno a la utilización de una Faster RCNN. Utilizando VGG16 como backbone y aplicando estrategias de transfer learning. Al fijarse en los trabajos detallados en el capítulo del estado del arte, se puede observar que los más recientes utilizan redes 3D en vez de 2D, esto es debido a que aunque son más costosas computacionalmente, suelen obtener mejores resultados. Al decir recientes

nos referimos a más cercanos al 2021, ya que hasta hace muy pocos años sí que se utilizaban más asiduamente, pero con el avance de las GPUs las aproximaciones en 3D han aumentado considerablemente.

5.3.1 Estructura red: Faster RCNN VGG16

Se ha seguido la arquitectura propuesta en [45], utilizando una Faster RCNN con un backbone de VGG16 sin la última capa de clasificación, a mayores se le ha añadido una capa de deconvolución. El tamaño de kernel, el tamaño de stride, el tamaño del padding y el número de kernel son 4, 4, 2 y 512 para la capa de deconvolución. La RPN funciona en el mapa de características de salida de la deconvolución, y consiste en deslizar una ventana 3 x 3 por este mapa de características con el objetivo de obtener propuestas de región candidatas con una puntuación de objetividad. Los tamaños de los anchor son 4 × 4, 6× 6, 10 × 10, 16 × 16, 22 × 22 y 32 × 32. La capa pooling de ROI consiste en un tamaño de cuadrícula de 7x7 haciendo max-pooling. La salida de esto va a las dos capas FC con un tamaño de 4096 que finalmente proporciona una puntuación de objetividad y un bounding-box. El código utilizado es el siguiente:

```
num_classes = 2
class BoxHead(nn.Module):
    def __init__(self, vgg):
        super(BoxHead, self).__init__()
        self.classifier = nn.Sequential(*list(vgg.classifier._modules.values())[:-1])
        self.in_features = 4096 # feature out from mlp

    def forward(self, x):
        x = x.flatten(start_dim=1)
        x = self.classifier(x)
        return x
vgg = torchvision.models.vgg16(pretrained=True)
backbone = vgg.features[:-1]
#for layer in backbone[:10]:
    #for p in layer.parameters():
        #p.requires_grad=False
backbone.out_channels=512
backbone.classifier=torch.nn.ConvTranspose2d(512,512,4,4,2)
anchor_generator = AnchorGenerator(sizes=((4,6,10,16,22,32),),aspect_ratios=((1),))
roi_pooler = torchvision.ops.MultiScaleRoIAlign(featmap_names=['0'], output_size=7, sampling_ratio=2)
box_head = BoxHead(vgg)
in_features = box_head.in_features

model = torchvision.models.detection.faster_rcnn.FasterRCNN(
                backbone, #num_classes,
                rpn_anchor_generator = anchor_generator,
                box_roi_pool = roi_pooler,
                box_head = box_head,
                box_predictor = FastRCNNPredictor(in_features, num_classes))

model.to(device)
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.01, momentum=0.9, weight_decay=0.0005)
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=12, gamma=0.1)
```

y el modelo resultante es:

```
FasterRCNN(
  (transform): GeneralizedRCNNTransform(
      Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

```
        Resize(min_size=(600,), max_size=600, mode='bilinear')
    )
    (backbone): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU(inplace=True)
      (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU(inplace=True)
      (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (6): ReLU(inplace=True)
      (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (8): ReLU(inplace=True)
      (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (11): ReLU(inplace=True)
      (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (13): ReLU(inplace=True)
      (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (15): ReLU(inplace=True)
      (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (18): ReLU(inplace=True)
      (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (20): ReLU(inplace=True)
      (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (22): ReLU(inplace=True)
      (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (25): ReLU(inplace=True)
      (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (27): ReLU(inplace=True)
      (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (29): ReLU(inplace=True)
      (classifier): ConvTranspose2d(512, 512, kernel_size=(4, 4), stride=(4, 4), padding=(2, 2))
    )
    (rpn): RegionProposalNetwork(
      (anchor_generator): AnchorGenerator()
      (head): RPNHead(
        (conv): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (cls_logits): Conv2d(512, 7, kernel_size=(1, 1), stride=(1, 1))
        (bbox_pred): Conv2d(512, 28, kernel_size=(1, 1), stride=(1, 1))
      )
    )
    (roi_heads): RoIHeads(
      (box_roi_pool): MultiScaleRoIAlign(featmap_names=['0'], output_size=(7, 7), sampling_ratio=2)
      (box_head): BoxHead(
        (classifier): Sequential(
          (0): Linear(in_features=25088, out_features=4096, bias=True)
          (1): ReLU(inplace=True)
          (2): Dropout(p=0.5, inplace=False)
          (3): Linear(in_features=4096, out_features=4096, bias=True)
          (4): ReLU(inplace=True)
          (5): Dropout(p=0.5, inplace=False)
        )
      )
      (box_predictor): FastRCNNPredictor(
        (cls_score): Linear(in_features=4096, out_features=2, bias=True)
        (bbox_pred): Linear(in_features=4096, out_features=8, bias=True)
      )
    )
)
```

5.3.2 Preprocesado

Partiendo del conjunto de datos de LUNA16, que como ya se ha dicho previamente, es un subconjunto de la base de datos LIDC-IDRI, en el que sólo se incluyen las TC con un corte menor a 2,5mm, lo que resulta en 888 TC con un total de 1189 nódulos. Las 888 TC están

divididos en 10 subconjuntos, con 89 TC cada uno a excepción de los 2 últimos que tienen 88.

El preprocesado para la aproximación 2D se ha centrado en identificar los cortes axiales de todos los escáneres en los que había nódulos. Se ha utilizado la herramienta Pylidc, que es una herramienta de mapeo relacional de objetos para los datos del conjunto LIDC, lo que posibilita consultar los datos de LIDC de una forma similar a SQL. Tiene cuatro clases principales en su API, Scan, Annotation, Contour y Utilities. Se seleccionan las TC con grosor de cortes axiales menores o iguales a 2,5mm. Para cada TC se busca si tiene nódulos o no, en caso de que no los tenga se descarta, si los tiene se crea una máscara con un consenso del 50% (aportado por los radiólogos que realizaron la base de datos, ver Figura 5.1) y se guarda el escáner en forma de imágenes .png. Antes de guardarlas se normaliza el volumen entre 400HU y -1000HU (el resto de valores no son de utilidad para el problema), y por último se aplica la técnica de zero center. A partir de la máscara creada se guardan los bounding-box correspondientes a cada nódulo, anotando en un archivo csv el path del corte axial, el seriesUID, los bounding-box y el subconjuntos al que corresponde. En la Figura 5.2 se puede ver un corte con el bounding-box dibujado en rojo y sus coordenadas en la parte superior.

