UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Descripción y clasificación de radiografías de tórax

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Miguel Feliu Descalzo
Tutor: Francisco Casacuberta Nolla

Curso 2019-2020

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor Francisco Casacuberta la ayuda prestada y la paciencia mostrada durante la realización de este trabajo. También a Álvaro Peris, por ayudarme a empezar y por la resolución de dudas al trabajar con Keras, sobre todo en la parte de clasificación de texto.

Seguidamente, al PRHLT por permitirme utilizar sus máquinas para el entrenamiento de los modelos y a Nvidia por proporcionar las tarjetas gráficas disponibles en esas máquinas.

Finalmente, agradecer a mi familia las oportunidades y el apoyo que me han dado durante todos estos años, también a mis amigos por ayudarme a desconectar cuando más lo necesitaba.

Gracias.

Resum

L'aprenentatje automàtic ha experimentat grans avanços en els últims anys, gràcies a les nombroses investigacions i a la gran quantitat de dades disponibles, propiciant la seua implantació en camps tant crítics com la seguretat viària o la sanitat entre altres, on ha mostrat bons resultats.

En aquest treball es du a terme el desenvolupament de diversos models basats en xarxes neuronals, que exemplifiquen el que es pot aconseguir amb l'aprenentatge de tasques per a assistir als professionals sanitaris en la presa de decisions, com pot ser durant el procés de diagnòstic. En concret, es dispossa de dades compostes per radiografies toràciques y les seues descripcions, a partir dels quals els models duen a terme dues tasques: classificació d'imatges y etiquetatge de text. Addicionalment, es comparà l'acompliment de diferents arquitectures en la realització de les mateixes tasques.

Paraules clau: Aprenentatge automàtic, Imatge mèdica, Xarxes neuronals artificials

Resumen

El aprendizaje automático ha experimentado grandes avances en los últimos años, gracias a las numerosas investigaciones y a la gran cantidad de datos disponibles, propiciando su implantación en campos tan críticos como la seguridad vial o la sanidad entre otros, donde ha mostrado buenos resultados.

En este trabajo se lleva cabo el desarrollo de varios modelos basados en redes neuronales, que ejemplifican lo que se puede conseguir con el aprendizaje de tareas para asistir a los profesionales sanitarios en la toma de decisiones, como puede ser durante el proceso de diagnóstico. En concreto, se dispone de datos compuestos por radiografías torácicas y sus descripciones, a partir de los cuales los modelos llevarán a cabo dos tareas: clasificación de imágenes y etiquetado de texto. Adicionalmente, se comparará el desempeño de distintas arquitecturas en la realización de las mismas tareas.

Palabras clave: Aprendizaje automático, Imagen médica, Redes neuronales artificiales

Abstract

Machine learning has experienced great progess in recent years, thanks to the numerous investigations and the great amount of data available, favouring its implementation in critical fields such as road safety or health among others, where it has shown good results.

In this work, several models based on neural networks are developed, which exemplify what can be achieved with the learning of tasks to assist healthcare professionals in decision making, such as during the diagnostic process. In particular, data is available consisting of chest x-rays and their descriptions, from which the models will carry out two tasks: image classification and text labelling. Additionally, the performance of different architectures in performing the same tasks will be compared.

Key words: Machine Learning, Medical imaging, Artificial neural networks

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

El aprendizaje automático o *Machine Learning* es una rama de la inteligencia artificial que ha experimentado una evolución sustancial en las dos últimas décadas y sigue en continua expansión. Son numerosas las aplicaciones de esta tecnología que usamos día tras día y ni tan siquiera somos conscientes de ello.

Estas aplicaciones están, en su mayoría, basadas en el entrenamiento de redes neuronales artificiales [1] con millones de datos y son capaces de cumplir sus objetivos en un gran porcentaje de situaciones diversas. Su aporte a la sociedad puede no verse de manera tan directa si pensamos, por ejemplo, en la recomendación de contenido de algunas plataformas de vídeo, que utilizan estas tecnologías [1]. Sin embargo, es ya una realidad en campos como la seguridad vial o la sanidad, como es el caso de este trabajo.

Los sistemas de salud pública de todo el mundo dependen en gran medida del soporte tecnológico que se ha ido implantando y ha ido evolucionando a lo largo de los años. Sistemas de citas, monitorización de pacientes, máquinas de diagnóstico... Estas tecnologías están orientadas a asistir a los profesionales sanitarios, mejorar el diagnóstico y tratamiento de los pacientes y, en última instancia, salvar vidas.

En particular, las máquinas de rayos X se utilizan para realizar exámenes no invasivos, generando radiografías que el facultativo evalúa para conocer el estado del paciente. La radiografía de tórax en particular es un tipo de examen muy común, genera imágenes de los pulmones, el corazón y las vías respiratorias entre otras estructuras corporales. Los diagnósticos que se pueden llegar a realizar, gracias al estudio realizado por parte de un radiólogo, van desde neumonía, pasando por insuficiencias cardíacas, hasta cáncer de pulmón. Se ha estado utilizando también en las unidades de cuidados intensivos para observar la evolución de los pacientes ingresados con COVID-19, podemos ver un ejemplo en [34].

De estos exámenes se pueden extraer datos compuestos por imágenes y texto, es decir, radiografías y sus informes radiológicos asociados. A partir de estos datos se pueden construir modelos computacionales basados en aprendizaje automático, y más concretamente en redes neuronales, que asistan al personal sanitario generando potenciales diagnósticos o ayudando a etiquetar los informes radiológicos.

En este trabajo se van a utilizar datos de este tipo, recopilados durante años en la sanidad pública, compuestos por radiografías torácicas de multitud de pacientes y sus informes, para intentar crear varias soluciones enfocadas al aprendizaje automático. En primer lugar trabajaremos con el texto de los informes, centrándonos en el desarrollo de

[1] Por simplicidad a partir de ahora llamaremos "redes neuronales" a las redes neuronales artificiales.

modelos que etiqueten automáticamente estos informes en base a los hallazgos realizados y descritos por el radiólogo. Seguidamente trabajaremos con las imágenes de radiografías torácicas, buscando diferenciar aquellas que puedan generar un diagnóstico anormal de las que muestren un estado normal del paciente; tratando así de mostrar un potencial sistema de ayuda a la decisión, que pueda descongestionar ciertos aspectos del proceso de diagnóstico.

Con todo, el aprendizaje automático puede dar un nuevo enfoque a esos datos recopilados durante décadas; no sólo en la sanidad pública, también en gran variedad de campos, que hasta ahora se utilizaban con fines estadísticos o de visualización y poder realizar predicciones o clasificaciones que puedan ayudar a las personas, como en nuestro ejemplo relacionado con la salud.

1.1 Motivación

La motivación que me ha conducido hasta la realización de este trabajo es querer explorar más en profundidad lo visto durante el grado y en la rama de computación. Concretamente respecto al aprendizaje automático o *machine learning*, centrado en las redes neuronales y sus aplicaciones en situaciones reales, como puede ser la que trata este trabajo.

Este campo ha experimentado un crecimiento sustancial gracias a los avances en investigación en la última década, además de un aumento generalizado de inversión en tecnologías de la información y la comunicación (TIC) en la sanidad pública española en los últimos años.

Como podemos ver en la primera fila de la tabla 1.1, el presupuesto de las comunidades autónomas (CCAA) ha ido aumentando desde 2015, a pesar de que en el último año haya descendido ligeramente. En cuanto a la segunda fila, el Ministerio de Sanidad (MS) ha aumentado la inversión de 2018 a 2019 en el Sistema Nacional de Salud (SNS); quedando la inversión global en TIC, representada en la última fila, en valores similares durante los últimos años. Todo esto anima a desarrollar un trabajo que muestre una pequeña parte de las aplicaciones que se podrían llegar a implementar, directamente relacionadas con el aprendizaje automático, en los servicios de salud.

Inversión global en TIC (en miles de €)	2015	2016	2017	2018	2019	% vari. 2019-2018
PRESUPUESTO TIC DE LAS CCAA	664.628	674.123	695.593	728.831	707.344	-2,95%
INVERSIÓN TIC DEL MS EN EL SNS (*)	13.848	9.853	11.464	5.879	7.224	22,88%
INVERSIÓN TIC DE RED.ES PARA EL SNS (**)	45.600	1.538	8.300	1.000	1.336	33,60%
INVERSIÓN GLOBAL EN TIC (CCAA, MS y RED.ES)	724.076	685.514	715.357	735.710	715.904	-2,69%
(*) Datos suministrados por el Ministerio de Sanidad (**) Datos suministrados por Red.es						

Tabla 1.1: Inversión global en TIC - ÍNDICE SEIS 2019

En concreto, la clasificación de radiografías de tórax mediante una red neuronal puede ayudar a descongestionar ciertas áreas de la sanidad pública y facilitar el trabajo a los radiólogos. Por ejemplo, si se consiguiera un modelo que clasificara las imágenes radiológicas con cierta exactitud, se podría dar prioridad a aquellos pacientes cuyas radiografías han sido clasificadas como anormales. Un ejemplo reciente puede ser el del hospital Vall d'Hebron de Barcelona, donde están tratando de diagnosticar la enfermedad COVID-19 con el uso de redes neuronales convolucionales [37].

Estos sistemas contribuyen a la toma de decisiones y mejoran la calidad de la atención a los pacientes, sirviendo incluso como triaje y permitiendo a los facultativos centrarse en los casos más importantes, reduciendo así el tiempo de diagnóstico y mejorando la precisión.

Por último, supone un reto lidiar con grandes cantidades de datos, quizá no tan idóneos como los conjuntos estándares vistos en algunas asignaturas y permite ver que modelar la realidad es un trabajo ciertamente complejo.

1.2 Objetivos

El objetivo principal de este trabajo es mostrar el desarrollo de varios modelos de aprendizaje automático basados en redes neuronales, que puedan ser útiles para asistencia en el diagnóstico de pacientes y la toma de decisiones y que sirvan como muestra de la gran utilidad que pueden llegar a tener los datos recopilados durante años en la sanidad pública, concretamente los datos de imágenes médicas [10], para entrenar estos modelos. Para ello propondremos y crearemos modelos para dos tareas, una basada en el etiquetado de las descripciones radiológicas y otra para clasificar las imágenes directamente en muestras normales y muestras con algún hallazgo anormal. Se expondrán también los resultados alcanzados con los diversos modelos desarrollados para ambas tareas, comparando las particularidades de cada uno.

Estos modelos y otros similares construidos a partir de los datos anteriormente mencionados, podrían ayudar a mejorar la atención proporcionada a los pacientes, sirviendo como herramientas a los profesionales sanitarios.

Además, contaremos con dos subobjetivos:

- Estudiar el estado del arte en la clasificación de texto y la clasificación de imágenes, basado todo ello en aprendizaje automático con redes neuronales, así como de las aplicaciones punteras que se dan en la actualidad y los tipos de redes que las componen.
- Examinar los datos con los que trabajaremos, exponer su contenido de forma gráfica y seleccionar los subconjuntos que utilizaremos en nuestros modelos para clasificación texto e imagen. También trataremos brevemente la procedencia y obtención de estos datos médicos desde un punto de vista ético.

1.3 Estructura de la memoria

Este documento está estructurado de la siguiente manera:

- Capítulo 2: Estado del arte

En este capítulo exploraremos el estado del arte en el aprendizaje automático, concretamente en la clasificación de texto e imágenes utilizando diversos modelos de redes neuronales, así como las aplicaciones más comunes que tienen este tipo de tecnologías.

Mostraremos también el potencial que tienen las librerías de Python usadas, orientadas a aprendizaje automático.

- Capítulo 3: Redes Neuronales Artificiales

  Seguidamente realizaremos una introducción a las redes neuronales, su trayectoria a lo largo de la historia, la composición básica y cómo se entrenan.

- Capítulo 4: Marco Experimental

  Este capítulo engloba todas las secciones experimentales del trabajo; es decir, el trabajo con los datos utilizados y la implementación de los modelos propuestos, así como de los experimentos realizados con ellos.

  - Conjunto de datos
    Esta primera sección trata los aspectos relacionados con los datos que usaremos durante el trabajo, de qué están compuestos y su cantidad y procedencia, entre otras cuestiones.
  - Implementación
    Aquí explicaremos el proceso de desarrollo de los modelos propuestos y el tratamiento inicial de los datos, partiendo de la base de librerías de aprendizaje automático del lenguaje de programación *Python*.
  - Experimentos
    En esta sección se pretende exponer los experimentos realizados en los procesos de entrenamiento de los modelos, variando los hiper-parámetros y viendo cómo afectan al aprendizaje final.

- Capítulo 5: Resultados

  A continuación, analizaremos los resultados obtenidos para ambos modelos en el capítulo anterior, mostrando gráficas y ejemplos concretos de clasificación.

- Capítulo 6: Conclusiones

  Finalmente, expondremos las conclusiones alcanzadas con este trabajo haciendo alusión a los objetivos iniciales, además de proponer mejoras sobre el trabajo realizado y futuras propuestas.

1.4 Propuesta

En esta sección vamos a proponer la creación de los modelos computacionales basados en redes neuronales que resuelvan los problemas planteados para alcanzar los objetivos propuestos. Estos modelos van a resolver las tareas de clasificación multi-etiqueta, para la parte de texto, y la clasificación binaria para la parte de imágenes.

1.4.1. Etiquetado de descripciones radiológicas

En primer lugar, el problema de etiquetar el texto de las descripciones de las radiografías. Este problema es de tipo *multi-label* o multi-etiqueta, ya que cada muestra puede pertenecer a una o varias etiquetas y estas no son excluyentes entre sí.

Para resolverlo, en base al análisis de los datos, preprocesaremos el texto de cada muestra vectorizándolo para poder pasarlo como entrada a la red. También agruparemos las etiquetas para concentrar la cantidad de muestras de cada una, obteniendo en número total de estas. Seguidamente construiremos la arquitectura de los modelos en base a este número de etiquetas, con la ayuda de varias librerías del *Python*.

Un primer modelo tendrá una arquitectura convolucional que ha mostrado buenos desempeño en tareas de este tipo. El segundo modelo tendrá una arquitectura basada en redes neuronales recurrentes, concretamente en LSTM. Y un tercer modelo C-LSTM, propuesto en [35] que combina las dos arquitecturas anteriores. Ambas arquitecturas las estudiaremos en el capítulo 2.

Finalmente entrenaremos ambos modelos, reservando algunos datos para probar la actuación en muestras nuevas, expondremos los resultados de diversas métricas de forma gráfica y mostraremos varios ejemplos del etiquetado automático de los informes.

1.4.2. Clasificación binaria de radiografías

En cuanto al problema de las imágenes, únicamente tenemos que diferenciar dos clases: muestra normal o muestra anormal. En este caso, nos apoyaremos en las etiquetas asociadas a cada imagen para determinar la pertenencia de cada una a las dos clases de nuestro problema. Filtraremos también las radiografías de los datos, para quedarnos únicamente con aquellas de vista posteroanterior o PA, que son las más comunes. Además, ajustaremos el tamaño de las imágenes para que todas tengan el mismo y poder manejar grandes cantidades al tener menor dimensionalidad.

Seguidamente construiremos el modelo usando redes neuronales convolucionales para extraer las características de cada imagen, de nuevo usando *Python*, con ciertas librerías que mostraremos más adelante.

Por último entrenaremos el modelo con la ayuda de GPUs para agilizar el proceso y lo probaremos con muestras no usadas durante el entrenamiento, analizando los resultados y mostrando algún ejemplo de clasificación concreto.

CAPÍTULO 2

Estado del arte

Como ya hemos indicado previamente, las investigaciones en el campo del aprendizaje automático y concretamente en *Deep Learning* [36] han sido sustanciales en el comienzo del siglo XXI. Concretamente durante la última década han surgido infinidad de artículos sobre este campo, que junto con la gran disponibilidad de grandes cantidades de datos, lo han hecho crecer a un ritmo imparable, destacando el surgimiento de la aceleración mediante GPUs en el proceso de entrenamiento [9].

A continuación, definiremos el aprendizaje automático y analizaremos algunos de los avances más recientes que involucran redes neuronales, enfocándolos a los problemas concretos tratados en este trabajo.

2.1 ¿Qué es el aprendizaje automático?

El aprendizaje automático o *machine learning* es una rama de la inteligencia artificial que se basa en el estudio de algoritmos que permiten a los computadores aprender de forma automática mediante la experiencia [39]. Según el tipo de algoritmo, podemos dividir el aprendizaje automático en tres grupos principales:

- **Aprendizaje supervisado**

  Los algoritmos de aprendizaje supervisado requieren de datos etiquetados, es decir, la entrada junto con la salida correcta. Estos aprenden la salida de nuevas muestras gracias a la optimización de una función, que asocia los valores reales de entrenamiento con su predicción realizada por el modelo, permitiendo generalizar nuevos valores.

  Como es habitual, los algoritmos dependen fuertemente del problema que queramos resolver y en este caso también de los datos de entrenamiento. Algunos de los algoritmos más conocidos son las máquinas de soporte vectorial (SVM) [3], el método k vecinos más cercanos [4] y las redes neuronales, las cuales tratamos con más profundidad en este trabajo.

- **Aprendizaje no supervisado**

  Los datos, a diferencia de los del aprendizaje supervisado, están compuestos por muestras no etiquetadas, por lo que el trabajo se centra en el reconocimiento de patrones entre las muestras. El ejemplo más representativo de este tipo de aprendizaje es el análisis de grupos o *clustering*, que se centra en agrupar una serie de objetos, siendo estos similares entre sí y ciertamente diferentes a los de otros grupos.

El algoritmo K-medias [5] se usa extensamente en este tipo de aprendizaje, donde k es el número de grupos deseados. Se busca el centro de estos grupos y se asigna cada punto al centro más cercano, minimizando la distancia.

- **Aprendizaje por refuerzo** [6]

  Se basa en el aprendizaje mediante prueba y error, es decir, aprendizaje maximizando recompensas o minimizando castigos según el éxito o fracaso de las acciones llevadas a cabo por un agente. Estos problemas se representan generalmente mediante procesos de decisión de Markov.

Las aplicaciones de este tipo de tecnología son numerosas y abarcan cualquier área que pueda obtener series de datos válidas, bien para análisis o para predicción. Desde la agricultura, pasando por las finanzas, hasta el diagnóstico médico, entre otros campos.

A continuación nos centraremos en el aprendizaje supervisado, más concretamente en las redes neuronales y los distintos tipos según la tarea deseada.

2.2 Clasificación de texto

Con el auge de las redes sociales y el contenido publicado por millones de usuarios en ellas, la clasificación de texto es cada vez un tema más interesante para que estas empresas puedan detectar y retirar mensajes no adecuados de manera automática. Podemos poner el ejemplo del *sentiment analysis*, siendo el caso más conocido la detección de discurso de odio en la red social Twitter [11], donde los autores utilizan un modelo basado en redes neuronales para detectar estos mensajes de odio, obteniendo mejores resultados que otros métodos anteriormente utilizados. Estos métodos denominados clásicos extraen características simples del texto y buscan la presencia de ciertas palabras en los mensajes que puedan indicar negatividad como insultos, seguidamente utilizan algoritmos de clasificación como SVM. Un ejemplo lo podemos ver en [7], donde extrajeron las características buscando palabras derogatorias contra ciertos colectivos y transformando los mensajes a vectores para posteriormente clasificarlos.

En cuanto a las arquitecturas de redes neuronales más utilizadas en estas tareas. se encuentran las redes neuronales recurrentes, más concretamente las *Long short-term memory* o LSTM. Contamos también con las redes neuronales convolucionales que, si bien están más extendidas en tareas relacionadas con imágenes, han demostrado gran utilidad al trabajar con texto.

Concretamente las **redes LSTM** admiten entradas de distinta longitud y están especialmente diseñadas para aprender dependencias a largo plazo presentes en los datos. En el capítulo 3 se trata más a fondo este tipo de arquitectura.

En los últimos años este tipo de redes se han utilizado para tareas de procesamiento de lenguaje natural o *natural language processing* (NLP), ya que han mostrado excelentes resultados.

Un buen ejemplo es el trabajo de varios investigadores de Google en el artículo "Exploring the limits of language modeling" [28], en el cual exploran varias opciones para modelar y procesar el lenguaje mediante redes neuronales. Concretamente proponen tres modelos basados en redes LSTM, en dos de los cuales utilizan capas convolucionales para la extracción de características. El conjunto de datos usado en este artículo es el "One Billion Words Benchmark" [29], compuesto por casi un billón de palabras y orientado a experimentación en modelado del lenguaje.

Tras la experimentación con varios modelos y configuraciones se llega a mejorar las métricas conseguidas por el estado del arte en ese conjunto mejorando la perplejidad de 51 a 30, utilizando un modelo con extracción de características convolucional, junto con una combinación de LSTMs a nivel de palabras y nivel de caracteres para generar nuevas secuencias.

En cuanto a clasificación de texto, en [30] utilizan también una arquitectura compuesta por redes convolucionales para extraer las características del texto a partir de los *embeddings* seguidos por redes LSTM. Los experimentos se centran en *sentiment analysis* y clasificación de preguntas en varias categorías, mostrando mejores resultados que modelos convolucionales y recurrentes por separado.

Las **redes neuronales convolucionales (CNN)** generalmente se aplican a reconocimiento de imágenes, vídeo y clasificación de imágenes, como en este trabajo, aunque también se ha mostrado que ofrecen excelentes resultados en tareas de clasificación de texto.

Como hemos visto en el apartado anterior, las redes neuronales convolucionales se han utilizado para la extracción de características del texto, junto con otro tipo de redes para realizar diversas tareas. Aquí nos centraremos en mostrar los resultados usando únicamente CNN en clasificación de texto.

El artículo que destacamos en este caso es [15] de Yoon Kim, donde propone el uso de redes neuronales convolucionales sobre vectores de palabras para la clasificación de frases. Lo realmente interesante de este artículo es la simplicidad del modelo y, a pesar de ello, los buenos resultados que se obtienen, mejorando los resultados del estado del arte en algunas tareas.

El modelo base se compone de una capa de *embedding* seguida por filtros convolucionales aplicados a las palabras, para obtener los mapas de características, seguidos por operaciones de reducción de muestreo sobre estas características para obtener los valores más representativos. Por último el modelo cuenta con una capa totalmente conectada con la función de activación softmax, para obtener las probabilidades de cada etiqueta. Además se añade una capa de *dropout* para regularización en la penúltima capa, que corta algunas conexiones entre neuronas aleatoriamente.

Los experimentos se llevan a cabo sobre 7 conjuntos de datos, compuestos por frases y sus etiquetas. Algunos están formados por reseñas positivas y negativas y otros por preguntas sobre distintas temáticas.

Los cuatro modelos probados son variaciones del modelo base: el primero contempla la inicialización aleatoria de las representaciones vectoriales del texto; para que el propio modelo las entrene, el siguiente utiliza vectores pre-entrenados de *word2vec* [31] estáticos (no entrenables), el penúltimo es similar a este pero permite re-entrenar estos vectores pre-entrenados y por último, un modelo con dos canales de vectores pre-entrenados.

Después de entrenar los modelos, los resultados muestran que aquellos con vectores pre-entrenados tienen un mejor desempeño en general y todavía mejor si se re-entrenan con cada tarea. En cuanto al modelo con dos canales, funciona bien en algunos casos pero el autor asegura que requiere más estudio.

Finalmente el autor concluye que con un modelo convolucional sencillo de una capa se ha alcanzado resultados superiores al estado del arte en 4 de 7 tareas y un buen desempeño general, en gran parte gracias al uso de vectores pre-entrenados para representar las palabras.

2.3 Reconocimiento de imágenes

El reconocimiento o clasificación de imágenes mediante redes neuronales, que entra dentro del campo de la visión por computador, está cada vez más presente en nuestra vida diaria. Podemos encontrar numerosos ejemplos como el desbloqueo facial de nuestro *smartphone*, la búsqueda de imágenes en Google o el reconocimiento de dígitos manuscritos (OCR) [16].

Todo esto es posible gracias a modelos computacionales que, en su mayoría, utilizan **redes neuronales convolucionales**. Como ya hemos indicado anteriormente, este tipo de arquitectura está especialmente enfocada para tratar con imágenes, es decir con datos bidimensionales compuestos por múltiples píxeles.

Este tipo de redes neuronales surgen en 1980 por parte de Kunihiko Fukushima [17], quien se basó en las ideas del procesamiento de la visión en los seres vivos, aunque nombre viene dado por la aplicación en la red de una operación matemática llamada convolución. Más adelante, en 1998, fueron mejoradas utilizando el método de entrenamiento por *backpropagation*, el cuál trataremos en el capítulo 3. Recientemente, en el año 2011, se implementaron utilizando GPUs [18] permitiendo así una computación de los modelos mucho más rápida. A partir de ese momento la accesibilidad a este tipo de redes y a grandes conjuntos de datos ha provocado numerosas investigaciones y avances, convirtiendo al *deep learning* en un campo en constante crecimiento.

Quizá una de las publicaciones más sonadas en la última década, en cuanto a redes neuronales convolucionales, es la referente a AlexNet [32], un modelo surgido en 2012 compuesto por CNN capaz de clasificar imágenes de la base de datos ImageNet [1] con a penas un 15 % de error.

El modelo de AlexNet cuenta con más de 650.000 neuronas, distribuidas en cinco capas convolucionales, algunas seguidas por capas de *max-pooling*, tres capas totalmente conectadas finalizando con otra de este tipo compuesta por 1000 neuronas (tantas como clases) y la función de activación softmax. Las imágenes que alimentan la red están compuestas por 224x224 píxeles y 3 canales de color.

También cabe destacar el uso de la función de activación ReLU en las capas convolucionales, que permite un entrenamiento más rápido en contraste con la anteriormente usada *tanh*, además del entrenamiento con múltiples GPUs.

Más adelante, en 2015, un equipo de investigación de Microsoft desarrolló ResNet [33] superando la puntuación anteriormente conseguida por AlexNet con poco más de un 3 % de error. El mayor modelo de ResNet cuenta con 152 capas, compuestas por convolucionales con distinto número de filtros y una última capa totalmente conectada, que varía en función del problema. La particularidad de esta red son conexiones adicionales llamadas residuales, de ahí su nombre, que conectan capas no consecutivas. Estas conexiones realizan "saltos" de dos o tres capas y ayudan en el entrenamiento de la red al reducir el problema de desvanecimiento del gradiente, que trataremos en el capítulo 3, permitiendo que fluyan directamente sin pasar por funciones de activación.

Relacionado con las radiografías de tórax contamos con ChestNet [24], donde los autores desarrollan un modelo de clasificación de imágenes multi-etiqueta basado en ResNet. Este modelo es capaz de etiquetar las radiografías entre 14 enfermedades distintas, los datos consisten en más de 100.000 imágenes de diversos pacientes, etiquetadas con

[1] www.image-net.org

las enfermedades halladas. En cuanto a los resultados obtenidos, la precisión media de los diagnósticos fue la segunda mejor respecto a otros métodos comparados.

Con todo, hemos visto que la clasificación de imágenes mediante redes neuronales convolucionales sigue avanzando, con propuestas de modelos profundos muy capaces, que realizan las tareas deseadas con buenos resultados.

2.4 Conclusiones

Durante este capítulo se ha presentado el campo del aprendizaje automático y sus inicios, además hemos repasado el estado del arte en cuanto a las distintas arquitecturas de redes neuronales y sus aplicaciones relacionadas con el trabajo.

En primer lugar la clasificación de texto, donde este tipo de aprendizaje juega un papel fundamental en multitud de aplicaciones, ha mostrando resultados realmente prometedores con el uso de modelos simples con pocas capas, a pesar de basarse en arquitecturas más avanzadas como son las LSTM o convolucionales.

Seguidamente, en cuanto a clasificación y reconocimiento de imágenes, se ha estudiado el uso de la arquitectura convolucional, fundamental en cualquier modelo que trabaje con imágenes puesto que el trabajo que realiza extrayendo las características de estas es crucial. Además hemos visto principalmente artículos referidos a modelos profundos como ResNet, que representan perfectamente el concepto de *deep learning*; con multitud de capas que requieren bastante tiempo de entrenamiento y unos datos abundantes pero que muestran una gran utilidad, todo esto gracias en gran parte a las GPUs y su poder computacional.

Finalmente, cabe destacar todos los grandes avances que están surgiendo en el campo del aprendizaje automático día a día, permitiendo nuevas aplicaciones tecnológicas y mejoras a las ya existentes, gracias a las investigaciones abiertas de multitud de científicos alrededor del mundo.

CAPÍTULO 3

Redes Neuronales Artificiales

En este capítulo vamos a centrarnos en este tipo de aprendizaje automático supervisado, explicando los conceptos básicos de las redes neuronales y el proceso de entrenamiento. Junto con los modelos ya explicados en el capítulo anterior llevaremos a cabo las tareas de clasificación de texto e imagen propuestas, siendo lo más adecuado debido a la cantidad de datos que disponemos.

3.1 Introducción

Las redes neuronales son modelos computacionales de aprendizaje supervisado que llevan a cabo tareas partiendo de múltiples ejemplos. La idea está inspirada en el comportamiento humano: se observan estímulos del exterior, se procesan y se actúa según lo observado. Por ejemplo, en una tarea de reconocimiento de dígitos manuscritos, la red genera características para cada dígito cuando procesa los ejemplos etiquetados y es capaz de detectar el dígito correspondiente a una nueva muestra.

La primeras apariciones de lo que podemos considerar como la unidad básica de una red neuronal se remontan al 1958, cuando Frank Rosenblatt desarrolla el algoritmo perceptrón [2], que podemos resumir como un clasificador lineal binario. La investigación se frenó a partir de 1969, cuando se probó que el perceptrón era incapaz de procesar funciones no lineales [8] y además los computadores no tenían suficiente potencia para procesar varias capas de perceptrón.

Seguidamente, en 1985 se consolidó el concepto de *backpropagation*, que permitió entrenar redes con múltiples capas y que trataremos en profundidad más adelante. Además, con el paso de los años, han surgido mejoras en la capacidad de cómputo de los ordenadores que han permitido un avance sustancial en las investigaciones. Por ejemplo, en 2009 se demostró que el uso de GPUs en el entrenamiento de redes neuronales [9] suponía grandes mejoras en los tiempos de cómputo.

3.2 Modelo básico y entrenamiento

Las redes neuronales más básicas se componen de una capa de entrada, una o varias capas ocultas y otra de salida. Estas, a su vez, se componen de varias neuronas, en este caso totalmente conectadas con las de la siguiente capa, como podemos observar en la figura 3.1. Todo esto compone un modelo totalmente conectado o modelo *fully connected*.

Cada capa cuenta con un conjunto de pesos w y valores de sesgo o *bias* b, estos pesos y valores se inicializan al crear la red, normalmente de manera aleatoria siguiendo alguna distribución. Estos pesos se utilizan para computar la función de cada capa, con los datos de entrada de esta. Para el caso de la primera capa tendríamos

$$s_1 = w_1 \cdot x + b_1$$

siendo w_1 y b_1 el conjunto de pesos y *bias* de la primera capa oculta.

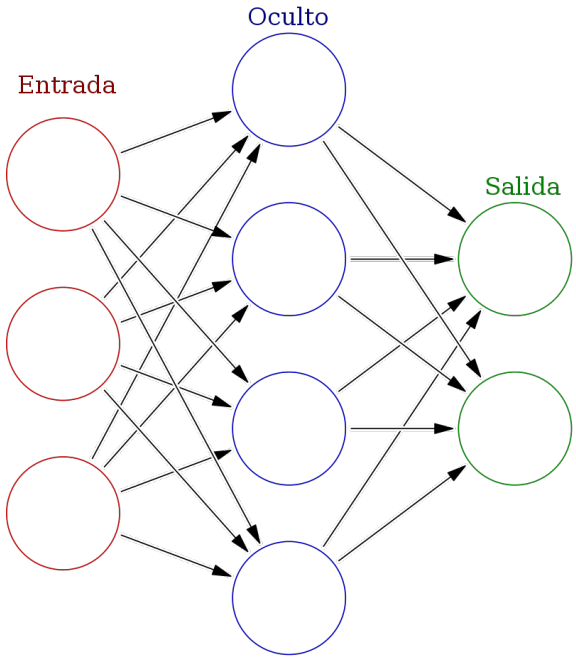


Figura 3.1: Red neuronal totalmente conectada con una capa oculta de cuatro nodos

Además, las redes neuronales cuentan con funciones de activación σ para cada neurona, que normalizan el valor de salida. Siguiendo con el ejemplo de la primera capa

$$a_1 = \sigma(s_1)$$

donde a_1 es el valor decidido por la función de activación según haya alcanzado el umbral definido y, por lo tanto, si debe propagar la información a la siguiente capa.

Las funciones de activación más usadas las podemos ver en la figura 3.2.

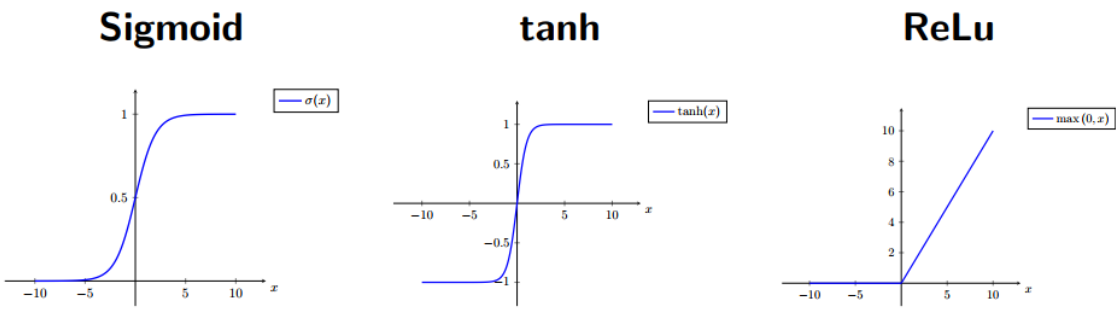


Figura 3.2: Funciones de activación más comunes

Por ejemplo, la función Sigmoid toma como entrada un número real y devuelve un valor entre 0 y 1. La selección de la función de activación también es un aspecto importante a tener en cuenta cuando se construye una red neuronal, ya que pueden afectar en gran medida al proceso de entrenamiento de la red.

El proceso de entrenamiento se compone de dos fases. La primera se denomina **propagación hacia adelante** o *forward propagation*, en este paso se comienza aplicando los datos de entrada a la función de la primera capa. Seguidamente se aplica la función de activación, cuya salida alimenta a la función de la siguiente capa.