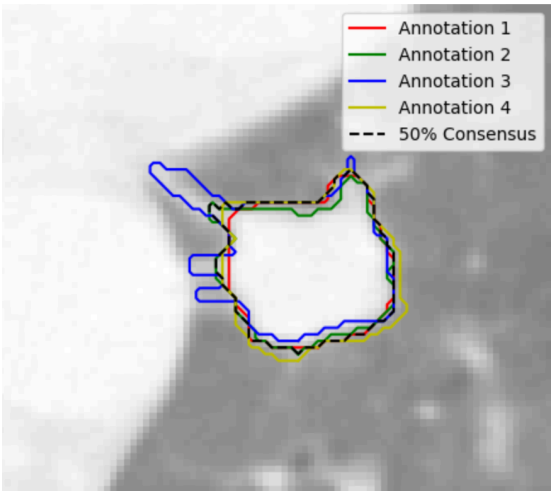


Figura 5.1: En la imagen se ven las diferentes anotaciones realizadas por los 4 radiólogos sobre el nódulos, y a mayores el consensus 50% calculado y usado para crear los bounding box .

```
MIN_BOUND = -1000.0
MAX_BOUND = 400.0

def normalize(image):
    image = (image - MIN_BOUND) / (MAX_BOUND - MIN_BOUND)
    image[image>1] = 1.
    image[image<0] = 0.
    return image
```

```
PIXEL_MEAN = 0.25 #Calculado para todo el dataset de LUNA16

def zero_center(image):
    image = image - PIXEL_MEAN
    return image
```

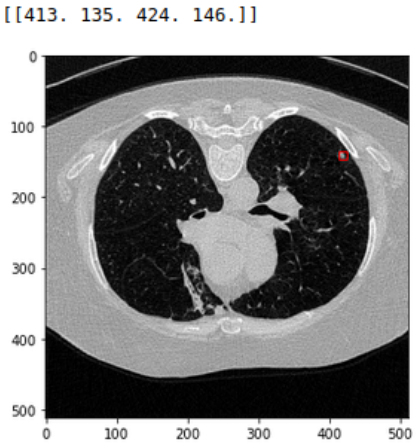


Figura 5.2: Corte con bbox y sus coordenadas.

5.3.3 Entrenamiento

El archivo csv se divide en 3 partes, entrenamiento, validación y test, con una proporción del 80% de las imágenes para entrenamiento, y un 20% para test y validación a partes iguales. Haciendo uso de la clase $torch.utils.data.Dataset$, se crea la forma en la que se cargarán los datos durante los procesos de entrenamiento, validación y test. Se ha hecho uso del tipo de Map-style Dataset que implementa los métodos $getitem()$ y $len()$, este último da la longitud del conjunto de datos que se usa, mientras que el primero se encarga de generar la lógica para coger los datos, la imagen y los bbox, coge los datos en función del índice que ocupe en el conjunto, por lo que, si el índice es 0, entonces cogerá la ruta a la imagen en la posición 0 del conjuntos de datos y los bbox que le correspondan a esta imagen.

Se aplica data augmentation a las imágenes de entrenamiento, con una probabilidad de 0,5 de que se aplique cada transformación, siendo estas flips horizontales, verticales y rotaciones en el rango de 20 a -20 grados. Los bounding box se guardan en formato Pascal VOC y se ajustan automáticamente a todas las transformaciones a las que la imagen es sometida gracias a la librería albumentations.

```
def get_transforms(phase):
    list_transforms = []
    if phase == 'train':
        list_transforms.extend([
                HorizontalFlip(p=0.5),
                VerticalFlip(p=0.5),
                Rotate((-20,20),p=0.5),
                  ])
    list_transforms.extend(
            [
    ToTensorV2(),
            ])
    list_trfms = Compose(list_transforms,
                         bbox_params={'format': 'pascal_voc', 'label_fields': ['labels']})
    return list_trfms
```

También se hace uso de la clase $torch.utils.data.Dataloader$, que es la que carga los datos (bien en cpu o gpu). Acepta como argumentos un dataset (en map-style o iterable style), el tamaño de batch (indica en número de muestras generadas en cada batch generado), shuffle (para mezclar o no los datos del dataset), num_workers (indica el número de procesos que generan batchs en paralelo) y collate_fn que recopila listas de muestras en lotes.

```
# batching
def collate_fn(batch):
    return tuple(zip(*batch))

train_data_loader = DataLoader(
    train_data,
    batch_size=4,
    shuffle=True,
    num_workers=2,
    collate_fn=collate_fn
)
```

5.3.4 Resultados

Se han seguido diferentes estrategias de entrenamiento, variando la forma de selección de las imágenes para entrenar, utilizando transfer learning, alterando los parámetros de learning-rate, momentum, step-size, weight-decay,epochs... En este apartado se explicarán directamente los resultados definitivos, haciendo mención al proceso para llegar hasta ellos pero sin profundizar en cada etapa del proceso.

Para configurar los hiperparámetros, primero el modelo fue probado para 12 epochs con un step-size de 3 y un learning-rate de 0,01, al ver que los resultados no eran buenos se fueron modificando hasta llegar a una etapa de refinamiento en la que se había conseguido un buen resultado y lo que se hizo a partir de ahí fueron pequeñas modificaciones de los hiperparámetros para mejorarlo. Lo más complicado fue encontrar el equilibrio necesario entre learning-rate y step-size, una vez encontrado ya se sabía el número de epochs que el modelo debía de estar con un determinado learning-rate para evitar que dejase de aprender y empezase a sobre-entrenarse. De esta forma fue como se consiguió llegar a los mejores resultados que se presentan en esta memoria.