Así se suceden estas aplicaciones hasta alcanzar la capa de salida, donde se obtiene la predicción de la red para cada muestra, que no es más que el valor resultante de la función de activación de esta capa. En este paso se calcula la **función de pérdida** $\mathcal{L}$, que la red debe minimizar, entre el valor de salida establecido por la red y el valor verdadero de la muestra. La elección de esta función dependerá del problema que pretendamos resolver, en tareas de clasificación donde las predicciones son probabilidades, se usa la entropía cruzada o *cross-entropy* entre el valor real y la predicción [36].

La segunda fase es la **propagación hacia atrás** o *backpropagation*, aquí se calcula el gradiente de la función de coste respecto de los pesos que componen la red, desde la capa de salida hasta la primera capa. Seguidamente estos pesos se actualizan con los gradientes calculados de la siguiente manera

$$w = w - \eta \cdot \nabla_w \mathcal{L}(w)$$

donde w son los pesos del modelo y $\nabla_w \mathcal{L}(w)$ el gradiente de la función de pérdida respecto de esos pesos. También destacamos η que representa el **factor de aprendizaje** de la red o *learning rate*.

Este proceso de entrenamiento se aplica en un método iterativo denominado **descenso de gradiente estocástico** o *stochastic gradient descent*, donde se realiza el proceso descrito pero en mini-lotes o **mini-batches** de muestras aleatorias. Esto permite procesar varias muestras al mismo tiempo y encontrar un equilibrio entre el tiempo de entrenamiento y la aleatoriedad de los lotes. Generalmente se necesitan varias iteraciones sobre todos los datos de entrenamiento, denominadas *epoch*.

3.3 Algoritmos de optimización

Los algoritmos de optimización u optimizadores [41] son una parte importante del proceso de entrenamiento de las redes neuronales. Estos son los encargados de ajustar los parámetros de la red para tratar de minimizar la función de coste con cada iteración.

El algoritmo básico de descenso de gradiente supone algunos problemas como el valor del factor de aprendizaje, aplicar distintos factores de aprendizaje según los parámetros o evitar puntos *saddle* o silla de la función, en los cuales la pendiente es cero pero no son un extremo local.

Existen diversos algoritmos basados en el descenso de gradiente que tratan de poner solución a algunos de estos inconvenientes, a continuación estudiaremos los más relevantes:

- **Descenso de gradiente estocástico con *momentum* [25]**

  Esta variante añade un término de *momentum* o velocidad v que se va sumando durante el entrenamiento:

$$v_{t+1} = \rho \cdot v_t + \nabla_w \mathcal{L}(w)$$

$$w = w - \eta \cdot v_{t+1}$$

donde podemos considerar ρ como una especia de fricción que ralentiza mínimamente la velocidad. Con la adición del *momentum* se evitan en parte los puntos *saddle*, al depender la actualización de los pesos de la velocidad que se ha ido acumulando.

- **RMSProp** [38]

  Este algoritmo es una variante del algoritmo AdaGrad [26], que mantiene una suma de los gradientes al cuadrado, dividiendo el gradiente actual entre la raíz cuadrada de esta suma, antes de realizar la actualización de los pesos, para que esta sea proporcional.

  El problema de AdaGrad es que el valor de la suma crece bastante cuando el entrenamiento tarda, provocando que las actualizaciones de los parámetros sean mínimas. El algoritmo RMSProp añade un término α que hace que el valor de la suma de los gradientes decaiga durante el entrenamiento, resolviendo así el problema.

- **Adam**

  El algoritmo Adam [27] combina el *momentum* visto anteriormente con la suma de gradientes de RMSProp para realizar actualizaciones proporcionales. El algoritmo es similar a los dos anteriores:

$$m_0 = 0, v_0 = 0$$

$$m_{t+1} = \beta_1 \cdot m_t + (1 - \beta_1) \cdot \nabla_w \mathcal{L}(w)$$
$$v_{t+1} = \beta_2 \cdot v_t + (1 - \beta_2) \cdot \nabla_w \mathcal{L}(w)^2$$
$$w = w - \frac{\eta}{\sqrt{v_{t+1}} + \epsilon} \cdot m_{t+1}$$

  En primer lugar tenemos el *momentum* m y la suma de los gradientes al cuadrado v se inicializan a cero, seguidamente observamos las actualizaciones de estos parámetros acompañadas por los factores β_1 y β_2 que evitan el crecimiento descontrolado. Estos factores se inicializan a 0.9 y 0.999 respectivamente. Finalmente la actualización de los pesos se realiza dividiendo el *momentum* m entre la raíz de la suma v, más un valor ϵ para evitar la división entre cero.

Además algunos de estos algoritmos adaptan el *learning rate* durante el proceso de entrenamiento, este se inicializa con valores entre 0.01 y 0.0001, aunque generalmente depende del problema en particular y conviene ajustarlo.

3.4 Arquitecturas más avanzadas

En esta sección explicaremos el funcionamiento de las arquitecturas avanzadas de redes neuronales, que serán usadas durante el trabajo en tareas de clasificación de texto e imágenes y que hemos tratado en el estado del arte: la arquitectura recurrente (RNN) destacando el modelo Long short-term memory (LSTM) y la arquitectura convolucional.

3.4.1. Redes Neuronales Recurrentes (RNN)

Las redes neuronales recurrentes o RNN son un tipo especial de redes neuronales artificiales, teniendo como principal atractivo frente a las arquitecturas convencionales, la capacidad de procesar secuencias de datos temporales, gracias a las conexiones de retroalimentación 3.3. Estas conexiones permiten que la información previamente procesada por la red persista durante futuras iteraciones, utilizando las salidas previas como entradas, creando así la capacidad de aprender dependencias presentes en los datos.

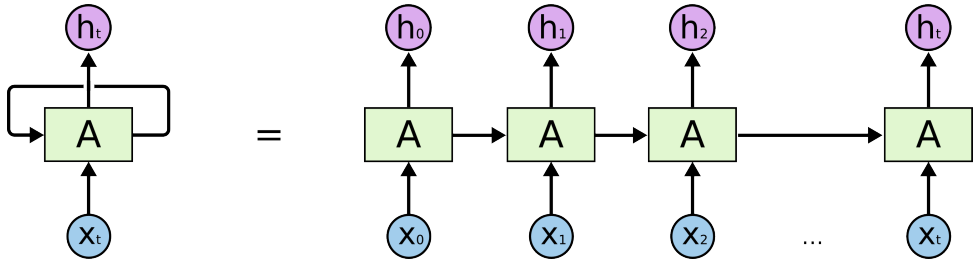


Figura 3.3: Desarrollo de una red neuronal recurrente. **Fuente:** Christopher Olah, colah.github.io

Alguna de las ventajas más destacadas de las RNN son la capacidad de admitir datos de entrada de longitud variable, como es el caso de las frases al trabajar con texto y la compartición de pesos lo largo del tiempo, que permite la característica anterior y además evita que el modelo crezca en tamaño con la entrada. Por otra parte, la computación y el entrenamiento de estos modelos suele ser más lento y el acceso a la información pasada presenta dificultades.

Para cada instante podemos representar la salida h y la retroalimentación a de la siguiente manera

$$h_t = \sigma_1(w_{ha} \cdot a_{t-1} + b_h)$$

$$a_t = \sigma_2(w_{aa} \cdot a_{t-1} + w_{ax} \cdot x_t + b_a)$$

donde w_{ha}, w_{aa}, w_{ax}, b_h y b_a son pesos compartidos temporalmente y σ_1 y σ_2 las funciones de activación.

Durante el entrenamiento de este tipo de redes también puede surgir el problema del desvanecimiento o la explosión de los gradientes [14], causado por la dificultad de capturar dependencias a largo plazo. Una solución que se aplica en la práctica es el recorte del gradiente o *gradient clipping*, que establece unos valores máximos y mínimos para los gradientes previniendo así que la red no aprenda.

Long short-term memory (LSTM)

En la práctica, las dependencias a largo plazo no pueden ser aprendidas por las RNNs convencionales, únicamente las dependencias a corto plazo. Para resolver este problema surgen las Long short-term memory o LSTM en 1997 [13], que se han ido popularizando rápidamente durante los últimos años.

Las LSTM son un tipo de redes neuronales recurrentes, que están especialmente diseñadas para resolver el problema de la dependencia a largo plazo. Los módulos de este tipo de redes neuronales están compuestos internamente por cuatro puertaso *gates*, que interactúan entre sí y con los datos de entrada, asumiendo diversos roles.

En la figura 3.4 podemos ver el esquema interno de un módulo LSTM, las entradas están formadas por el *cell state* c; por donde fluye la información con pocas alteraciones, el estado oculto a y la entrada de información actual x. Contamos también con el estado $\tilde{c}$ que junto con $c^{<t-1>}$ produce el nuevo estado $c^{<t>}$.

Las cuatro puertas vienen denotadas por Γ, seguida por una letra que indica su función. En primer lugar la *forget gate* Γ_f, que decide qué cantidad de información de c pasa a través del módulo. Seguidamente la puerta de actualización o *update gate* Γ_u, que decide la cantidad de información del pasado que debe importar en este instante t. La puerta de relevancia Γ_r, que tiene el rol de eliminar información previa no relevante. Finalmente la *output gate* o puerta de salida, que decide la salida basándose en el *cell state* c.

Estas puertas realmente representan funciones de la forma

$$\Gamma = \sigma(Wx^{<t>} + Ua^{<t-1>} + b)$$

donde σ es la función sigmoid y W, U y b son pesos específicos de cada puerta.

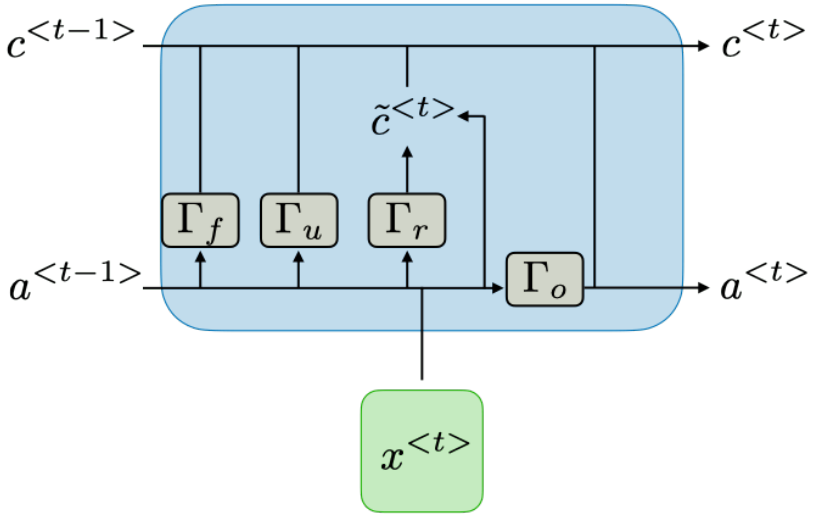


Figura 3.4: Esquema interno de una red neuronal LSTM. **Fuente:** Afshine Amidi; Shervine Amidi, stanford.edu/~shervine/teaching

Además, este tipo de arquitectura resuelve, gracias a las puertas y sus interacciones, el problema de desvanecimiento del gradiente mencionado anteriormente.

3.4.2. Redes Neuronales Convolucionales (CNN)

Este tipo de redes aplican operaciones de convolución a los datos de entrada para extraer las características más relevantes. Generalmente están orientadas al tratamiento de imágenes pero también se usan en tareas relacionadas con texto, como hemos visto en el capítulo anterior.

CNN aplicadas a texto

El proceso que llevan a cabo este tipo de redes en datos bidimensionales se puede utilizar en secuencias de datos unidimensionales, como por ejemplo cadenas de texto. Este proceso es el encargado de extraer las características de las representaciones vectoriales

de cada palabra del texto, para luego pasarlas a las capas totalmente conectadas que se encargan de la clasificación.

Las características se extraen gracias a una ventana deslizante que se desplaza entre varias palabras, computando así estos vectores de características. En la figura 3.5, perteneciente al artículo [15], podemos ver una red neuronal simple que utiliza una capa convolucional con varios filtros, para extraer los vectores de características de la representación matricial del texto.

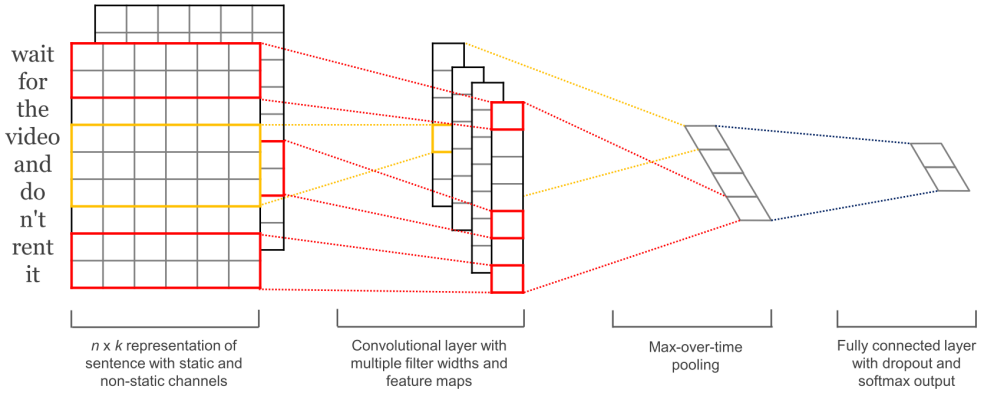


Figura 3.5: Extracción de características mediante una red neuronal convolucional. Podemos ver que la frase "wait for the video and don't rent i" queda representada como una matriz sobre la cual se aplican las operaciones de convolución. **Fuente:**[15]

Los filtros convolucionales, en este caso unidimensionales, se aplican sobre varias palabras al mismo tiempo, permitiendo la detección de n-gramas para los cuales se especializa cada filtro. Seguidamente la capa de *max-pooling* extrae los más relevantes [19]. Por último el resto de la red clasifica el texto en base a esta información extraída.

CNN aplicadas a imágenes

Las aplicaciones más comunes para las redes neuronales convolucionales se dan en tareas relacionadas con imágenes, donde es necesario extraer sus características debido a la dimensionalidad y variabilidad de las imágenes y los objetos que representan.

Los modelos convolucionales reciben como entrada imágenes, representadas en matrices compuestas por sus píxeles. Si las imágenes son en color, las matrices son tridimensionales y cuentan con tres canales uno por cada color en la representación RGB. Estas matrices pasan por los procesos de la red, que resultan en la obtención de las características para que las capas finales puedan realizar las tareas deseadas de clasificación.

La arquitectura de una red neuronal convolucional está generalmente formada por varios bloques que procesan las imágenes y reducen su tamaño, siendo el principal la **capa convolucional**. Esta capa aplica la operación de convolución a la matriz de entrada mediante un filtro o *kernel*, obteniendo como salida un **mapa de características** o *feature map*.

La operación de convolución se representa mediante un asterisco $*$, podemos aplicarla a una entrada x con un filtro f de la siguiente manera

$$s = x * f$$

obteniendo las caractrerísticas de la entrada.

En la figura 3.6 podemos observar un ejemplo sencillo de la operación que lleva a acabo esta capa, el *input* o entrada es una matriz bidimensional a la cual se le aplica el filtro de tamaño 2x2 con desplazamiento igual a una unidad. El resultado es el mapa de características de la entrada, en el primer elemento del mismo observamos la operación aplicada gracias a las flechas, la suma de los valores de la entrada multiplicados cada uno por el valor correspondiente del filtro. Seguidamente este se desplaza, en este caso una unidad hacia la derecha, para seguir computando la convolución. Observamos también que el tamaño de la salida queda reducido, disminuyendo así el número de parámetros para la siguiente capa.

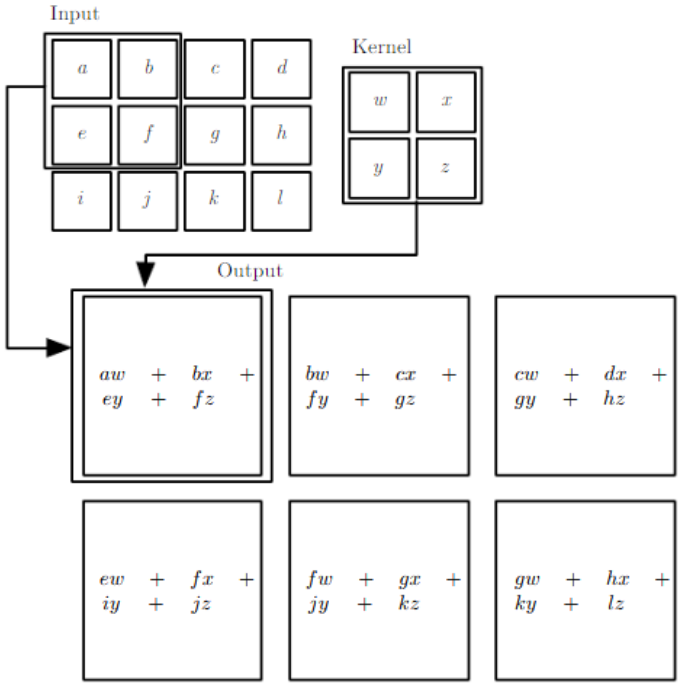


Figura 3.6: Ejemplo de convolución bidimensional. **Fuente:** [36]

Otro bloque importante dentro de la arquitectura convolucional es la capa de **reducción de muestreo** o *pooling*, esta capa reduce el tamaño de la entrada preservando las características más relevantes según el filtro aplicado. La capa más común es la *max-pooling*, que calcula el máximo de los valores sobre los que se desplaza la ventana, generalmente de tamaño 2 x 2, conservando y resumiendo así las características de la entrada en menos parámetros. Con ese tamaño de ventana, la operación toma cuatro píxeles en cada desplazamiento reduciendo la salida a la mitad.