Las técnicas de transfer learning empleadas han consistido en utilizar un modelo preentrenado como extractor de características y también el fine tuning del modelo, dejando durante X etapas del aprendizaje los pesos congelados de determinadas capas del modelo.

En cuanto a las imágenes se ha probado con pasarle a la red un único corte o también el corte junto al anterior y posterior, convirtiendo la imagen de 1x512x512 a 3x512x512. Optando finalmente por pasarle a la red el corte y sus adjuntos. Cada vez que una imagen entra en la red la primera operación es cambiarle el tamaño a 600x600 píxeles.

Los mejores resultados han sido utilizando un lr=0.01 ,momentum=0.9 ,step-size=12,weight-decay=0.0005 ,epochs=36, apilando junto al corte axial original sus dos vecinos y utilizando

un modelo de VGG16 pre-entrenado en la base de datos COCO. Consiguiendo un AP50 (Average Precision en 0.5 IoU) de 70,56% y un mAP (mean Average Precision, esto es la suma de los average precision desde 0.5 a 0.95 IoU siguiendo incrementos de 0.05) del 37,32%. Se puede ver en la Figura 5.9a la curva Precision-Recall. Los resultados obtenidos han sido mejores que en [60], pero distan mucho de ser unos resultados equiparables a los mejores sistemas.

Con el fin de caracterizar el modelo, se hicieron pruebas en las que se varía el área de los bounding-box del conjunto de test. A continuación se detallan los resultados, además se pueden ver el AP en la Tabla 5.1 y la curva Precision-Recall en la Figura 5.3.

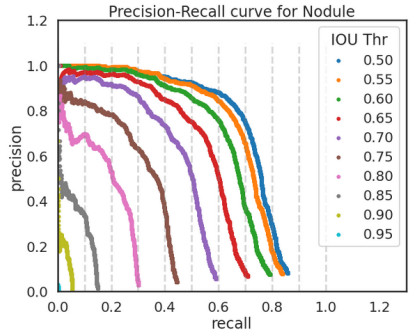
- La sensibilidad máxima del sistema fue de 85,88%, AP50 de 70,56% y mAP de 37,32 para las 1211 muestras totales de test.
- La sensibilidad máxima del sistema fue de 87,45%, AP50 de 73,16% y mAP de 39,36 para las 1116 muestras con un área mayor o igual a 20 píxeles de test.
- La sensibilidad máxima del sistema fue de 87,23%, AP50 de 75,02% y mAP de 42,04 para las 817 muestras con un área mayor o igual a 50 píxeles de test.
- La sensibilidad máxima del sistema fue de 85,05%, AP50 de 74,33% y mAP de 41,95 para las 585 muestras con un área mayor o igual a 100 píxeles de test.

IoU	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
AP_{total}	0.7056	0.6814	0.6209	0.5509	0.4534	0.3284	0.2064	0.1217	0.0606	0.0028
$AP_{area>=20}$	0.7316	0.7142	0.6629	0.5805	0.4822	0.3458	0.2324	0.123	0.0606	0.0028
$AP_{area>=50}$	0.7502	0.7378	0.6926	0.6258	0.5357	0.4026	0.2667	0.1287	0.0606	0.003
$AP_{area>=100}$	0.7433	0.7269	0.6811	0.6183	0.5313	0.3975	0.2711	0.1614	0.0606	0.0034

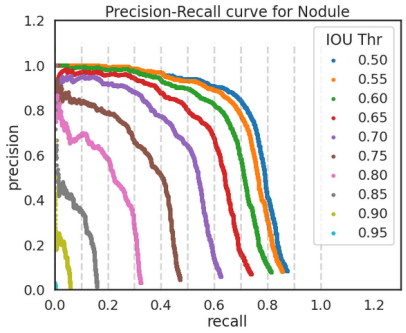
Tabla 5.1: Average Precision para un determinado IoU.

Al analizar los resultados obtenidos, se puede observar que el sistema es capaz de generalizar y obtener resultados parecidos independientemente del tamaño de nódulo. Los nódulos difusos son aquellos que más le cuesta identificar. En función del tamaño del área de la muestra, el grupo cuyo área es menor a 20 píxeles, le supone una mayor dificultad de detección, su sensibilidad máxima la alcanza en el rango de áreas 20-50 píxeles, mientras que su mejor precisión es el rango 50-100 píxeles. Por lo que es posible afirmar que las muestras que mejor reconoce son los nódulos de tamaño mediano. Además, se observa cómo los resultados empeoran cuando las áreas de las muestras a tener en cuenta son mayores a 100, esto se debe a que el sistema distingue mejor un determinado tipo de nódulos que otro.

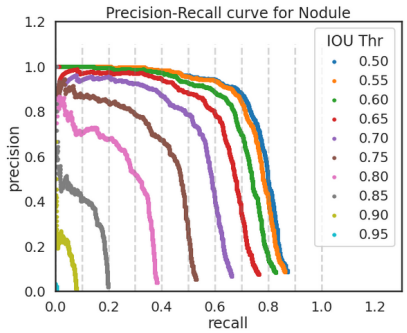
Con el objetivo de mejorar el sistema se podría realizar un estudio sobre los nódulos que tiene más dificultades en detectar, porque como se aprecia en los resultados, el sistema funciona correctamente independientemente del tamaño del nódulo. Habría que estudiar las características de los nódulos no detectados y buscar rasgos comunes identificables. Una vez



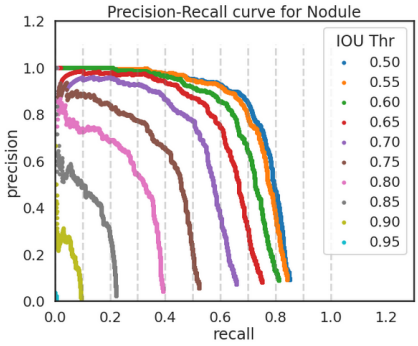
(a) Total



(b) Area>=20



(c) Area>=50



(d) Area>=100

Figura 5.3: Curvas Precision Recall

identificados se modificaría el sistema, siendo necesaria una etapa de preprocesado más óptima y una reconfiguración de los anchor de la RPN.