En cualquier caso, esta operación ayuda a preservar la invarianza de la representación obtenida con la capa convolucional ante pequeños cambios en la entrada, como puede ser la rotación de una imagen, mejorando así la eficiencia estadística de la red.

Otro tipo de capa de *pooling* también interesante es la *average pooling*, que calcula la media de los píxeles de la ventana, existiendo también la variante que calcula la media ponderada según la distancia de los píxeles a uno central.

Finalmente, antes de las capas totalmente conectadas que realizan la clasificación, debemos añadir una capa de aplanamiento o *flatten*, esta capa no realiza operaciones de convolución pero es usada en la arquitectura. La operación de aplanamiento transforma

los datos de entrada de tamaño N x M a una matriz de tamaño $(N \cdot M)$ x 1, para ser tratados posteriormente por estas capas finales.

3.5 Conclusiones

Como hemos visto, las redes neuronales son modelos computacionales realmente potentes y cada vez más utilizados, pero hay algunos aspectos que cabe comentar.

El primero es la complejidad, aunque cada vez es más sencillo crear modelos de redes neuronales, estas no dejan de ser complejas y requieren ciertos conocimientos previos. Muchas veces, sobre todo si no poseemos gran cantidad de datos, es mejor considerar otros algoritmos, en algunas ocasiones igual de potentes, como por ejemplo las máquinas de soporte vectorial.

Continuando con el tema de los datos, para el proceso de entrenamiento hace falta contar con una cantidad relevante, además de unos mínimos de calidad. Si no se ajusta la complejidad de los modelos a la cantidad de datos que poseemos, se puede producir un sobreajuste u *overfitting*, esto ocurre cuando el modelo entrenado no puede generalizar correctamente nuevas muestras. Para evitar este problema se debe cuidar el proceso de entrenamiento, no realizando demasiadas iteraciones sobre los datos o utilizando técnicas de regularización.

Además, las redes neuronales requieren de cierta potencia computacional para ser entrenadas y manejar esas grandes cantidades de datos. Si bien es cierto que los modelos más simples o con datos no tan pesados, como el texto, se pueden entrenar en casi cualquier equipo, otros más elaborados que pueden llegar a tener cientos de capas requieren de equipos con varias GPUs y muchas horas de cómputo. Hoy en día estos equipos están al alcance de cualquiera que pueda costearlos, ya que existen plataformas de computación en la nube como Google Cloud o Amazon Web Sevices que ofrecen servicios de aprendizaje automático.

Por último, una cuestión importante en los modelos funcionales compuestos por redes neuronales es la explicabilidad de los mismos. Es decir, por qué toman las decisiones que toman, lo cual está directamente relacionado con la confianza de los usuarios en el modelo. En algunas aplicaciones, como por ejemplo las financieras o médicas es muy importante justificar las decisiones tomadas por los modelos. Esta falta de confianza, entre otros factores, puede estar dificultando la implantación de estas tecnologías, aunque ya existen algoritmos [20] que pueden explicar las decisiones de algunos clasificadores.

CAPÍTULO 4

Marco experimental

En este capítulo vamos a tratar toda la parte experimental del trabajo, centrándonos en la creación de los modelos de etiquetado de texto compuesto por descripciones de radiografías de tórax y de clasificación de estas radiografías mediante redes neuronales. También veremos cómo ha sido el proceso de entrenamiento de los modelos con los datos disponibles y su consiguiente experimentación para la obtención de resultados. Aplicaremos algunos conocimientos expuestos durante el estudio del estado del arte en el capítulo 3.

En primer lugar se analizará el conjunto de datos con el que se va a trabajar, describiremos de qué tratan los datos, cuántos hay, qué información los compone y su procedencia. Además, al ser datos médicos, trataremos su obtención desde un punto de vista más ético.

Seguidamente expondremos algunas librerías interesantes del lenguaje de programación Python para trabajar con redes neuronales, que facilitan el trabajo de desarrollo de los modelos y su entrenamiento. Además explicaremos brevemente el hardware del que hemos dispuesto durante el trabajo, junto con el conjunto de software usado para la implementación, entrenamiento y pruebas de los modelos. A continuación se detallará la implementación de varios modelos, explicando la función y características de cada capa en relación a los datos de entrada y la salida deseada, además de mostrar de forma gráfica su arquitectura.

Finalmente, en la sección de experimentos, entrenaremos los modelos probando varias arquitecturas y parámetros para ver cuáles producen resultados más prometedores, mostrando algunos ejemplos de aprendizaje de forma gráfica.

4.1 Conjunto de datos: PadChest

Las radiografías de tórax son un método muy común de diagnóstico de problemas pulmonares y cardiovasculares, entre otros. Una interpretación correcta de las imágenes radiológicas por parte del personal médico con años de experiencia, concretamente de los radiólogos, es fundamental para el correcto diagnóstico del paciente. Los modelos que vamos a entrenar con estos datos pueden servir como muestra de que los sistemas de ayuda al análisis de las radiografías torácicas pueden asistir satisfactoriamente a los profesionales médicos.

El conjunto de datos que vamos a usar para entrenar y validar nuestros modelos es PadChest [21], un conjunto compuesto por imágenes de radiografías torácicas de pacientes del Hospital San Juan de Alicante, con sus correspondientes descripciones etiquetadas.

La información completa sobre los datos se encuentra accesible en la página web del Banco digital de Imagen Médica de la Comunidad Valenciana (BIMCV) [1].

4.1.1. Descripción

Como ya se ha indicado previamente, PadChest está compuesto por imágenes de alta resolución de radiografías de tórax junto con sus informes en castellano. Las más de 160.000 imágenes, de más de 67.000 pacientes, se obtuvieron desde el año 2009 al año 2017. Para cada imagen contamos con numerosos campos que las acompañan, estos se pueden dividir en dos tipos:

- Datos técnicos sobre las propias imágenes y su obtención, como la resolución, la exposición, la vista e incluso información sobre el fabricante de la máquina de rayos x usada. El sexo del paciente también está incluido en este tipo. Estos datos usan el estándar DICOM [2] para información sobre imágenes médicas.

- Información que complementa las imágenes y enriquece el conjunto de datos, como los identificadores de cada imagen, estudio, paciente e informe. Dentro de este tipo también se encuentran los datos de texto plano de los informes, las etiquetas correspondientes a estos informes, el método usado para su obtención y la proyección de las radiografías, entre otros campos.

En este trabajo haremos uso de los campos de informes escritos, con sus etiquetas, para la parte de clasificación de texto y las imágenes para su clasificación, según los informes, en normales o no normales.

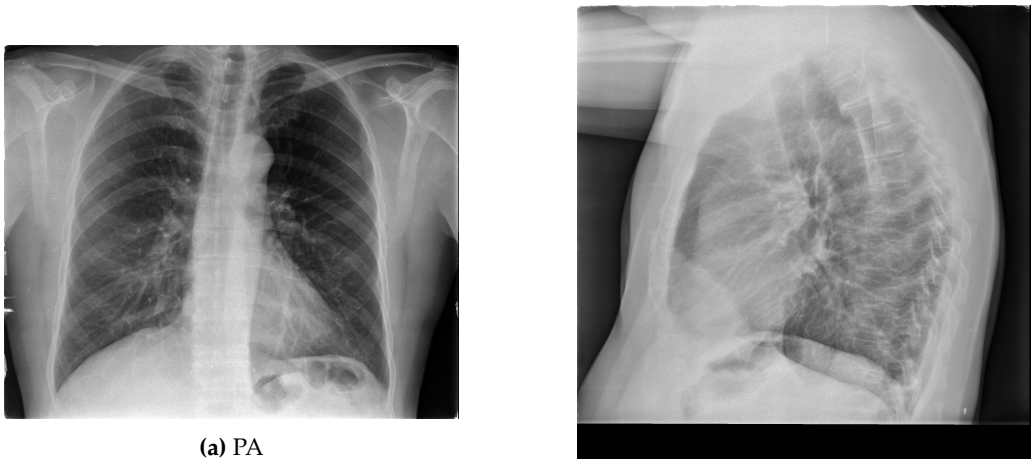
Al solicitar el acceso a los datos mediante un formulario en la página web del BIMCV, obtenemos un enlace para descargarlos. El conjunto cuenta con un fichero de valores separados por comas (CSV) con toda la información referente a cada imagen, una por línea. Las imágenes se descargan por partes, están divididas en 50 directorios debido al gran tamaño que ocupan.

A continuación, en la figura 4.1, podemos ver un ejemplo de dos radiografías que pertenecen al mismo estudio y paciente, un varón de 55 años. La imagen a) es una radiografía en vista posteroanterior (PA), la más común, en este tipo de proyección los rayos X

[1] bimcv.cipf.es/bimcv-projects/padchest
[2] www.dicomstandard.org

penetran por la espalda del paciente, saliendo por la parte delantera del tórax. Mientras que la vista de la radiografía b) es lateral (L).

El texto descriptivo asociado al estudio radiológico es el siguiente: "*sign enfisem pulmon con bull lsi . aument tram broncovascul maner bilateral probabl relacion con proces infecci*". Nótese que el texto ha pasado por un proceso de *stemming*, ya incluido en el conjunto de datos. Las etiquetas, traducidas al castellano, asociadas al informe son: ”marcas broncovasculares”, ”enfisema”, ”neumonía” y ”bulas”.



(a) PA

(b) L

Figura 4.1: Ejemplo de radiografías incluidas en el conjunto de datos. La imagen a) muestra un radiografía en proyección posteroanterior. La proyección de la imagen b) es lateral. Las etiquetas asociadas al estudio radiológico de las imágenes son ['bronchovascular markings', 'emphysema', 'pneumonia', 'bullas'].

4.1.2. Cantidad y calidad

El conjunto de datos cuenta con exactamente 160.861 imágenes, para las cuales existen un total de 107.783 informes de un total de 67.625 pacientes. En la creación del conjunto de datos, el etiquetado de los informes de las imágenes se llevó a cabo de dos maneras. La primera fue el etiquetado a mano de 22.120 frases únicas, por parte de especialistas médicos. Seguidamente se entrenó un modelo compuesto por redes neuronales recurrentes, con estos datos etiquetados a mano, para etiquetar automáticamente el resto de muestras.

Los números totales los podemos ver desglosados en la tabla 4.1, aquí distinguimos entre pacientes, informes e imágenes distintas, según sus informes hayan sido etiquetados manual o automáticamente.

	Pacientes	**Informes**	**Imágenes**
Informe etiquetado manualmente	24.491	26.412	39.053
Informe etiquetado automáticamente	43.134	81.371	121.808
Conjunto etiquetado final	67.625	107.783	160.861

Tabla 4.1: Estadísticas globales del conjunto de datos PadChest.

En total el conjunto cuenta con 193 etiquetas diferentes y cada muestra cuenta con hasta 20 etiquetas, siendo la media de 2.07 etiquetas por muestra.

Las etiquetas asociadas a cada informe e imagen tienen un organización y jerarquía específicas. Las podemos dividir en dos tipos:

- **Hallazgos radiológicos (*Radiological findings*)**

  Este tipo de etiquetas pertenecen a hallazgos o diagnósticos que "los radiólogos pueden afirmar basándose únicamente en la interpretación de la radiografía de tórax sin ningún conocimiento adicional sobre la información clínica del paciente". Por ejemplo, aquellos que afectan a los huesos como fracturas o enfermedades degenerativas, o los que suponen alteraciones en la distribución del aire en los pulmones.

- **Diagnóstico diferencial (*Differential diagnosis*)**

  Los diagnósticos, al contrario que los hallazgos radiológicos, solo se pueden proponer por parte de los radiólogos como una lista de posibles diagnósticos diferenciales, según la información clínica completa del paciente junto con estudios complementarios. Por ejemplo, un hallazgo radiológico puede significar una larga lista de diagnósticos diferenciales; pero si además el paciente presenta otros síntomas, se puede acotar esta lista y realizar un diagnóstico fiable.

Además, el conjunto de datos incluye tres árboles jerárquicos de etiquetas. El principal propósito de estos es ayudar con las tareas de explotación de las imágenes que estén pensadas para niveles jerárquicos superiores. El contenido y la jerarquía de los árboles fueron revisados por un radiólogo con 20 años de experiencia. Las relaciones padre-hijo presentes en estos árboles indican que una entidad hijo ”es una” entidad padre, por ejemplo una ”fractura de clavícula” es una ”fractura”.

Los tres árboles son: hallazgos radiológicos, diagnósticos diferenciales y localizaciones. El último incluye términos anatómicos y espaciales donde se encuentran los hallazgos radiológicos y los diagnósticos. En el caso de este trabajo, estos árboles han sido especialmente útiles para agrupar las etiquetas con menos apariciones en términos más generales.

Etiquetado automático de los informes

Para la creación del conjunto de datos, los autores obtuvieron más de 22.000 informes etiquetados a mano. El resto, fueron etiquetados utilizando un modelo de redes neuronales recurrentes con mecanismo de atención entrenado con estos informes.

Durante este proceso, los autores propusieron cuatro modelos: uno convolucional (CNN), uno recurrente (RNN), convolucional con mecanismo de atención (CNN-ATT) y uno recurrente con mecanismo de atención (RNN-ATT). El entrenamiento de los modelos se produjo con 20.439 frases, mientas que el conjunto de validación contaba con 2.271 frases. Para este conjunto, los resultados obtenidos fueron mejores en el modelo RNN-ATT, obteniendo una precisión de un 86,4 % y una puntuación MicroF1 de 0.924.

Finalmente todos los modelos fueron probados con un conjunto de test independiente, formado por 500 muestras aleatorias pertenecientes al periodo 2009-2013, mientras que las particiones de entrenamiento y validación pertenecían al periodo 2014-2017. El mejor modelo fue de nuevo el RNN-ATT con una puntuación MicroF1 de 0.93.

En la figura 4.2 observamos las estadísticas del número de informes asociados a los hallazgos radiológicos más comunes, parte han sido etiquetados de manera manual y la mayoría de manera automática.

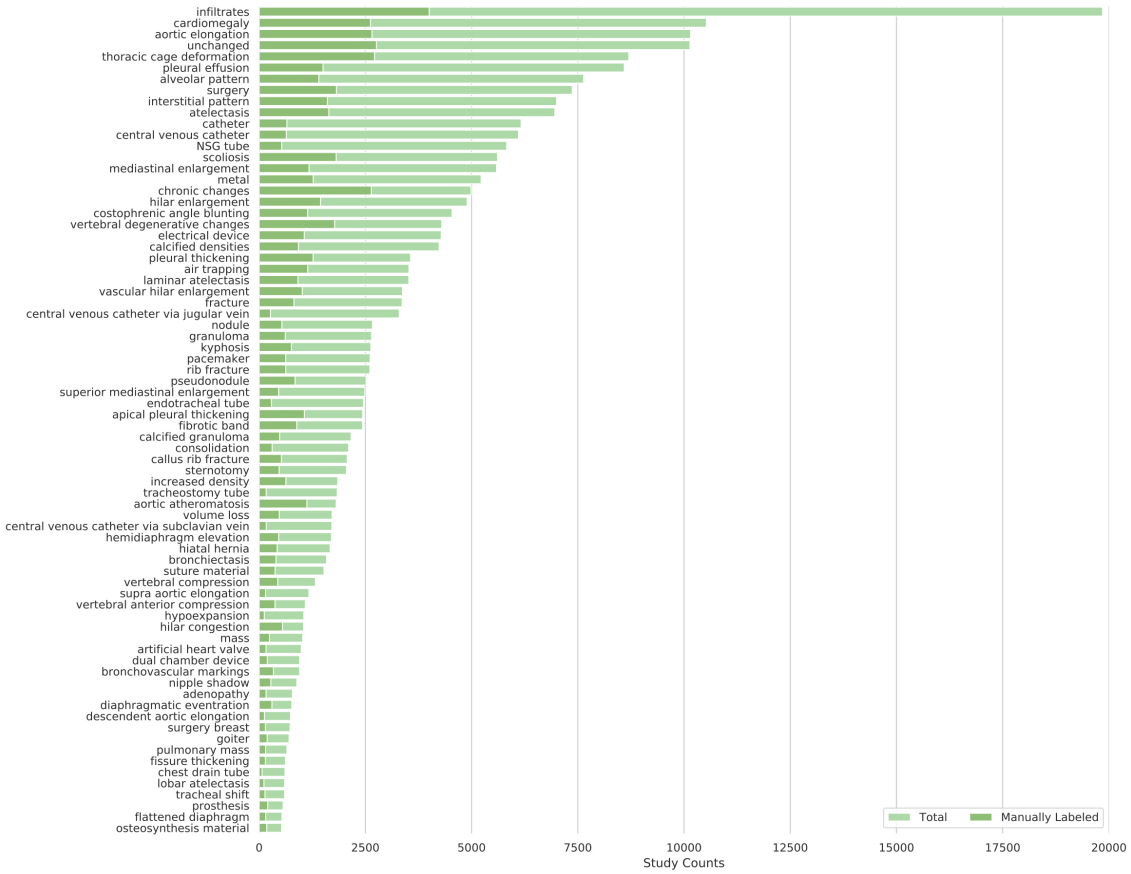


Figura 4.2: Hallazgos radiológicos más comunes. En color oscuro se muestra el número etiquetado manualmente, en color claro el etiquetado de manera automática. **Fuente:** [21]

Preprocesado de texto

El conjunto de datos contiene un informe para cada estudio, si un estudio está formado por varias imágenes con distintas proyecciones, el texto del informe es el mismo para todas. Debido a los distintos formatos en los que se escribieron los informes y la desestructuración al tratarse de texto libre, fue necesario aplicar una serie de procesos para eliminar información no relevante y compactar el texto. Los procedimientos aplicados fueron:

1. Se transformó el texto a minúsculas y se eliminaron los acentos.

2. La identificación y extracción del texto descriptivo se llevó a cabo usando expresiones regulares, 167 informes se descartaron porque no eran adecuados o estaban vacíos.

3. Se eliminaron los caracteres no alfanuméricos, manteniendo los puntos y espacios.

4. Se eliminaron las palabras vacías o *stopwords*, a excepción de ”sin”, ”no”, ”ni” y ”con”.

5. Las palabras sufrieron un proceso de *stemming*, que las reduce a su raíz. Por ejemplo, la palabra ”infiltración” quedó reducida a ”infiltr”.

Visión global

A continuación vamos a presentar algunas estadísticas interesantes sobre los datos, para conocer a qué tipo de pacientes pertenecen las radiografías y cómo están distribuidas.

Los pacientes fueron sometidos a una media de 1.62 estudios radiológicos, cada uno de los cuales contiene una o varias imágenes en diferentes proyecciones, siendo la más común la posteroanterior (PA), seguida por la lateral. La media de imágenes por paciente es de 2.37. El rango de edad de los pacientes va desde los 0 a los 105 años, siendo la media 58.5 años y la mediana 62 años.

En la figura 4.3 observamos la densidad de imágenes por edad y sexo en los pacientes, vemos que la distribución está desviada hacia edades más avanzadas. También existen diferencias en cuanto al sexo, predominando la cantidad de imágenes correspondientes a mujeres en el intervalo de 40 a 60 años y las correspondientes a hombres, de 70 a 80 años.

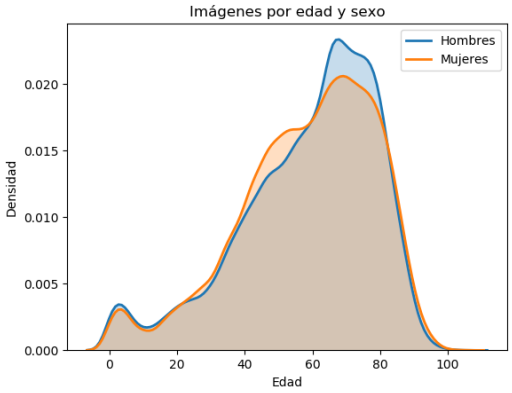


Figura 4.3: Densidad de imágenes por edad y sexo. **Fuente:** Elaboración propia a partir de los datos.

Errores encontrados

En este apartado comentaremos varios problemas o errores encontrados a la hora de trabajar con los datos para aprendizaje automático. El conjunto de datos está, en general, bien mantenido y la presencia de errores es mínima, aún así cabe comentar algunos fallos:

- El primero es la presencia de NaNs (Not a Number) en varias imágenes dentro de campos esenciales como las etiquetas o el texto del informe. El total de imágenes con este problema es de 120, que han sido eliminadas durante el preprocesamiento de los datos.

- También hay algunas imágenes que carecen de etiquetas, quedando este campo vacío. El total en este caso es de 777, todas ellas etiquetadas automáticamente.

- El campo de proyección de algunas imágenes está equivocado, por ejemplo una imagen con proyección lateral está etiquetada como PA (posteroanterior). Este tipo de errores no se pueden comprobar automáticamente, pero afortunadamente parece que son poco comunes, habiendo encontrado pocos casos mediante revisión manual. La causa es probablemente el uso de un modelo neuronal para establecer la etiqueta de proyección de algunas imágenes.

4.1.3. Ética y protección de datos

Como hemos ido observando durante todo el capítulo, los datos con los que estamos trabajando pertenecen a pacientes del Hospital San Juan de Alicante. Personas de todas las edades, que han sufrido enfermedades graves y han necesitado un diagnóstico o un seguimiento mediante radiografías torácicas. Algunas de ellas es probable que hayan fallecido como consecuencia de estas enfermedades.

Aquí entra en juego el aspecto ético de la recopilación y uso de este tipo de datos. Por parte de las instituciones y autores, los objetivos están claros: se utilizan los datos, en este caso las imágenes, para obtener avances tecnológicos y médicos, mejorando las herramientas de diagnóstico y en última instancia la salud de las personas.

También surgen otras consideraciones éticas a tener en cuenta, como por ejemplo los hallazgos incidentales de importancia para la salud. El BIMCV indica, en el apartado "Consideraciones Éticas" de su página web, que los avances en medicina y tecnología pueden provocar la detección de anormalidades asintomáticas. La comunicación de estos hallazgos puede traer consecuencias para el paciente en particular, entrando en debate la revelación de la identidad para hacerla efectiva, junto con el impacto psicosocial tanto en la persona como en familiares [22].

Éticamente existen además ciertos riesgos del uso de datos masivos en investigaciones sanitarias, como por ejemplo los sesgos que pueden llegar a producir. Este tipo de aspectos deben ser evaluados por comités de ética de la investigación (CEI), que deben identificar responsables y decidir el alcance de las investigaciones para evitar la aparición de una **discriminación** o estigmatización inintencionadas [23].

En cuanto al aspecto legal de las investigaciones sanitarias con datos masivos, en general los CEI deben analizar el cumplimiento de los principios de protección de datos según la legislación vigente. Concretamente qué datos se van a almacenar, en qué condiciones y durante cuánto tiempo para evitar riesgos en la seguridad. Se debe definir también el uso concreto que van a tener los datos y los **objetivos perseguidos**.

En el caso del conjunto de datos en cuestión, el proyecto fue a aprobado por el comité de investigación institucional y las imágenes y sus informes asociados pasaron por un proceso de **anonimización**. El intento o la re-identificación de los sujetos de los datos está estrictamente prohibido en las condiciones de uso.

Por último, todas las personas que dieron su **consentimiento** para el uso de sus datos tienen derecho al acceso, rectificación y cancelación de sus datos, revocando el consentimiento, siendo estos propiedad del paciente.

4.2 Librerías Python

Actualmente existen numerosas formas de programar modelos de aprendizaje automático en gran variedad de lenguajes de programación. En nuestro caso vamos a exponer las opciones que ofrece *Python* con algunas librerías que facilitan enormemente la implementación de las redes neuronales y, al ser ampliamente usadas, cuentan con documentación extensa y una comunidad implicada.

4.2.1. TensorFlow Keras

En primer lugar contamos con TensorFlow [3], una librería software de código abierto para aprendizaje automático desarrollada por Google y disponible para varios lenguajes como Java, JavaScript o la versión más extendida para Python. La librería permite al usuario crear modelos de aprendizaje automático de manera modular y permite la aceleración del proceso de entrenamiento utilizando GPUs.

Junto con TensorFlow encontramos su API de alto nivel Keras [40], que facilita las tareas de creación de modelos de redes neuronales, su proceso de entrenamiento, el manejo de datos y el guardado de los modelos ya entrenados, entre otras funcionalidades.

4.2.2. Scikit-learn

Por otra parte Scikit-learn [4] es otra librería para aprendizaje automático de código abierto disponible en Python. Esta librería incluye algoritmos de aprendizaje automático para clasificación, regresión y análisis de grupos como por ejemplo máquinas de soporte vectorial o K-medias. Además, cuenta con numerosas funciones para procesar datos y para obtener métricas del desempeño de los modelos como la precisión, el *recall* o el F1-score, algunas de las cuales hemos usado durante la obtención de resultados y veremos más adelante.

4.2.3. PyTorch

Por último, PyTorch [5] es otra librería también de código abierto orientada específicamente a la visión por ordenador y al procesamiento de lenguaje natural, desarrollada por el laboratorio de investigación IA de Facebook. Esta librería se considera de nivel inferior a TensorFlow o Keras al trabajar directamente con tensores, que son arrays multidimensionales de números. También dispone de soporte para GPUs. PyTorch se usa en algunas aplicaciones o programas conocidos basados en *deep learning*, como puede ser el piloto automático de los vehículos eléctricos Tesla.

4.3 Entorno de trabajo

Para la realización de este trabajo, concretamente para la parte más costosa del entrenamiento, hemos contado con varias máquinas proporcionadas por el departamento de investigación Pattern Recognition and Human Language Technology (PRHLT) de la Universidad Politécnica de Valencia y un ordenador personal propio.

En cuanto al hardware, estas máquinas cuentan con dos GPUs Nvidia GeForce RTX 2080, que permiten la aceleración del proceso de entrenamiento de nuestros modelos, compuestos por redes neuronales. Por otra parte el ordenador personal cuenta con una GPU Nvidia GeForce GTX 770, que ha servido principalmente para análisis de los datos y entrenamiento básico de algunos modelos.

En lo referente a software, las máquinas del PRHLT cuentan con el sistema operativo Ubuntu 20 y con la librería Nvidia CUDA para aceleración por GPU. El entorno de ejecu-

[3] Página web de TensorFlow: www.tensorflow.org
[4] Página web de Scikit-learn: scikit-learn.org
[5] Página web de PyTorch: pytorch.org

ción está compuesto por la distribución libre Anaconda [6] del lenguaje de programación Python. Con esta distribución se ha instalado la versión 3.7 de Python y varias librerías necesarias para los distintos procesos llevados a cabo:

- TensorFlow

  Ya mencionada anteriormente, librería orientada a aprendizaje automático mediante redes neuronales.

- Keras

  API de alto nivel de TensorFlow que facilita la creación de modelos y la experimentación.

- Scikit-learn

  Librería de aprendizaje automático, en este caso la usaremos para obtener las métricas necesarias para algunos de nuestros problemas, como puede ser la precisión multi-etiqueta o el F1-score, también la obtención de las matrices de confusión.

- Pandas

  Esta librería gestiona la parte de carga y análisis de datos, con ella podemos leer ficheros CSV, crear estructuras de datos y manipularlas.

- Matplotlib

  Finalmente, esta librería ofrece infinitas posibilidades en la creación de gráficos. Ha sido usada para visualizar algunas partes de los datos durante el análisis y para obtener gráficos de los resultados de los modelos.

4.4 Métricas empleadas

Las métricas que vamos a utilizar para la evaluación de los modelos varían según la tarea, en este caso las más diferentes se van a usar en la clasificación de texto multi-etiqueta: F1-score y la exactitud multi-etiqueta o *multi-label accuracy*. En la parte de clasificación binaria, al ser una tarea más sencilla, se utilizan métricas más típicas como es la *accuracy*.

4.4.1. Accuracy y Multi-label accuracy

Esta métrica proporciona la fracción de muestras que han sido correctamente etiquetadas, en el caso de un problema binario se define de la siguiente manera

$$Accuracy = \frac{(TP + TN)}{N}$$

siendo TP y TN *true positives* y *true negatives* respectivamente y N el número de muestras.

En el caso del problema multi-etiqueta, al contar con tantas clases diferentes la métrica debe se más estricta y únicamente se cuenta como correcta la predicción de una muestra si esta es exactamente igual al conjunto de etiquetas correcto. La *accuracy* calculada para en un conjunto de datos, es la media de todas las muestras de ese conjunto.

[6] Página web de Anaconda: www.anaconda.com

Por ejemplo, si una predicción realizada por el modelo para un informe cualquiera consiste en las tres etiquetas "cardiomegaly", "sternotomy" y "suture material", y la predicción realizada por el modelo solo incluye dos de esas tres, el valor de *accuracy* para esta muestra sería 0.

4.4.2. F1-score

El F1-score o Valor-F combina los valores de precisión (*precision*) y exhaustividad (*recall*) de las predicciones. Concretamente usaremos la métrica MicroF1, que se calcula globalmente en lugar de realizar la media. A continuación vemos las fórmulas de *precision* (MicroP), *recall* (MicroR)y F1-score (MicroF1) para la variante micro:

$$MicroP = \frac{\sum_{\ell=1}^{|\mathcal{L}|} TP_\ell}{\sum_{\ell=1}^{|\mathcal{L}|} (TP_\ell + FP_\ell)}$$

$$MicroR = \frac{\sum_{\ell=1}^{|\mathcal{L}|} TP_\ell}{\sum_{\ell=1}^{|\mathcal{L}|} (TP_\ell + FN_\ell)}$$

$$MicroF1 = 2 \cdot \frac{MicroR \cdot MicroP}{MicroR + MicroP}$$

donde $|\mathcal{L}|$ es el número de etiquetas, en nuestro caso $|\mathcal{L}| = 98$ y ℓ la etiqueta en cuestión; TP, FP y FN son *true positives*, *false positives* y *false negatives* respectivamente.

4.5 Implementación

La presente sección va a tratar a fondo la implementación de los diversos modelos de aprendizaje automático en el lenguaje de programación Python, tanto los referentes a texto como a imágenes. Además, trataremos el proceso que han necesitado los datos antes de poder entrenar los modelos con ellos.

Durante el preprocesado de los datos y la implementación de los modelos, han surgido problemas e ideas o correcciones que han generado cambios que también comentaremos.

En lo referente a los modelos de texto, se ha ido avanzando de manera incremental, primero con un modelo de clasificación binaria básico, para terminar en el modelo de clasificación multi-etiqueta. Para este último problema contamos con dos variantes, una que contiene redes LSTM y otra compuesta por redes convolucionales.

Seguidamente, el modelo de imágenes resolverá un problema de clasificación binaria 'normal' vs. 'no normal', utilizando redes neuronales convolucionales.

4.5.1. Preprocesado de datos

El conjunto de datos usado, PadChest, ya cuenta con ciertos datos preprocesados, como puede ser el texto de los informes visto en el apartado 4.1.2. Aún así, ha sido necesario aplicar ciertos procesos adicionales, para resolver algunos errores, transformar el

texto plano a las estructuras de datos necesarias y agrupar las etiquetas para reducir el número de clases.

En primer lugar, utilizando la librería Pandas, se cargó el fichero que contiene los datos de texto. Seguidamente, se eliminaron las líneas cuyo campo de etiquetas contenía la etiqueta 'exclude' o estaba vacío. Al intentar crear una lista a partir del texto de las etiquetas surgió un error provocado por la presencia de NaNs en este campo, estas líneas se consiguieron eliminar gracias a una función de la librería que elimina estas indeterminaciones.

Seguidamente se procedió a eliminar las líneas con el mismo campo de ReportID, puesto que para varias imágenes del mismo estudio, el informe es el mismo. Después de estos procesos ya fue posible transformar el texto de las etiquetas a una lista de etiquetas para cada muestra.

A continuación se procedió a agrupar varias etiquetas en términos más generales, basados en el árbol de términos incluido en el conjunto de datos. Esta agrupación facilita el problema de clasificación multi-etiqueta al contar con menos etiquetas y la pérdida de información no es significativa, ya que se agrupan en términos más generales y muchas etiquetas agrupadas cuentan con pocas apariciones, la mayoría sin llegar a doscientas. El número de etiquetas distintas se redujo de 193 a 98. En la figura 4.4 podemos ver un ejemplo del árbol de términos de una agrupación de etiquetas realizada, en este caso los términos hijo tienen muy pocas apariciones (campo "counts").

```
pleural effusion [CUI:C2073625, counts:8282, 8585]\\
├── loculated pleural effusion [CUI:C0747639, counts:184, 184]\\
├── loculated fissural effusion [counts:86, 86]\\
├── hydropneumothorax [CUI:C0020303, counts:29, 29]\\
├── empyema [CUI:C0014009, counts:4, 4]\\
└── hemothorax [CUI:C0019123, counts:0, 0]\\
```

Figura 4.4: Extracto del árbol de términos donde se puede ver un ejemplo de agrupación de etiquetas realizado. **Fuente:** [21]

En este punto era necesario transformar los datos de texto plano a vectores numéricos, para poder servirlos de entrada al modelo. En primer lugar, cada lista de etiquetas asociada a un informe se transformó a un vector *One-hot*, es decir; un vector compuesto por ceros y unos, donde las posiciones con un '1' indican que la etiqueta correspondiente a esa posición está asociada a la muestra en cuestión. Este proceso se realizó utilizando la función *MultiLabelBinarizer* de la librería Scikit-learn.

El texto de los informes también se transformó a vectores mediante un proceso de tokenización, este proceso crea un vector para cada informe, asignando a cada palabra un índice en base al vocabulario total extraído de todos los informes. En este caso se usó la función *Tokenizer* perteneciente a la librería Keras. Seguidamente se rellenaron los vectores con el índice 0, que no representa ninguna palabra, para que todos tuviesen la misma longitud.

En cuanto al preproceso de las imágenes, después de descargarlas se seleccionaron las radiografías con proyecciones posteroanteriores (PA) al ser las más comunes, eliminando el resto de imágenes y con ellas las líneas de informe correspondientes en el fichero CSV. Seguidamente se redimensionaron las imágenes a 512x512 píxeles para facilitar su manejo, en la figura 4.5 podemos ver un ejemplo de este proceso. Esta resolución fue elegida intentando buscar un compromiso entre tamaño del conjunto y la conservación de la información. La relación de aspecto se mantuvo, ya que el método de redimensión rellenaba el espacio restante con ceros (correspondientes a color negro).