5.4 Aproximación 3D

Se ha seguido el trabajo de Yuemeng Li, Young Fan [61]. Proponen una nueva red llamada Deep Squeeze-and- Excitation Encoder-Decoder, DeepSEED, que detecta nódulos en un solo paso (no cuenta con etapa de reducción de falsos positivos). Está construida sobre una 3D RPN con una estructura squeeze-and-excitation [53] y a su vez la 3D RPN está construida sobre una estructura Encoder-Decoder. Para combatir el desequilibrio de muestras, la red utiliza focal loss, función de pérdida focal, que es una función de pérdida de entropía cruzada escalada dinámicamente.

5.4.1 Estructura de la red

Se puede ver la arquitectura de la red en la Figura 5.4. El número en la parte superior izquierda de cada bloque indica el número de canales y el número en la parte inferior derecha indica las dimensiones espaciales de los mapas de características de entrada. (a) Un encoder

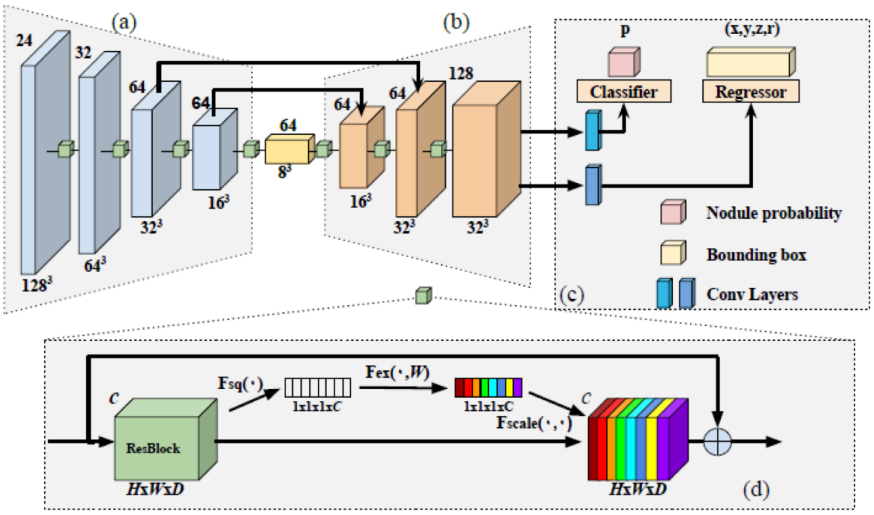


Figura 5.4: Arquitectura de DeepSEED.

con estructura 3D ResNet-18. (b) Un decoder con su dimensión de característica ampliada. (c) Una RPN para identificar nódulos candidatos y predecir sus boundig-boxes. (d) Los bloques residuales de squeeze-and-excitation (SE) que se adoptan en todos los bloques convolucionales en (a) y (b). En particular, cada bloque residual se comprime en un vector de características de c-dimensión (bloques blancos) que pasa a través de un mecanismo de activación para aplicar la codificación espacial (bloques de color) al bloque residual, con diferentes colores que indican diferentes pesos.

Para cada capa convolucional, definen $U = F(X)$, donde X son las características de entrada, y $U = [u_1, u_2, ..., u_c]$, $u_i \epsilon R^{HxWxD}$ es la salida de la matriz de características después de la convolución. Para "apretar"(squeeze) la información espacial global en un descriptor de canal, utilizan la agrupación de promedios globales (global average pooling) por canal. La salida que se define como $z_i = F_{sq}(u_i)$, es un elemento de $Z \epsilon R^C$, que se ha generado al "apretar" U a través de las dimensiones espaciales $HxWxD$.

La salida de canal de la operación de squeeze se utiliza para modular las interdependencias de todos los canales a través de la excitación, un mecanismo de puerta con activación sigmoidea:

$$s_i = F_{ex}(u_i) = \sigma(g(Z, W)) = \sigma(W_2 \delta(W_1 Z)) \tag{5.1}$$

donde $\sigma(\cdot)$ es la función de activación sigmoide, $\delta(\cdot)$ es la función ReLU, $W_1 \epsilon R^{\frac{C}{r} x C}$ contiene los parámetros para las capas de reducción de la dimensionalidad, $W_2 \epsilon R^{C x \frac{C}{r}}$ contiene los parámetros para las capas de aumento de la dimensionalidad. Usaron $r = 16$.

Finalmente, se aplica una multiplicación F_{scale} basada en canales entre cada escalar de

excitación s_i y el mapa de características u_i para generar la salida de características U final:

$$U = F_{scale}(u_i, s_i) = s_i \cdot u_i \tag{5.2}$$

5.4.2 Funciones de pérdida de clasificación y regresión

El entrenamiento de la RPN se centra en minimizar, primero, la función de pérdida de clasificación para diferenciar los nódulos positivos de los negativos, y segundo, la función de pérdida de regresión para predecir los bounding-box. Como ya hemos dicho la función de pérdida de clasificación es focal loss, que dada la probabilidad de salida de clasificación de una muestra positiva p, la focal loss se define como:

$$L_{cls} = -\alpha(1 - p_t)^{\gamma} \log(p_t) \tag{5.3}$$

donde $p_t = p$ si la verdad fundamental es $\gamma = 1$, si no $p_t = 1 - p$, α es un factor balanceador para la focal loss, y γ es un parámetro de enfoque ajustable. En este trabajo usaron $\alpha = 0.5$ y $\gamma = 2$.

La función de pérdida de regresión se define como:

$$L_{reg} = \sum_{k} S(G_k, P_k) \tag{5.4}$$

donde $S(\cdot)$ es la función smooth L1-norm, $G_k = (\frac{x_g - x_a}{r_a}, \frac{y_g - y_a}{r_a}, \frac{z - g_a}{r_a}, \log(\frac{r}{r_a}))$ es la verdad fundamental parametrizada, $P_k = (\frac{x - x_a}{r_a}, \frac{y - y_a}{r_a}, \frac{z_a}{r_a}, \log(\frac{r}{r_a}))$ es la predicción parametrizada, x_a,y_a,z_a y r_a son la localización espacial y el radio de un anchor. k es el índice de los anchor en un mini-batch. La función de pérdida general es:

$$L = L_{cls} + p^* L_{reg} \tag{5.5}$$

donde $p^* = 1$ para las muestras positivas y 0 para las negativas.