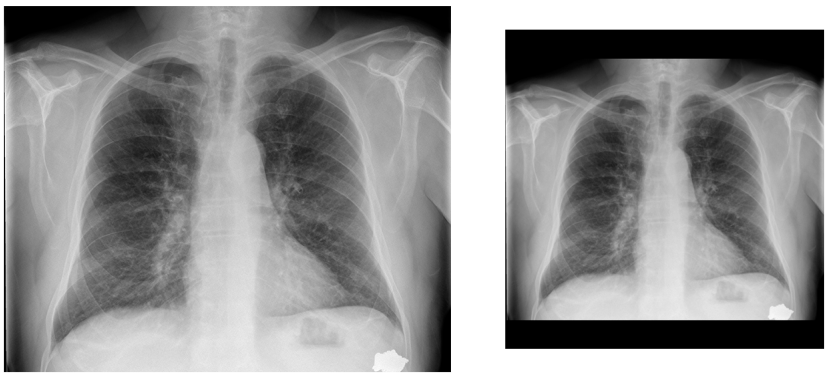


Figura 4.5: Redimensión de una radiografía. A la izquierda podemos ver la imagen original con un tamaño de 2973x2432 píxeles, a la derecha su versión en 512x512, con el relleno en color negro para preservar la proporción original.

Finalmente, tanto en texto como en imágenes, antes de alimentar los modelos con los datos se crearon tres particiones aleatorias: una para entrenamiento; otra de validación, para ajustar el modelo durante el proceso de entrenamiento y una de test, para probar el desempeño del modelo entrenado.

4.5.2. Modelos de texto

Las tareas de clasificación de texto resultan muy útiles en variedad de aplicaciones y su implementación y entrenamiento resultan procesos relativamente sencillos. En nuestro caso, el problema requiere etiquetar algunas frases, pertenecientes a informes radiológicos, con términos generales que puedan estar referidos en el propio texto o en el contexto.

Para la creación de los modelos de texto se ha seguido un proceso incremental. En primer lugar se construyó un modelo sencillo, con capas totalmente conectadas, de clasificación binaria de los informes, entre informes normales y no normales. Seguidamente se procedió a implementar los modelos de clasificación multi-etiqueta, basados en parte en el estado del arte, con dos arquitecturas distintas, LSTM y convolucional, que detallamos más adelante.

Además de tratar la arquitectura, también examinaremos la importancia de la última capa en los modelos y las funciones de activación, así como las funciones de pérdida elegidas y la capa de *embedding*, que se utiliza para representar el texto vectorizado de entrada en múltiples dimensiones.

Capa de *Embedding*

La capa de *embedding* es generalmente la primera capa de un modelo de texto neuronal. Esta capa representa el texto, previamente vectorizado, de manera multidimensional. Cada palabra está representada por un vector de ciertas dimensiones, que se especifican en las opciones de la capa.

Estas representaciones se comportan dentro de la red neuronal como un conjunto de pesos más, los cuales la red debe ajustar. A medida que la red va aprendiendo estos parámetros las palabras con significados similares cuentan también con representaciones cercanas en el espacio.

En el caso de nuestros modelos, con las capas de *embedding* obtenemos representaciones vectoriales de 100 dimensiones para cada palabra, que se van ajustando a medida que el proceso de entrenamiento progresa y son procesadas por el resto del modelo. Se puede utilizar también *embeddings* pre-entrenados, que han mostrado buenos resultados en clasificación de texto [15]. En nuestro caso, al ser vocabulario más específico, se ha preferido no utilizarlos.

A continuación, en la figura 4.6, se muestra un ejemplo de la representación inicial de un informe con etiqueta 'normal'. En primer lugar observamos el texto plano, seguido de la representación *tokenizada*, donde el índice 2 representa la palabra 'sin'; el 8, 'hallazg'; el 27, 'patolog' y el 1 representa el punto. En este caso no se muestra el relleno con ceros por simplicidad, sin embargo el vector real cuenta con una longitud de 150, siendo esta la longitud de la secuencia de un informe más larga. Por último contamos con la representación que genera la capa de *embedding*, un vector de longitud 100 para cada palabra. En este caso también se ha omitido parte de la representación por simplicidad.

```
"sin hallazg patolog ."

[ 2  8 27  1 ... ]

[[ 0.03243002 -0.03734557  ... -0.02072297  0.02724857 ]
 [ 0.0278206  -0.03395475  ...  0.04185159 -0.04495212 ]
 [-0.02249765 -0.01747997 ... 0.01541792 -0.02420899 ]
 [ 0.04943588 -0.04131027 ... 0.00765505 -0.00046446 ]
 ... ]
```

Figura 4.6: Distintas representaciones del texto, en orden descendente: texto plano, texto tokenizado y *embeddings*.

Clasificación binaria

Para la tarea de clasificación binaria de texto 'normal' vs. 'no normal' se ha tenido que hacer un ajuste en los datos para obtener las dos clases. Todas las muestras que contenían la etiqueta 'normal' se han sumado a la clase del mismo nombre y las que carecían de esta etiqueta, a la clase 'no normal'.

Este modelo cuenta, como el resto de modelos en esta sección, con la primera capa de *Embedding* para la representación multidimensional del texto. Seguidamente contamos con una capa *GlobalMaxPooling* de una dimensión, esta capa obtiene los valores máximos en la dimensión temporal, en este caso el valor máximo de cada característica entre todas las palabras, produciendo un vector de longitud 100. A continuación, una capa oculta totalmente conectada con 16 neuronas y función de activación ReLU.

Finalmente, la última capa cuenta con 1 neurona y la función de activación *sigmoid*, que devuelve un valor entre 0 y 1. La predicción de la clase se realiza según la correspondiente probabilidad de este valor.

La función de coste utilizada a minimizar en este modelo es la *cross-entropy* (CE) o entropía cruzada, en este caso binaria (*binary cross-entropy*). La función se define de la siguiente manera

$$CE = -\sum_{i}^{C} t_i \log(s_i)$$

donde C es el número total de clases, t_i es el valor real de la clase i y s_i es la probabilidad predicha por el modelo, después de pasar por la función de activación. En nuestro caso $C = 2$ y la función se puede expresar como

$$CE = -t_1 \log(s_i) - (1 - t_1) \log(1 - s_1)$$

estando la clase C_1 enfrentada contra su contraria, de manera que si la predicción no es correcta el coste será $-\log(s_1)$ si $t_1 = 1$ o por otra parte, $-\log(1 - s_1)$ si $t_1 = 0$.

Clasificación multi-etiqueta

Para esta tarea vamos a tratar de etiquetar el texto de los informes radiológicos entre 98 etiquetas diferentes, correspondientes a hallazgos radiológicos y diagnósticos diferenciales. Cada texto puede estar asociado a varias etiquetas, ya que estas no son excluyentes entre sí. La tarea de clasificación multi-etiqueta la podemos ver como varios problemas diferentes de clasificación binaria, tantos como etiquetas potenciales.

En este caso hemos desarrollado dos modelos, uno con arquitectura recurrente LSTM y otro con arquitectura convolucional (CNN). El objetivo buscado al realizar ambos modelos es probar el desempeño de cada uno en esta tarea y comparar sus rendimientos. Además entrenaremos el modelo C-LSTM propuesto en [35], que combina la arquitectura convolucional con redes LSTM, para también compararlo con nuestros modelos.

Todos los modelos tienen como primera capa el *embedding*, con las mismas características expuestas en el apartado de clasificación binaria. Particularmente la capa de *embedding* del modelo LSTM procesa secuencias de texto de diferente longitud, eliminando los ceros de relleno, ya que la arquitectura recurrente admite entradas de longitud variable.

Como segunda capa, en el **modelo CNN**, contamos con dos capas convolucionales de una dimensión, con 128 y 256 filtros respectivamente, una longitud de ventana de 3, desplazamiento de 1 y función de activación ReLU. Entre estas dos capas y al final de la segunda capa convolucional se ha establecido una capa de *MaxPooling* que reduce el tamaño de la representación manteniendo los valores más representativos. En la figura 4.7 observamos la arquitectura del modelo, tomando como entrada la salida de la capa inicial de *embedding*, seguidamente las dos capas convolucionales con *MaxPooling* y una capa de aplanamiento para finalizar con las capas totalmente conectadas.

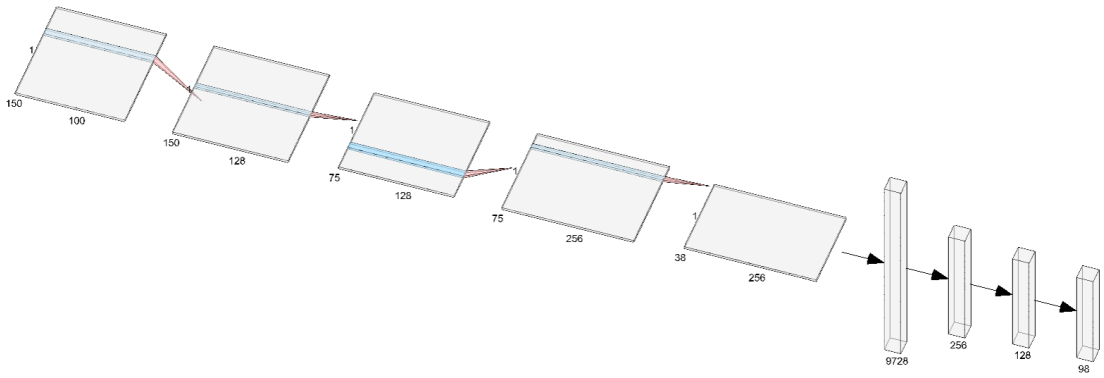


Figura 4.7: Arquitectura convolucional del modelo de clasificación de texto multi-etiqueta. **Fuente:** Generada en alexlenail.me/NN-SVG

En cuanto al **modelo LSTM**, después del *embedding* tenemos una unica capa LSTM con 128 unidades, seguida de una capa de *dropout* que corta conexiones entre las neu-

ronas para añadir regularización, en este caso con una probabilidad de 0.4. En la figura 4.8 observamos una representación del modelo LSTM comenzando por la capa de *embedding* seguida por la capa LSTM, con funciones de activación tanh y función de activación recurrente sigmoid, seguida de las capas totalmente conectadas. Se ha omitido la representación de las capas de *dropout* por simplicidad.

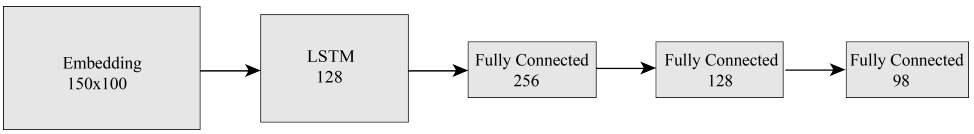


Figura 4.8: Arquitectura LSTM del modelo de clasificación de texto multi-etiqueta.

Para el modelo **C-LSTM** perteneciente a [35] contamos con una capa de *dropout* después de los *embeddings* para añadir regularización, seguida de una única capa convolucional con 32 filtros de tamaño 3 para extraer las características y con función de activación ReLU. Estas características se alimentan directamente a la capa LSTM que cuenta con 150 unidades. Seguidamente el modelo cuenta con las capas finales totalmente conectadas, en este caso la última capa tiene además un factor de regularización L2.

Finalmente, para todos los modelos las **capas finales** son idénticas. Contamos con dos capas totalmente conectadas con 256 y 128 neuronas y función de activación ReLU. Entre estas capas y antes de la de salida también se ha añadido cierta regularización mediante *dropout*, en este caso con una probabilidad del 20 %. La capa de salida cuenta con tantas neuronas como etiquetas posibles, en nuestro caso 98. La función de activación de esta capa es la *sigmoid*, que devuelve un vector con 98 valores entre 0 y 1, siendo estos la probabilidad de pertenencia de la muestra a cada etiqueta.

La predicción de las etiquetas se realiza mediante un umbral, si la probabilidad de pertenencia a cierta clase supera el umbral establecido, esa muestra se etiqueta en consecuencia. En nuestro caso el umbral seleccionado es 0.5.

La función de coste utilizada en este problema es *binary cross-entropy*, la misma que en clasificación binaria anterior, ya que la tarea se ha modelado como varios problemas de clasificación binaria.

4.5.3. Modelo de imágenes

La clasificación de imágenes, en concreto la tarea de clasificación binaria, es realmente interesante en nuestro caso. Como ya se ha comentado a lo largo del trabajo, la posibilidad de ofrecer un diagnóstico más rápido mediante un modelo de aprendizaje automático puede suponer grandes beneficios, tanto para pacientes como para el personal sanitario.

Existen varios modelos convolucionales conocidos y ya entrenados, como ResNet [33], que con algunos ajustes han producido buenos resultados incluso en clasificación multi-etiqueta entre varios hallazgos radiológicos [24]. En nuestro caso se pretende mostrar como un modelo más sencillo, formado también por redes neuronales convolucionales, puede diferenciar con un porcentaje significativo de acierto radiografías de tórax con algún hallazgo fuera de lo normal.

A continuación trataremos la arquitectura propuesta para el modelo y sus particularidades, así como la función de coste usada. Además, mostraremos un ejemplo de dos imágenes que componen los datos de entrenamiento, una normal y otra con algún hallazgo anómalo.

Clasificación binaria

Las imágenes, a parte del proceso de redimensión, han necesitado de un etiquetado, normal o no normal, en base a las etiquetas que figuran el fichero CSV, así como un manejo por lotes debido al gran tamaño del conjunto, que trataremos en detalle en próxima sección. Además se han normalizado los valores de los píxeles para establecerlos entre 0 y 1, mejorando así la eficiencia de la red.

El modelo propuesto para esta tarea con imágenes es un modelo sencillo adecuado a este tipo de clasificación, compuesto por tres capas convolucionales de dos dimensiones con filtros de tamaño 11x11, 7x7 y 3x3, con capas de *max-pooling* después de cada capa convolucional. El número de filtros de las capas convolucionales son 32, 64 y 128, respectivamente, como podemos ver en la figura 4.9.

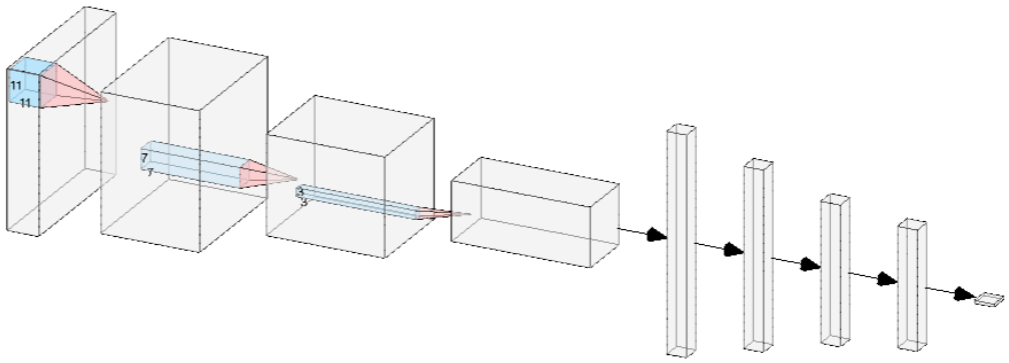


Figura 4.9: Arquitectura convolucional del modelo de clasificación binaria de imágenes. No se muestran las capas de MaxPooling por simplicidad.

El modelo cuenta además con capas de *batch normalization* después de cada capa convolucional, que normalizan los datos y ayuda al proceso de entrenamiento. Por último contamos con una capa de aplanamiento o *flatten* antes de las cuatro capas totalmente conectadas que forman el clasificador, con 512, 128 y 64 neuronas respectivamente. La última capa, como en las tareas binarias anteriores, está compuesta por una única neurona y la función de activación sigmoid.

La función de coste es también idéntica a las otras tareas de clasificación binaria, binary *cross-entropy*.

Además de este modelo sencillo, también realizaremos experimentos sobre una variante de ResNet llamada **ResNet50**, ya que está compuesta por 50 capas convolucionales. El único cambio que ha sido necesario añadir ha sido la capa final, para establecer una sola neurona debido al problema binario y la función de activación *sigmoid*. En la próxima sección entrenaremos ambos modelos para los mismos datos de radiografías torácicas.

4.6 Experimentos y entrenamiento

Un correcto entrenamiento de los modelos de redes neuronales es fundamental para un buen aprendizaje de los pesos y la adecuada generalización de nuevas muestras, en definitiva, para que el modelo sea útil.

En la presente sección se van a exponer los diferentes procesos de entrenamiento para los modelos implementados anteriormente, además de los ajustes en los parámetros de entrenamiento para tratar de obtener los mejores resultados, sin que se produzca un sobreajuste en los modelos. Para ello se estudiarán las particiones de datos usadas y los tiempos de entrenamiento que han resultado adecuados.

Se mostrarán también algunas gráficas de las funciones de *accuracy* y de coste, obtenidas durante los procesos de entrenamiento de algunos modelos.

4.6.1. Modelos de texto

En primer lugar trataremos el proceso de entrenamiento de los modelos de texto. El **modelo de clasificación binaria**, que clasifica informes en normales vs. no normales fue muy fácil de entrenar, ya que disponemos de muchos datos y la tarea es muy sencilla.

Las muestras totales ascendieron a 106.237, de las cuales 57.366 pertenecen al conjunto de entrenamiento, 6.375 al conjunto de validación y 42.496 al conjunto de test, todos ellos aleatorios. La distribución de ambas clases está desbalanceada, alrededor de un 33 % de las muestras pertenecen la clase 'normal' y más o menos un 66 % a la 'no normal', aunque en este caso a penas afecta al entrenamiento.

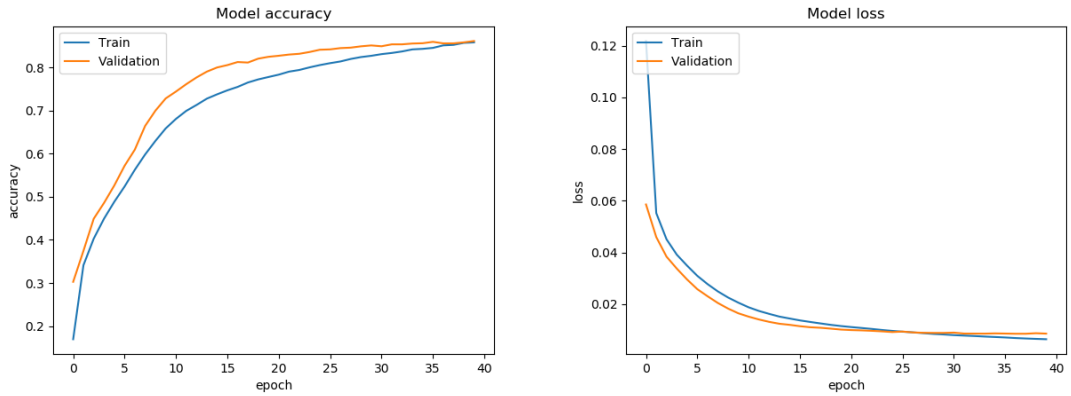
Como ya hemos indicado, la tarea es muy sencilla y por tanto el entrenamiento es rápido. Se ha entrenado el modelo buscando minimizar el resultado de la función de coste del conjunto de validación, para evitar un sobreaprendizaje de las muestras de entrenamiento. Hemos usado el algoritmo de optimización Adam, con un tamaño de lote o *batch size* de 256. El tiempo de entrenamiento ha sido de 6 *epochs*, momento en el cuál el valor de la función de coste para el conjunto de validación ha resultado mínimo.

En cuanto a ambos **modelos de clasificación multi-etiqueta**, que etiquetan los informes radiológicos entre 98 etiquetas diferentes, necesitaron de un proceso de entrenamiento más adecuado a la dificultad de la tarea.

El número de muestras totales usadas para entrenar ambos modelos es el mismo que para el modelo binario anterior, 106.235. El conjunto de entrenamiento en este caso está formado por 66.927 muestras, el de validación 7.437 y el de test por 31.871. En este caso el desbalanceo de clases es obvio, contando con clases con 5 muestras y otras con más de 10.000, siendo en gran parte esta la dificultad de la tarea para el modelo.

Los experimentos realizados para los modelos consistieron en variar el *batch size* y el algoritmo de entrenamiento entre RMSProp y Adam. Los resultados no variaron significativamente, por lo que se decidió usar un *batch size* de 512 y el algoritmo Adam, con los parámetros por defecto de Keras, siendo el *learning rate* 0.001 y los parámetros β_1 y β_1 establecidos a 0.9 y 0.999 respectivamente. El número de *epochs* máximo se estableció en 100, pero se añadió una condición de parada para evitar el sobreaprendizaje: si el valor de la función de coste en el conjunto de validación no disminuye en 3 *epochs*, se detiene el entrenamiento.

En primer lugar, el entrenamiento del **modelo CNN** con el conjunto de entrenamiento se llevó a cabo ajustándolo con el conjunto de validación, con el valor de su función de coste como condición de parada. En la figura 4.10 observamos dos gráficas, que representan la *accuracy* (a) y la función de coste (b) de ambos conjuntos durante el entrenamiento, que en este caso ha durado 40 *epochs*. Aunque realmente el mejor modelo para el conjunto de validación se da en la *epoch* 37 y es el que usaremos en el capítulo de resultados.



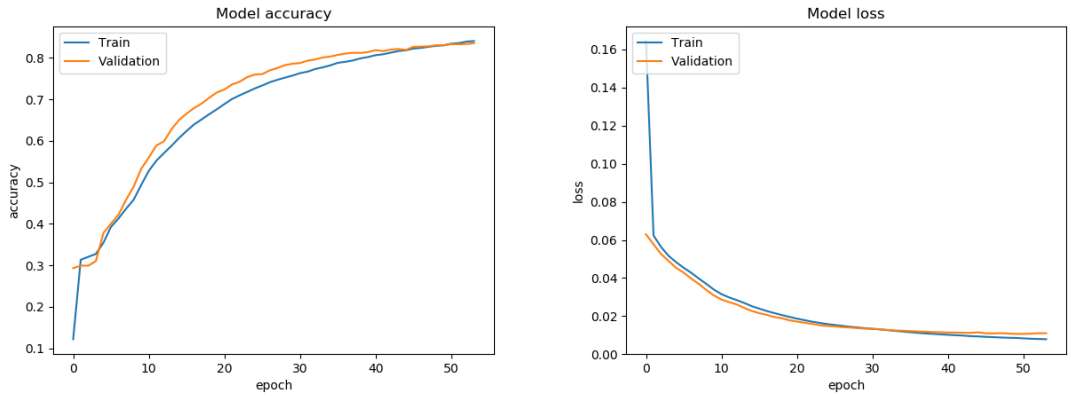
(a) *Accuracy* **(b)** Función de coste

Figura 4.10: Gráficos de *accuracy* y coste durante el proceso de entrenamiento del modelo convolucional de clasificación de texto multi-etiqueta. Las curvas representadas en color azul pertenecen al conjunto de entrenamiento y las representadas en color naranja, al conjunto de validación.

Podemos observar que durante gran parte del proceso de entrenamiento, el valor de la función de coste en el conjunto de validación es inferior a su valor en el conjunto de entrenamiento. Esto puede ser debido a varios factores, entre ellos la aplicación de regularización durante el entrenamiento y no durante la validación o la medición de validación realizada al final de cada *epoch*, mientras que la de entrenamiento se realiza durante.

Seguidamente, en cuanto al **modelo LSTM**, los parámetros usados y las particiones de los datos han sido idénticos a los del modelo convolucional. Este modelo, al contar con la capa LSTM, requiere un tiempo de entrenamiento superior. En total el proceso consistió en 54 *epochs*, cada una con un coste computacional más alto que el modelo CNN. En este caso, el valor de la función de coste para el conjunto de validación quedó estancado en la *epoch* 51, seleccionando este como mejor modelo.

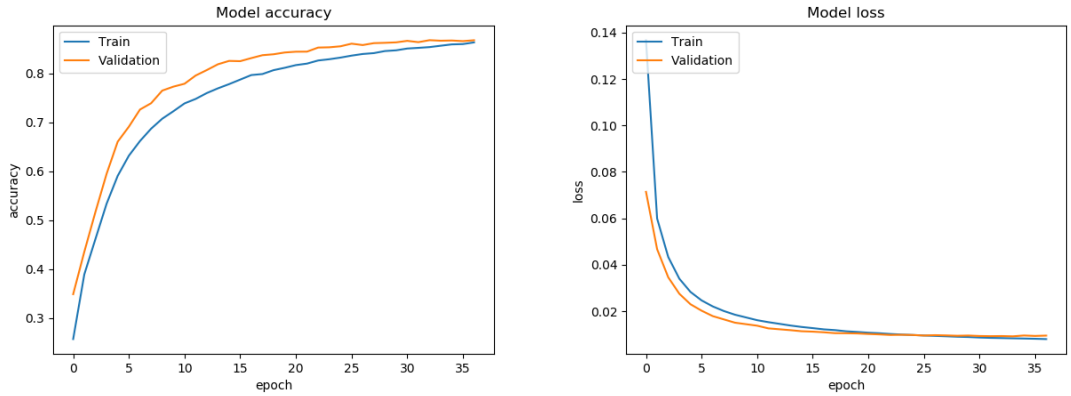
En la figura 4.11 observamos las dos gráficas de *accuracy* y de coste para ambos conjuntos, similares a las del modelo convolucional pero con una progresión más lenta.



(a) *Accuracy* **(b)** Función de coste

Figura 4.11: Gráficos de *accuracy* y coste durante el proceso de entrenamiento del modelo LSTM de clasificación de texto multi-etiqueta. Las curvas representadas en color azul pertenecen al conjunto de entrenamiento y las representadas en color naranja, al conjunto de validación.

Finalmente, el **modelo C-LSTM** ha sido entrenado con los mismos datos y parámetros que los anteriores. Los tiempos de entrenamiento han sido ligeramente superiores a los del modelo CNN pero inferiores a los del modelo LSTM, requiriendo 37 *epochs* y seleccionando el modelo de la *epoch* 34 como el mejor, debido al valor de la función de coste en el conjunto de validación. Podemos observar los gráficos de *accuracy* y coste para el entrenamiento de este modelo en la figura 4.12.



(a) *Accuracy* **(b)** Función de coste

Figura 4.12: Gráficos de *accuracy* y coste durante el proceso de entrenamiento del modelo C-LSTM de clasificación de texto multi-etiqueta. Las curvas representadas en color azul pertenecen al conjunto de entrenamiento y las representadas en color naranja, al conjunto de validación.

4.6.2. Modelos de imágenes

En este caso tratamos el entrenamiento del **modelo convolucional de clasificación binaria de imágenes**. Este modelo ha requerido de unos ajustes más importantes que el resto de modelos y de un tiempo de entrenamiento bastante superior, ya que el trabajo con imágenes y redes convolucionales es computacionalmente costoso.

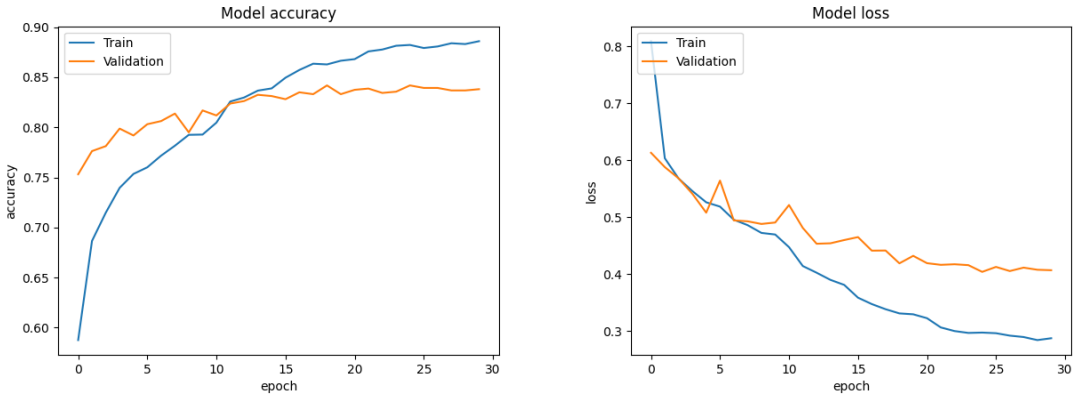
Se dividieron los datos en tres conjuntos, como en los casos anteriores: entrenamiento, validación y test. El total de las imágenes seleccionadas asciende a 16.459, el conjunto de entrenamiento cuenta con 11.189, el de validación con 1.629 y el de test con 3.641. La distribución de clases es aproximadamente un 56 % normal y un 44 % no normal, en esta última clase se incluyen radiografías etiquetadas con neumonía, derrame pleural, neumotórax o enfisema; por este motivo y por la eliminación de muestras normales para balancear las clases, se ha reducido el número de imágenes seleccionadas.

El primer experimento sobre el modelo base se realizó añadiendo las capas de *Batch-Normalization*, que normalizan los datos de los *batches* o lotes. Esta adición produjo mejores resultados y se conservaron. Para el siguiente experimento se añadieron más capas de regularización, obteniendo pequeñas mejoras en el resultado, alrededor de un 1 % en *accuracy* del conjunto de test. El algoritmo de optimización elegido fue Adam, con variaciones dinámicas del factor de aprendizaje o *learning rate*, reduciéndolo a medida que avanzaba el entrenamiento para conseguir ajustes de los pesos más precisos.

Al resultar los tiempos de entrenamiento largos, se eligió el tamaño de lotes o *batch size* lo más grande posible, ya que está limitado por la capacidad de las tarjetas gráficas, en este caso fue de 32.

En la figura 4.13 observamos las gráficas de *accuracy* y coste durante el entrenamiento, para los conjuntos de entrenamiento y validación en nuestro modelo. Podemos ver que

la reducción del *learning rate* provoca un entrenamiento más lento en las *epochs* finales, lo que favorece el alcance de un mínimo para la función de coste.



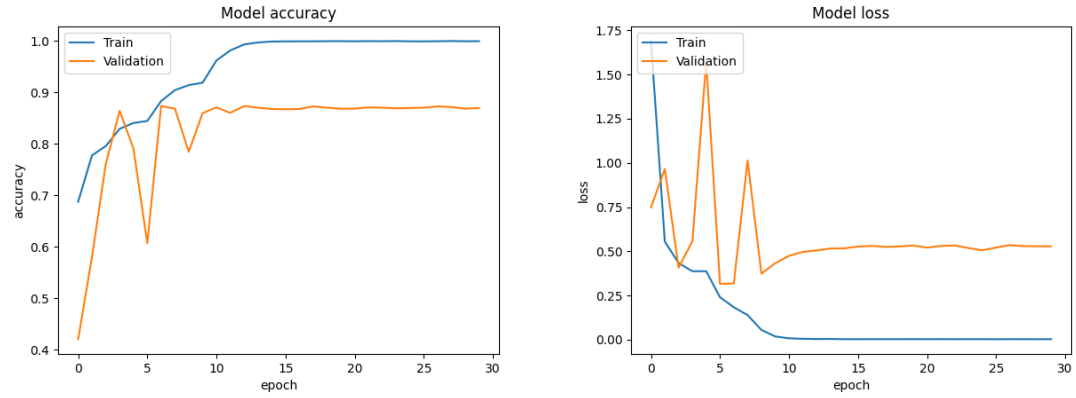
(a) *Accuracy* **(b)** Función de coste

Figura 4.13: Gráficos de *accuracy* y coste durante el proceso de entrenamiento de nuestro modelo CNN de clasificación binaria de imágenes. Las curvas representadas en color azul pertenecen al conjunto de entrenamiento y las representadas en color naranja, al conjunto de validación.

El modelo entrenado seleccionado es el de la *epoch* 24, ya que produjo mejores valores para la función de coste del conjunto de validación.

Seguidamente se probó a entrenar, con los mismo datos, el modelo **ResNet50** con la inicialización de pesos de la tarea ImageNet.

La gráficas de *accuracy* y coste para este entrenamiento las podemos ver en la figura 4.14, observamos que el entrenamiento al principio es bastante irregular debido al ajuste de los pesos iniciales y más adelante se estabiliza. El modelo seleccionado que produjo mejores resultados en el conjunto de validación se consiguió en la *epoch* 9.



(a) *Accuracy* **(b)** Función de coste

Figura 4.14: Gráficos de *accuracy* y coste durante el proceso de entrenamiento del modelo ResNet50 de clasificación binaria de imágenes. Las curvas representadas en color azul pertenecen al conjunto de entrenamiento y las representadas en color naranja, al conjunto de validación.

Además las imágenes se han normalizado para ambos modelos, dividiéndolas entre 255 para establecer los valores entre 0 y 1, también se han redimensionado a 256x256 píxeles para adecuarlas a ambos modelos.

CAPÍTULO 5

Resultados experimentales

Una vez se ha realizado el proceso de entrenamiento en los diferentes modelos propuestos, se prueban los modelos con datos diferentes a los usados para este proceso y se evalúa el rendimiento y capacidad de generalización de los mismos ante nuevas muestras.

En el presente capítulo expondremos los resultados obtenidos por los modelos de clasificación de texto e imagen en los **conjuntos de test** construidos en cada caso y compararemos los distintos modelos para cada caso.

Los resultados mostrados sobre los conjuntos de test elegidos aleatoriamente pueden no representar completamente la realidad de los modelo frente a nuevas muestras, pero da una idea general del desempeño y capacidad conseguidas mediante el entrenamiento.

5.1 Resultados en etiquetado de texto

En esta sección mostraremos los resultados obtenidos con los modelos de texto y compararemos ambos modelos de clasificación multi-etiqueta, utilizando las métricas descritas anteriormente. En primer lugar el modelo inicial binario, cuya tarea era sencilla y los resultados han sido buenos. Seguidamente el modelo convolucional de clasificación multi-etiqueta comparándolo con el modelo LSTM y el C-LSTM para la misma tarea, cuyos resultados han sido similares. Además mostraremos algún ejemplo de las predicciones de los modelos multi-etiqueta, tanto erróneas como satisfactorias.

El **modelo de clasificación binaria** está orientado a resolver una tarea sencilla y fue implementado para familiarizarse con los datos. Por lo tanto, los resultados medidos con el conjunto de test son casi perfectos, con un 99.2 % de *accuracy*.

En cuanto a los **modelos de clasificación multi-etiqueta**, contamos con dos variantes ya estudiadas, convolucional y LSTM, además de una fusión de ambas, la C-LSTM. Con estos modelos se han obtenido resultados similares al evaluarlos con el conjunto de test, los cuales podemos ver en la tabla 5.1.

Como podemos observar el modelo C-LSTM, que combina capas convolucionales con LSTM, genera mejores resultados con el mismo conjunto de test tanto en *accuracy* como en MicroF1, además el proceso de entrenamiento es menos costoso. Con una arquitectura más elaborada sería posible conseguir mejores resultados, aunque con un proceso de entrenamiento más largo.

Modelo	Epochs	Multi-label accuracy (%)	MicroF1
Conjunto de Test			
CNN	37	86.2	0.937
LSTM	51	83.0	0.923
C-LSTM	34	**86.8**	**0.950**

Tabla 5.1: Resultados de multi-label accuracy y MicroF1 en el conjunto de test para los modelos convolucional, LSTM y C-LSTM. En negrita están resaltados los mejores resultados.