5.4.3 Preprocesado

Lectura de la imagen

Las imágenes están guardadas en formato .raw y tienen un archivo .mhd en el que se detalla la información sobre las mismas. Estos archivos contienen una gran cantidad de metadatos (como el tamaño de píxel, que se corresponde a la longitud de un píxel en cada dimensión en el mundo real). Este tamaño de píxel/grosor del escaneo difiere de un escáner TC a otro (por ejemplo, la distancia entre los cortes puede diferir), lo que puede afectar el rendimiento de los enfoques de CNN. La forma de lidiar con esto es mediante un remuestreo isomórfico, que se

hará más adelante. En la Figura 5.5 se puede ver un corte axial de la imagen de entrada.

Lo primero es leer el archivo .mhd, para obtener el volumen 3D, el origen y el spacing del volumen. Después, se extrae la segmentación de pulmón proporcionada por Luna16, que se utiliza para crear la máscara de volumen TC y obtener los bounding-box de cada nódulo anotado en el volumen correspondiente.

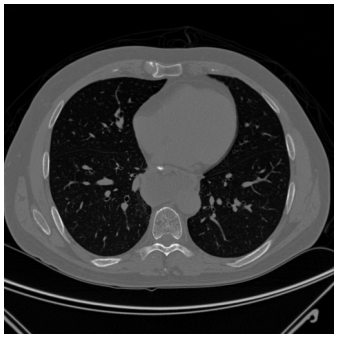


Figura 5.5: Corte de la imagen de entrada.

Lectura de la segmentación pulmonar

De la misma forma que en el apartado anterior se obtine toda la información sobre el volumen TC, en este se obtienen las segmentaciones pulmonares. Estas han sido previamente generadas por el algoritmo propuesto en [46]. Un ejemplo de segmentación pulmonar se observa en la Figura 5.6. A esta segmentación se le aplica un procesado con el fin de dilatar cada lóbulo pulmonar por separado. El objetivo de esto es solventar posibles segmentaciones erróneas en las que se hayan podido obviar nódulos que estén pegados a la pleura pulmonar. En la Figura 5.7 se puede observar la nueva máscara para la segmentación y el espacio que los lóbulos pulmonares ocuparían en esta nueva segmentación.

Figura 5.6: Corte de la segmentación pulmonar.

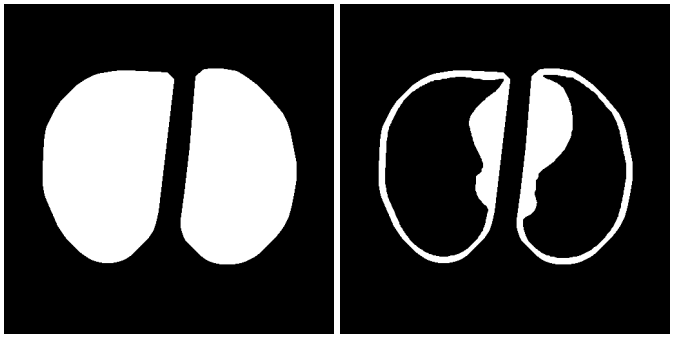


Figura 5.7: Dilatación de la segmentación y segmentación extra resultante.

Remuestreo de la imagen

Es importante hacer que los cortes sean lo más homogéneos posible. Para hacer esto, se ha cambiado la escala de cada corte para que cada vóxel representara un volumen de 1x1x1 mm, lo que conlleva un remuestreo isomórfico de la imagen. Además todas las intensidades se recortaron en el valor hounsfield mínimo (-1200 HU), máximo (600HU) y luego se escalaron entre 0 y 1 para normalizarlos. El resultado de esto se puede ver en la Figura 5.8.

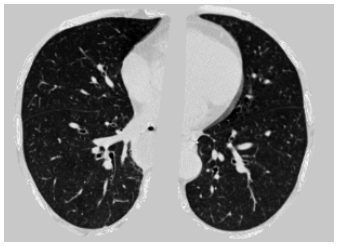


Figura 5.8: Corte de la segmentación pulmonar.

Creación bounding-box

Por último tras guardar la imagen, su spacing, su origen y el extended box, se guarda para cada imagen con nódulos, la posición de este de la imagen remuestrada con un formato de x,y,z,mm. Donde la x corresponde a la posición del nódulo en el eje x, la y para el eje y, la z para el eje z y mm, hace referencia al diámetro del nódulo.

5.4.4 Entrenamiento

Haciendo uso de la clase $torch.utils.data.Dataset$ se crea una clase dataset que será la encargada de crear las muestras de cubos de tamaño 128x128x128 que pasan a la red, estas muestras han sido ajustadas de forma aleatoria al tamaño necesario y han sufrido un proceso de data augmentation con transformaciones horizontales y rotaciones. Para el entrenamiento

los anchor boxes con IoU>=0,5 fueron definidas como muestras positivas y las que tenían un IoU<=0,02 como negativas. Los anchor utilizados por la RPN fueron de 5, 10 y 20.

Con $torch.utils.data.Dataloader$ se crea una clase dataloader para la carga de los datos, a este dataloader se le pasa un dataset, el tamaño batch (16 para el modelo usado), shuffle igual a True para el dataloader de entrenamiento, num_workers=32 y pin_memory=True. Este último parámetro sirve para cuando las muestras del dataset se cargan en cpu, pero se quieren entrenar en la gpu, activando pin_memory se permite al dataloader guardar las muestras en page-locked memory lo que permite una mayor velocidad de transferencia entre cpu y gpu.

Se utiliza el optimizador SGD, la tasa de learning-rate es de 0,01, el momentum es 0,9 y el weight-decay es 1e-4. En función del número de epochs el learning-rate se modifica, se deja el valor inicial para el 20% del total de epochs, se multiplica por 0,1 hasta el 40%, se vuelve a multiplicar pero por 0,05 hasta el 60% y para el resto se vuelve a multiplicar por 0,01.

5.4.5 Resultados

Se ha utilizado un modelo pre-entrenado con 8 NVIDIA 1080Ti GPUs un tamaño batch de 40 y el learning rate de 0,01. Este modelo ha sido entrenado y testado usando una estrategia de 10 fold training. Las estrategias de k-fold cross validation sirven para estimar lo bueno que es un modelo cuando la cantidad de datos es limitada. Consiste en hacer k grupos distintos, en cada k grupo se deja un parte del total de muestras para entrenar el modelo y otra para testarlo. De esta manera el modelo se entrena y prueba en datos distintos para cada k iteración, lo que provoca que la estimación del modelo sea más imparcial y menos optimista. El procedimiento que se utiliza es el siguiente:

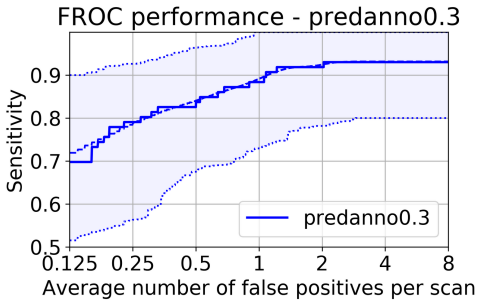
- Mezclar el conjunto de datos de forma aleatoria.
- Dividir el conjunto de datos en k grupos
- Para cada grupo único:
  - Tomar el grupo como un conjunto de datos de prueba o de reserva
  - Tomar los grupos restantes como un conjunto de datos de entrenamiento
  - Colocar un modelo en el conjunto de entrenamiento y evaluarlo en el conjunto de prueba
  - Conservar la puntuación de la evaluación
- Finalmente la habilidad del modelo será la media de las puntuaciones obtenidas

En este caso, el fold utilizado tenía el 90% de las imágenes para entrenar y el 10% para la parte de validación, por lo que se ha utilizado la parte de validación para testar el modelo,

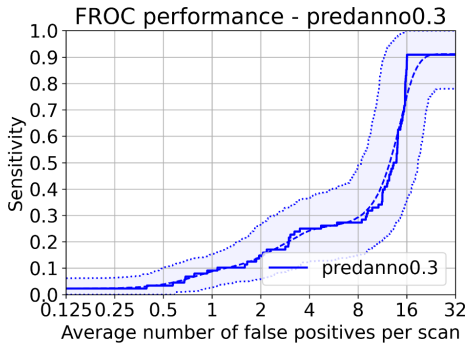
ya que aunque esta haya sido evaluada durante el entrenamiento, no ha tenido nada que ver a la hora de calcular los pesos del modelo. Para las pruebas, las imágenes se dividen en regiones de tamaño 208 × 208 × 208 con una superposición de 32 píxeles para las regiones vecinas. El rendimiento del modelo se evaluó con el script oficial de la competición LUNA16. Con esto se ha obtenido una sensibilidad para el modelo de 90,09%. Habiendo detectado como verdaderos positivos 80 nódulos, como falsos positivos 1435 nódulos y como falsos negativos 8 nódulos. Lo que provoca un número medio de falsos positivos por TC de 19,6, este número si se compara con otros sistemas de detección es muy alto para la sensibilidad que tiene el sistema. Por ejemplo, [45] obtiene una sensibilidad de 0,922 y 0,944 para 1 y 4 falsos positivos por exploración, o [50] obtiene 0,923 y 0,927 para 1 y 2 falsos positivos por exploración. Sin embargo, al compararlo con [41] obtiene un mejor balance, al conseguir este artículo una sensibilidad 0,944 para 611,7 falsos positivos por exploración.

Como en este trabajo no se busca crear soluciones nuevas, no se ha visto necesario desarrollar un nuevo modelo, en caso de que lo que se buscase fuese crear un modelo de cero se tendría que intentar mirar la capacidad de generalización del modelo y para eso sería necesario hacerlo con un conjunto de datos distinto al que se usó para entrenarlo y repetir el proceso una serie de veces (debido a que el conjunto de datos con el se cuenta es muy limitado comparado con otras tareas de deep learning). En [61] la puntuación FROC media fue de 86,2% para el conjunto de datos LUNA16 y 77,3% para 122 escáneres no presentes en LUNA16 de la base de datos LIDC-IDRI. Viendo estos resultados vemos el gran impacto, comentado anteriormente, que suponen las TC con un grosor de corte mayor que 2,5mm.

Los resultados obtenidos en este TFG son muy diferentes a los obtenidos en [61], con respecto a la sensibilidad del sistema apenas existe diferencia, pero en cuanto al número de falsos positivos existe una gran diferencia. Esto es debido a que los autores de este artículo han sobre-entrenado el sistema con respecto a sus datos. Se puede ver en las siguientes figuras:



(a) Resultado del artículo



(b) Resultado del TFG

Figura 5.9: FROC curves

Capítulo 6

Conclusiones

Último capítulo de la memoria, donde se presentan las conclusiones a las que se ha llegado durante la realización del trabajo.

- Se ha realizado un estudio y análisis de las principales técnicas de deep learning para la detección de nódulos pulmonares, poniendo el foco de atención en las más recientes y con mayor proyección, lo que ha permitido observar el gran avance que han supuesto las CNNs para el problema de detección en imágenes médicas (no sólo en TAC torácicos), haciendo posible que con un menor procesado previo de las imágenes y por consiguiente un menor conocimiento específico sobre el problema, una persona con conocimientos de programación y redes de neuronas, pueda obtener resultados, mejores incluso, que aquellos que habían sido obtenidos por expertos en la materia hace una década.

- Se han podido probar técnicas de transfer learning que, para problemas como el abordado, en los que el conjunto de datos es limitado, facilitan el proceso de entrenamiento en gran medida. También el hecho de usar técnicas de data augmentation ayuda considerablemente a mejorar el rendimiento de los modelos. Además, se han observado los grandes requisitos computacionales que son necesarios para el tratamiento de imágenes, sobretodo en la aproximación 3D, ya que de no contar con modelos pre-entrenados en los conjuntos de imágenes que han sido utilizadas, habría sido imposible la realización de esta aproximación con el modelo utilizado.

- La red 2D probada muestra una buena capacidad de generalización al tener una precisión y sensibilidad estables independientemente del tamaño del nódulo. Si bien, un mayor tamaño del nódulo le ayuda a mejorar la precisión con un umbral IoU mayor (detecta un mayor porcentaje del área del bounding-box del nódulo). Además, podemos observar en las gráficas de los resultados obtenidos que el sistema genera un número reducido de falsos positivos, con un buen balance entre precisión y sensibilidad.