Un ejemplo de predicción correcta del modelo C-LSTM, perteneciente al conjunto de test, es para el texto de informe

"catet venos central con entrad ven yugul derech extrem distal ven cav superior . aparicion infiltr alveolar perihiliar bilateral pinzamient sen costofren derech respect estudi previ fech 28 05 2013 descart edem agud pulmon .",

el modelo consigue predecir las cuatro etiquetas asociadas

"catheter", "costophrenic angle blunting", "infiltrates" y "pulmonary edema",

siendo sus probabilidades 0.99, 0.98, 0.99 y 0.99 respectivamente.

Un ejemplo de predicción errónea sería el texto

"sutur esternotomi medi . valvul cardiac metal . marcapas . marc elongacion aortic . hili congest ."

para el cual el modelo consigue predecir 4 etiquetas de 5, quedando la quinta etiqueta con a penas un 0.15 de probabilidad, no superando el umbral de 0.5 y, por tanto, no siendo seleccionada.

5.2 Resultados en clasificación de imágenes

En el caso de la clasificación de imágenes, compararemos nuestro modelo con el modelo ResNet50, ambos entrenados con los mismos datos y evaluados con el mismo conjunto de test compuesto por 3.641 muestras. También mostraremos las matrices de confusión de cada modelo en el conjunto de test y alguna predicción realizada por los modelos con la probabilidad o el nivel de confianza de las mismas.

En primer lugar **nuestro modelo convolucional**, relativamente sencillo ya que está compuesto por 3 capas convolucionales, ha obtenido un 84.8 % de *acccuracy* en el conjunto de test.

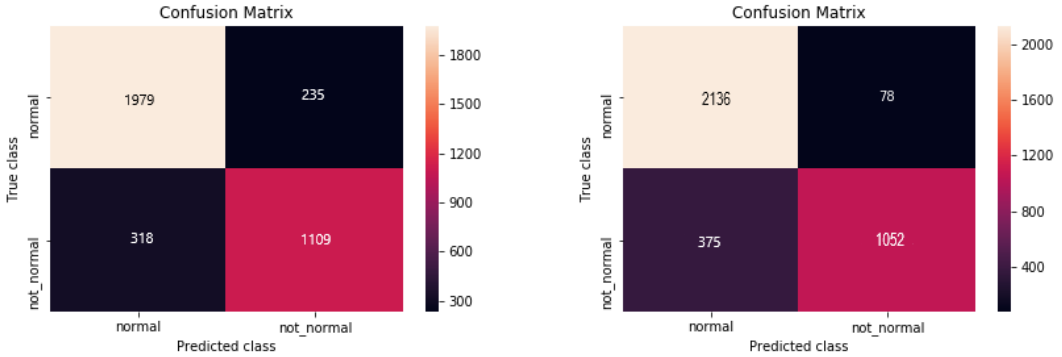
Seguidamente, el modelo **ResNet50** inicializado con los pesos de la tarea ImageNet y afinado con los datos de entrenamiento propuestos ha obtenido un 87.6 % de *accuracy* en el conjunto de test. A continuación, se presentan los resultados en forma de tabla 5.2.

Modelo	Epochs	Accuracy (%)
Conjunto de Test		
Modelo Propio	24	84.8
ResNet50	9	**87.6**

Tabla 5.2: Resultados de accuracy en el conjunto de test para el modelo convolucional propio y para el modelo ResNet50. En negrita está resaltado el mejor resultado.

Observamos que con el modelo ResNet50 obtenemos mejores resultados de *accuracy* para el conjunto de test y con menos *epochs* en el proceso de entrenamiento. Cabe resaltar que la diferencia de resultados entre ambos modelos no es demasiado significativa, teniendo en cuenta las diferencias en tamaño, contando nuestro modelo con más de 1 millón de parámetros entrenables y ResNet50 con más de 23 millones.

Es también interesante observar los resultados de cada modelo en una matriz de confusión. En la figura 5.1 se presentan estas matrices para ambos modelos, en el eje Y se establecen las clases verdaderas y en el eje X las predicciones de cada modelo.

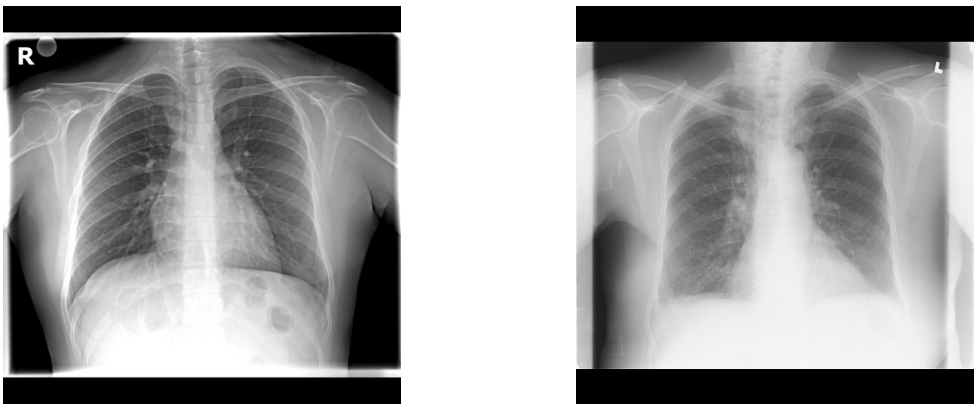


(a) Modelo Propio **(b)** ResNet50

Figura 5.1: Matrices de confusión para el modelo propio (a) y el modelo ResNet50, entrenados con los datos propuestos del conjunto PadChest.

En la figura observamos que nuestro modelo predice mejor las muestras no normales, mientras que ResNet50 predice mejor las muestras normales. En una aplicación real, y más en el caso de la salud, son relevantes este tipo de criterios, ya que puede interesar predecir más falsos positivos que falsos negativos. Por supuesto para una aplicación real se requeriría mucho más testeo y modelos más específicos, pero estos nos dan una idea de la complejidad de la tarea y los aspectos a tener en cuenta.

Finalmente, pondremos dos ejemplos de radiografías extraídas aleatoriamente del conjunto de test y las predicciones de ambos modelos. En la figura 5.2 observamos dos radiografías pertenecientes al conjunto de test, la primera (a) pertenece a la clase "normal" y la segunda (b) a la clase "no normal", concretamente el paciente sufre neumonía y derrame pleural entre otras patologías.



(a) Radiografía normal **(b)** Radiografía no normal

Figura 5.2: Radiografías pertenecientes al conjunto de test correctamente predichas por ambos modelos.

Las dos muestras han resultado correctamente predichas por ambos modelos. Para nuestro modelo, la radiografía (a) es ”normal” con una probabilidad de 0.941 y la (b) es ”no normal” con una probabilidad de 0.895. En el caso del modelo ResNet50, la predicción para la muestra (a) es ”normal” con una probabilidad de 0.966 y la (b), ”no normal” con 0.967.

En definitiva, ambos modelos producen buenos resultados aunque en esta tarea más sencilla puede ser conveniente utilizar un modelo más pequeño y reservar los modelos grandes como ResNet para tareas más complejas.

CAPÍTULO 6

Conclusiones

Como hemos visto durante todo el trabajo, cada vez es más importante la puesta en valor de los datos masivos en todos los ámbitos. La tecnología actual permite en gran medida este proceso, desde el análisis más básico hasta la implementación de modelos con técnicas de aprendizaje automático, que consigan resultados en aplicaciones reales. En concreto, la aplicación de este tipo de tecnologías a la sanidad podría mejorar enormemente la calidad de la atención a los pacientes reportando beneficios a largo plazo.

Durante el estudio del estado del arte, hemos podido ver diversas aplicaciones de los distintos tipos de redes neuronales, muchas de las cuales se pueden aplicar a los datos médicos, como puede ser el conjunto usado PadChest. Además, las explicaciones del funcionamiento de las arquitecturas más básicas a las más complejas, ampliando los conocimientos estudiados durante el grado y la rama de computación, han ayudado a entender el proceso mediante el cual estos modelos consiguen aprender a partir de ejemplos.

El cumplimiento de los objetivos planteados al principio del trabajo ha sido satisfactorio, se ha conseguido el desarrollo de varios modelos que han llevado a cabo dos tareas: el etiquetado de informes radiológicos y la clasificación de radiografías de tórax. En conjunto, los distintos modelos han conseguido resultados prometedores en las distintas particiones de test seleccionadas, además se ha comparado su desempeño final ante estos nuevos datos. Concretamente, para la clasificación multi-etiqueta de texto se han comparado los modelos convolucional, LSTM y C-LSTM, mostrando este último modelo mejores resultados para nuestros datos. En cuanto a la clasificación binaria de las radiografías, ambos modelos convolucionales han generado buenas predicciones, aunque el modelo más profundo ResNet50 ha sido superior. Estos modelos y otros similares podrían ser usados en aplicaciones concretas en el campo de la sanidad, aunque primero serían necesarias revisiones estrictas ya que se estarían aplicando a la salud de las personas, por ejemplo asistiendo a los profesionales durante el proceso de diagnóstico de los pacientes.

Cabe destacar también algunas dificultades encontradas durante la realización del trabajo, sobre todo al ser la primera vez que se ha trabajado con tal cantidad de datos reales, como puede ser el manejo de archivos grandes y su procesado en el caso de las imágenes. Además, en ocasiones el trabajo con las librerías ha resultado frustrante al no conocer en profundidad todas las posibilidades que ofrecen o durante largos procesos de entrenamiento de los modelos que no resultaron satisfactorios. A pesar de esto, se han extendido los conocimientos adquiridos durante el grado en materia de aprendizaje automático, sobre todo en el manejo de gran cantidad de datos y la creación de modelos de redes neuronales.

En lo personal, este trabajo me ha ayudado a profundizar más en las redes neuronales de forma tanto práctica como teórica, ha sido interesante el uso de las librerías de Python y la búsqueda y lectura de numerosos artículos relacionados, que de una manera u otra han dado forma a este documento.

Finalmente, como trabajo futuro se propone un estudio en profundidad de las necesidades reales en la sanidad pública referentes a tecnología, donde al aprendizaje automático pueda ser clave, y en base a estas desarrollar modelos concretos más adecuados. En el caso de este trabajo se podrían ajustar los parámetros de las arquitecturas a los datos, utilizando el conjunto de validación para realizar dichos experimentos y mostrar los resultados, en lugar de solo basarlos en el proceso de entrenamiento. Por poner otro ejemplo siguiendo la línea de este trabajo, se podrían crear también varios modelos que detectasen hallazgos específicos en las radiografías y combinarlos con técnicas de *ensemble learning* para producir un modelo mucho más capaz de asistir en diagnósticos complejos.

Bibliografía

[1] Paul Covington; Jay Adams; Emre Sargin (2016). "Deep Neural Networks for YouTube Recommendations". *Proceedings of the 10th ACM Conference on Recommender Systems*. 191–198

[2] Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain". *Psychological Review.* 65 (6): 386–408.

[3] Cortes, C.; Vapnik, V. (1995). "Support-vector networks". *Machine Learning*. 20: 273–297.

[4] Thomas M. Cover; Peter E. Hart (1967). "Nearest neighbor pattern classification". *IEEE Trans. Inf. Theory*. 13: 21-27.

[5] MacQueen, J. B. (1967). "Some methods for classification and analysis of multivariate observations". *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. 1: 281–297.

[6] Kaelbling, Leslie P.; Littman, Michael L.; Moore, Andrew W. (1996). "Reinforcement Learning: A Survey". *Journal of Artificial Intelligence Research*. 4: 237–285.

[7] P. Burnap; M. L. Williams. (2015). "Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making". *Policy and Internet*. 7(2): 223–242.

[8] Minsky, Marvin; Papert, Seymour (1969). "Perceptrons: An Introduction to Computational Geometry". *MIT Press*.

[9] Madhavan, Anand; Ng, Andrew. (2009). "Large-scale deep unsupervised learning using graphics processors". *Proceedings of the 26th International Conference On Machine Learning*. 873-880.

[10] Ernestina Menasalvas; Consuelo Gonzalo; Alejandro Rodríguez González (2017). "Big Data en salud: retos y oportunidades". *Economía industrial*. 405: 87-97

[11] Zhang, Ziqi & Robinson, D. & Tepper, Jonathan. (2018). "Detecting hate speech on Twitter using a convolution-GRU based deep neural network". *European semantic web conference*. 745-760.

[12] Xiang Zhang; Junbo Zhao; Yann LeCun (2015). "Character-level Convolutional Networks for Text Classification". *Advances in neural information processing systems*. 649-657.

[13] Hochreiter, S.; Schmidhuber, J. (1997). "Long Short-Term Memory". *Neural Computation*. 9: 1735-1780.

[14] Pascanu, R.; Mikolov, T.; Bengio, Y. (2013). "On the difficulty of training Recurrent Neural Networks" *International conference on machine learning*. 1310-1318.

[15] Kim, Yoon (2014). "Convolutional Neural Networks for Sentence Classification" *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1746-1751.

[16] Feiyang Chen; Nan Chen; Hanyang Mao; Hanlin Hu (2018). "Assessing four Neural Networks on Handwritten Digit Recognition Dataset (MNIST)" *arXiv preprint*. arXiv: 1811.08278.

[17] Fukushima, Kunihiko (1980). "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position" *Biological Cybernetics*. 36 (4): 193–202.

[18] Ciresan, Dan; Ueli Meier; Jonathan Masci; Luca M. Gambardella; Jurgen Schmidhuber (2011) "Flexible, High Performance Convolutional Neural Networks for Image Classification". *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*. 2: 1237–1242.

[19] Yoav Goldberg (2015). "A Primer on Neural Network Models for Natural Language Processing". *Journal of Artificial Intelligence Research*. 57: 345-420.

[20] Marco Túlio Ribeiro; Sameer Singh; Carlos Guestrin (2016). "Why Should I Trust You?: Explaining the Predictions of Any Classifier" *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135-1144.

[21] Aurelia Bustos; Antonio Pertusa; Jose-Maria Salinas; Maria de la Iglesia-Vayá (2019). "PadChest: A large chest x-ray image dataset with multi-label annotated reports" *arXiv preprint*. arXiv:1901.07441, 2019.

[22] Wolf, SM; Lawrenz FP; Nelson CA; et al. (2008). " Managing incidental findings in human subjects research: analysis and recommendations". *J Law Med Ethics.* 36(2): 219-211.

[23] Itziar de Lecuona. (2018). "Evaluación de los aspectos metodológicos, éticos, legales y sociales de proyectos de investigación en salud con datos masivos (big data)". *Gaceta Sanitaria*. 32(6): 576-578.

[24] Hongyu Wang; Yong Xia (2018). "ChestNet: A Deep Neural Network for Classification of Thoracic Diseases on Chest Radiography" *arXiv preprint*. arXiv: 1807.03058.

[25] Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (1986). "Learning representations by back-propagating errors". *Nature*. 323 (6088): 533–536.

[26] Duchi, John; Hazan, Elad; Singer, Yoram (2011). "Adaptive subgradient methods for online learning and stochastic optimizatio". *JMLR*. 12: 2121–2159.

[27] Diederik, Kingma; Ba, Jimmy (2014). "Adam: A method for stochastic optimization". *arXiv preprint*. arXiv: 1412.6980.

[28] Rafal Jozefowicz; Oriol Vinyals; Mike Schuster; Noam Shazeer; Yonghui Wu. (2016). "Exploring the limits of language modeling". *arXiv preprint*. arXiv: 1602.02410, 2016.

[29] Chelba, Ciprian; Mikolov, Tomas; Schuster, Mike; Ge, Qi; Brants, Thorsten; Koehn, Phillipp; Robinson, Tony. (2013). "One billion word benchmark for measuring progress in statistical language modeling". *arXiv preprint*. arXiv: 1312.3005.

[30] Zhou, C.; Sun, C.; Liu, Z.; Lau, F. (2015). "A C-LSTM neural network for text classification". *arXiv preprint*. arXiv: 1511.08630.

[31] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. (2013). "Efficient estimation of word representations in vector space". *arXiv preprint*. arXiv:1301.3781.

[32] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks". *Advances in neural information processing systems*. 1097-1105.

[33] He, K.; Zhang, X.; Ren, S.; Sun, J. (2016). "Deep residual learning for image recognition". *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 770-778.

[34] Ng, M. Y.; Lee, E. Y.; Yang, J.; Yang, F.; Li, X.; Wang, H.;Hui, C. K. M. (2020). "Imaging profile of the COVID-19 infection: radiologic findings and literature review". *Radiology: Cardiothoracic Imaging*. 2(1): e200034.

[35] Zhou, C.; Sun, C.; Liu, Z.; Lau, F. (2015). "A C-LSTM neural network for text classification". *arXiv preprint*. arXiv:1511.08630.

[36] Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016) *Deep Learning*. MIT Press. www.deeplearningbook.org

[37] Vall d'Hebron evalúa la eficacia de un software de inteligencia artificial para acelerar el diagnóstico de la COVID-19 mediante la tomografía computarizada de tórax https://www.vallhebron.com/es/noticias/vall-dhebron-evalua-la-eficacia-de-un-software...

[38] Hinton, Geoffrey. "Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude". https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf p. 26.

[39] Introduction to Machine Learning *Draft of Incomplete Notes* by Nils J. Nilsson https://ai.stanford.edu/people/nilsson/mlbook.html

[40] Chollet, François and others. "Keras". https://keras.io

[41] Artem Oppermann. "Optimization Algorithms in Deep Learning". https://towardsdatascience.com/optimization-algorithms-in-deep-learning