- Se han probado diferentes implementaciones actuales para la tarea de detección, con distintas dificultades a la hora de su implementación. Particularmente las aproximaciones 3D requieren de una mayor dificultad técnica. Como ya se ha dicho, estas últimas no se habrían podido reproducir con los recursos hardware que se disponían, ya que la mayoría de estas se han entrenado y probado con 8 Nvidia GTX 1080Ti (88 GB de memoria GPU), mientras que el equipo utilizado contaba con 1 GTX 1070 con 8 GB de memoria. Por lo que entrenar un modelo de cero sería imposible, al dar unos pobres resultados debido entre otras cosas al batch-size que se estaría utilizando. Pero gracias a los pesos pre-entrenados con batch-size adecuados, ha sido posible la utilización de estas redes, aunque no sin dificultades. Para probar el modelo, la GPU no tenía el tamaño de memoria suficiente para cargar el batch de test de una única TC. Por este motivo se tuvo que adaptar el modelo para procesarlo por CPU con una cantidad de RAM de 24 GB + 48 GB de memoria SWAP porque ni 48GB totales (24GB + 24 SWAP) eran suficientes para hacer el test del modelo. Es posible que estos cambios hayan sido los culpables de no alcanzar el rendimiento presentado en el trabajo original, aunque no justifican la gran diferencia entre el número de falsos positivos del modelo del artículo y el expuesto en este TFG, esto es provocado al haber sobre-entrenado el modelo en [61] para que consiga mejores resultados en su conjunto de datos.

- Se deja abierta la posibilidad de continuar con el estudio de estas técnicas, utilizando nuevas redes con el objetivo de crear un sistema CAD completo, en particular, analizar un nuevo tipo de redes como SCPM-Net [50] que es un nuevo tipo de red anchor-free y que ha obtenido los mejores resultados publicados en la detección de nódulos pulmonares en el conjunto de datos LUNA.

- No cabe duda de que los futuros sistemas CAD incluirán técnicas de deep learning, tanto para la detección de nódulos pulmonares como para su posterior clasificación, o incluso para la detección de otras lesiones en la estructura pulmonar. Estamos en pleno ”boom” del deep learning y en los próximos años es más que probable que se creen nuevas redes que mejoren los resultados actuales. Esta tendencia ya es visible en la actualidad y con el gran aumento del número de investigadores en este campo los resultados no harán más que mejorar. La pregunta es, ¿hasta dónde?

Bibliografía

[1] F. Bray, J. Ferlay, I. Soerjomataram, R. Siegel, L. Torre, and A. Jemal, "Global cancer statistics 2018: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries," *CA Cancer Journal for Clinicians*, vol. 68, no. 6, pp. 394–424, Nov. 2018.

[2] "Red española de registros de cáncer (redecan). redecan, estadísticas 2020," 2020. [En línea]. Disponible en: https://redecan.org/redecan.org/es/index.html

[3] V. Zadnik, A. Mihor, S. Tomsic, T. Zagar, N. Bric, K. Lokar, and I. Oblak, "Impact of covid-19 on cancer diagnosis and management in slovenia â€“ preliminary results," *Radiology and Oncology*, vol. 54, no. 3, pp. 329 – 334, 01 Sep. 2020. [En línea]. Disponible en: https://content.sciendo.com/view/journals/raon/54/3/article-p329.xml

[4] I. F. Bush, "Lung nodule detection and classification," 2016.

[5] D. A. Muñoz, "Cáncer. genes y nuevas terapias," 1997.

[6] "Aecc cáncer de pulmón," https://www.aecc.es/es/todo-sobre-cancer/tipos-cancer/cancer-pulmon/diagnostico.

[7] A. oncológica, "Etapas del cáncer." [En línea]. Disponible en: https://ayudaoncologica.wordpress.com/todo-sobre-el-cancer/cancer-de-pulmon/etapas-del-cancer/

[8] cancer.gov. [En línea]. Disponible en: https://www.cancer.gov/espanol/tipos/timoma/paciente/tratamiento-timoma-pdq

[9] A. M. Turing, *Computing Machinery and Intelligence*. Dordrecht: Springer Netherlands, 2009, pp. 23–65. [En línea]. Disponible en: https://doi.org/10.1007/978-1-4020-6710-5_3

[10] W. Mcculloch and W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 127–147, 1943.

[11] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[12] A. B. Novikoff, "On convergence proofs on perceptrons," in *Proceedings of the Symposium on the Mathematical Theory of Automata*, vol. 12. New York, NY, USA: Polytechnic Institute of Brooklyn, 1962, pp. 615–622.

[13] Wikipedia, "Perceptrón — wikipedia, la enciclopedia libre," 2021, [Internet; descargado 7-junio-2021]. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Perceptr%C3%B3n&oldid=133276686

[14] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. [En línea]. Disponible en: http://www.nature.com/articles/323533a0

[16] K. Fukushima, "Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol Cybern*, vol. 36, no. 4, pp. 193–202, 1980.

[17] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, *Efficient BackProp*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 9–50. [En línea]. Disponible en: https://doi.org/10.1007/3-540-49430-8_2

[18] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," 2012.

[19] Q. Yang, "An introduction to transfer learning," in *Advanced Data Mining and Applications*, C. Tang, C. X. Ling, X. Zhou, N. J. Cercone, and X. Li, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–1.

[20] A. S. Qureshi and A. Khan, "Adaptive transfer learning in deep neural networks: Wind power prediction using knowledge transfer from region to region and between different task domains," *Computational Intelligence*, vol. 35, pp. 1088 – 1112, 2019.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on*

Neural Information Processing Systems - Volume 1, ser. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012, p. 1097–1105.

[22] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, ser. NIPS'06. Cambridge, MA, USA: MIT Press, 2006, p. 1137–1144.

[23] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," 01 2010, pp. 92–101.

[24] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, pp. 107–116, 04 1998.

[25] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018.

[26] D. Misra, "Mish: A self regularized non-monotonic activation function," 2020.

[27] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[28] L. Bottou, "Online algorithms and stochastic approximations," Online Learning and Neural Networks. Cambridge University Press. ISBN 978-0-521-65263-6., 1998.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," , vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[30] "Geoffrey hinton neural networks for machine learning nline course." https://www.coursera.org/learn/neural-networks/home/welcome.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[33] R. Girshick, "Fast r-cnn," 2015.

[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[35] H. Kim, T. Nakashima, Y. Itai, S. Maeda, J. Tan, and S. Ishikawa, "Automatic detection of ground glass opacity from the thoracic mdct images by using density features," *2007 International Conference on Control, Automation and Systems*, pp. 1274–1277, 2007.

[36] Y. O. T. T. Teramoto A, Fujita H, "Automated detection of pulmonary nodules in pet/ct images: Ensemble false-positive reduction using a convolutional neural network technique," in *2016*, 2009, pp. 2821–2827.

[37] S. Shimoyama, N. Homma, M. Sakai, T. Ishibashi, and M. Yoshizawa, "Auto-detection of non-isolated pulmonary nodules connected to the chest walls in x-ray ct images," in *2009 ICCAS-SICE*, 2009, pp. 3672–3675.

[38] V. M. P. M. v. G. B. van Rikxoort EM, de Hoop B, "Automatic lung segmentation from thoracic computed tomography scans using a hybrid approach with error detection," in *2009*, 2009, pp. 2934–47.

[39] M. Alilou, V. Kovalev, E. Snezhko, and V. Taimouri, "A comprehensive framework for automatic detection of pulmonary nodules in lung ct images," *Image Analysis Stereology*, vol. 33, no. 1, pp. 13–27, 2014. [En línea]. Disponible en: https://www.ias-iss.org/ojs/IAS/article/view/1081

[40] T. F. B. R. Keshani M, Azimifar Z, "Lung nodule segmentation and recognition using svm classifier and active contour modeling: a complete intelligent system," pp. 287–300, 2013.

[41] A. A. A. Setio, F. Ciompi, G. Litjens, P. Gerke, C. Jacobs, S. J. van Riel, M. M. W. Wille, M. Naqibullah, C. I. Sánchez, and B. van Ginneken, "Pulmonary nodule detection in ct images: False positive reduction using multi-view convolutional networks," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1160–1169, 2016.

[42] Özgün Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: Learning dense volumetric segmentation from sparse annotation," 2016.

[43] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[44] Y. Qin, H. Zheng, Y. Zhu, and J. Yang, "Simultaneous accurate detection of pulmonary nodules and false positive reduction using 3d cnns," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1005–1009, 2018.

[45] J. Ding, A. Li, Z. Hu, and L. Wang, "Accurate pulmonary nodule detection in computed tomography images using deep convolutional neural networks," 2017.

[46] T. T. S. E. d. J. P. K. J. O. M. d. K. H. P. M. S.-P. C. v. G. B. Jacobs C, van Rikxoort EM, "Automatic detection of subsolid pulmonary nodules in thoracic computed tomography images," 2014.

[47] K. Murphy, B. van Ginneken, A. Schilham, B. de Hoop, H. Gietema, and M. Prokop, "A large-scale evaluation of automatic pulmonary nodule detection in chest ct using local image features and k-nearest-neighbour classification," *Medical Image Analysis*, vol. 13, no. 5, pp. 757–770, 2009, includes Special Section on the 12th International Conference on Medical Imaging and Computer Assisted Intervention. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S1361841509000516

[48] G. J. v. G. B. JSetio AA, Jacobs C, "Automatic detection of large pulmonary solid nodules in thoracic ct images," 2015.

[49] J. . Kuhnigk, V. Dicken, L. Bornemann, A. Bakai, D. Wormanns, S. Krass, and H. . Peitgen, "Morphological segmentation and partial volume analysis for volumetry of solid pulmonary lesions in thoracic ct scans," *IEEE Transactions on Medical Imaging*, vol. 25, no. 4, pp. 417–434, 2006.

[50] X. Luo, T. Song, G. Wang, J. Chen, Y. Chen, K. Li, D. N. Metaxas, and S. Zhang, "Scpm-net: An anchor-free 3d lung nodule detection network using sphere representation and center points matching," 2021.

[51] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.

[52] M. Zlocha, Q. Dou, and B. Glocker, "Improving retinanet for ct lesion detection with dense masks from weak recist labels," 2019.

[53] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," 2019.

[54] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," 2019.

[55] W. Zhu, C. Liu, W. Fan, and X. Xie, "Deeplung: Deep 3d dual path nets for automated pulmonary nodule detection and classification," 2018.

[56] T. Song, J. Chen, X. Luo, Y. Huang, X. Liu, N. Huang, Y. Chen, Z. Ye, H. Sheng, S. Zhang *et al.*, "Cpm-net: A 3d center-points matching network for pulmonary nodule detection in ct scans," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2020, pp. 550–559.

[57] E. A. Kazerooni, J. H. Austin, W. C. Black, D. S. Dyer, T. R. Hazelton, A. N. Leung, M. F. McNitt-Gray, R. F. Munden, and S. Pipavath, "ACR-STR practice parameter for the performance and reporting of lung cancer screening thoracic computed tomography (CT): 2014 (Resolution 4)," *J Thorac Imaging*, vol. 29, no. 5, pp. 310–316, Sep 2014.

[58] D. Manos, J. M. Seely, J. Taylor, J. Borgaonkar, H. C. Roberts, and J. R. Mayo, "The Lung Reporting and Data System (LU-RADS): a proposal for computed tomography screening," *Can Assoc Radiol J*, vol. 65, no. 2, pp. 121–134, May 2014.

[59] E. M. van Rikxoort, B. de Hoop, M. A. Viergever, M. Prokop, and B. van Ginneken, "Automatic lung segmentation from thoracic computed tomography scans using a hybrid approach with error detection," *Medical Physics*, vol. 36, no. 7, pp. 2934–2947, 2009. [En línea]. Disponible en: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.3147146

[60] A. Traoré, A. O. Ly, and M. A. Akhloufi, "Evaluating deep learning algorithms in pulmonary nodule detection*," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, 2020, pp. 1335–1338.

[61] Y. Li and Y. Fan, "Deepseed: 3d squeeze-and-excitation encoder-decoder convolutional neural networks for pulmonary nodule detection," in *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. IEEE, 2020, pp. 1866–1869.