



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA INFORMÁTICA

Detección de glándulas en biopsias de próstata

Autor

Álvaro Fernández García

Director

Nicolás Pérez de la Blanca Capilla



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, septiembre de 2019

Detección de glándulas en biopsias de próstata

Álvaro Fernández García

Palabras clave: próstata, cáncer, glándulas, aprendizaje profundo, biopsia, red neuronal convolucional, detección de objetos, patología digital, patología computacional, imágenes WSI, procesamiento de imágenes.

Resumen

El cáncer está constituido como la segunda causa de muerte global y concretamente el de próstata, es el segundo tipo con el mayor número de nuevos casos identificados cada año en España y el más mortal entre la población masculina. Una forma eficaz de diagnosticarlo es a través de la anatomía patológica: se extrae una muestra de tejido del órgano en cuestión (biopsia) que posteriormente es analizada bajo el microscopio por un especialista. Sin embargo, los avances en el campo del procesamiento de imágenes digitales, han permitido el surgimiento de una nueva técnica: la patología digital. Las biopsias son procesadas por potentes escáneres que generan imágenes de alta resolución que pueden ser analizadas a través de un software. Se plantea la creación de un algoritmo (una herramienta de ayuda al diagnóstico) para detectar, en estas imágenes, las células glandulares prostáticas, las estructuras biológicas en las que el cáncer se hace visible, haciendo uso de redes neuronales convolucionales y técnicas de visión por computador, al mismo tiempo que se estudia su desempeño en imágenes de gran tamaño.

Glands detection in prostate biopsies

Álvaro Fernández García

Keywords: prostate, cancer, glands, deep learning, biopsy, convolutional neural network, object detection, digital pathology, computational pathology, WSI imaging, image processing.

Abstract

Cancer is constituted as the second cause of global death and specifically that of prostate, it is the second type with the highest number of new cases identified each year in Spain and the most deadly among the male population. An effective way to diagnose it is through the pathological anatomy: a sample of tissue is removed from the organ in question (biopsy) that is subsequently analyzed under a microscope by a specialist. However, advances in the field of digital image processing have allowed the emergence of a new technique: digital pathology. Biopsies are processed by powerful scanners that generate high resolution images that can be analyzed through software. The creation of an algorithm (a diagnostic aid tool) is proposed to detect, in these images, prostate glandular cells, the biological structures in which cancer becomes visible, using convolutional neural networks and computer vision techniques, while studying its performance in large images.

Yo, **Álvaro Fernández García**, alumno de la titulación INGENIERÍA INFORMÁTICA de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada, con DNI 75572750V, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Álvaro Fernández García

Granada a 1 de septiembre de 2019.

D. Nicolás Pérez de la Blanca Capilla, Profesor del Área de Ciencias de la Computación e Inteligencia Artificial del Departamento del mismo nombre de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado *Detección de glándulas en biopsias de próstata*, ha sido realizado bajo su supervisión por **Álvaro Fernández García**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 1 de septiembre de 2019.

El director:

Agradecimientos

Al director de este proyecto, por servir de guía, por responder a mis dudas siempre que lo necesitaba y por hacer posible su realización. También a toda mi familia y amigos, no solo por haberme apoyado en este trabajo, sino también por haber sido mi bastón durante estos cuatro años de durísimo esfuerzo.

Índice general

1. Introducción: problema y motivación	1
2. Análisis y enfoque adoptado	7
2.1. Elección del modelo	10
2.1.1. Arquitectura propia	10
2.1.2. Transfer Learning	11
2.2. Tratamiento de las imágenes	15
2.2.1. Eliminación del fondo	18
2.2.2. Etiquetas	20
2.2.3. Procesamiento a dos escalas	21
2.3. El problema del desbalanceo	23
2.4. Detección de glándulas	25
2.5. Algoritmo final	26
3. Proceso de etiquetado	29
4. Experimentos	33
4.1. Experimentos sobre la red propia	34
4.1.1. Elección de hiperparámetros	34
4.1.2. Experimento principal: estimación del error	36
4.1.3. Batch Normalization	44
4.2. Experimentos de Transfer Learning	48
4.3. Discusión de los experimentos	52
4.4. Test de detección	57
5. Desarrollo del Software	61
5.1. Especificación de requisitos	61
5.1.1. Requisitos funcionales	61
5.1.2. Requisitos no funcionales	62
5.2. Planificación	62
5.3. Diseño del Software	63
5.3.1. Lenguaje y bibliotecas	65
5.4. Pruebas	66
5.4.1. Pruebas del módulo DataUtils	66

5.4.2. Pruebas del proceso de entrenamiento y validación	67
5.5. Análisis DAFO	68
6. Conclusión	69
A. La validación cruzada en detalle	71
Bibliografía	75
Glosario	77

Índice de figuras

1.1.	Ejemplo de QuPath, un software para el análisis patológico digital.	2
1.2.	Patología de una célula glandular sana.	3
1.3.	Regiones con células glandulares sanas.	4
1.4.	Ejemplos de distintos grados de cáncer en células glandulares.	5
2.1.	Red neuronal tradicional frente a una CNN.	8
2.2.	Diagrama con la arquitectura implantada.	11
2.3.	Diagrama con la arquitectura de la red VGG16.	13
2.4.	Adaptación de la red VGG16 al problema. Las capas más altas se sustituyen por otras. Los tamaños se ajustan a la entrada.	14
2.5.	Diagrama con la arquitectura de la red InceptionV3.	14
2.6.	Visualización de los conceptos de tamaño de parche y solapamiento.	15
2.7.	Pirámide de resoluciones. Incorpora distintas escalas de una misma imagen.	16
2.8.	Combinaciones probadas para el tamaño de parche y la escala a la que se procesarán las imágenes.	17
2.9.	Efecto del solapamiento escogido. En la imagen se muestran 4 parches solapados.	18
2.10.	Fondo grisáceo de las imágenes.	18
2.11.	Proceso de construcción de la máscara.	19
2.12.	Comparativa entre estroma “puro” y “contaminado”.	21
2.13.	Mismo parche, distintos tamaños de glándula.	21
2.14.	Proceso de extracción de las dos escalas.	22
2.15.	Comparativa de la cantidad de parches de cada clase.	23
2.16.	Ilustración del proceso de detección.	27
3.1.	Se divide la imagen de entrada en parches, el clasificador les asigna una etiqueta y se exportan los resultados a un fichero de texto con extensión .pred.	31

3.2. Este script de QuPath lee el archivo con extensión .pred (indicado en la variable filePath) e importa los parches y sus correspondientes etiquetas al programa.	31
3.3. Así se ven los parches importados en QuPath. A continuación, se corrigen las etiquetas erróneas y con un segundo script se exportan a un fichero de texto con extensión .qpdat.	32
3.4. A partir de la imagen y el fichero .qpdat, podemos refinar los pesos de un clasificador cualquiera pasado como argumento.	32
4.1. Resultados de los experimentos para estimar la penalización L2.	36
4.2. Validación cruzada I: dropout = 0.25 (curvas de aprendizaje por partición)	40
4.3. Validación cruzada I: dropout = 0.25 (Curva ROC media + Matriz de confusión final)	40
4.4. Validación cruzada II: dropout = 0.5 (curvas de aprendizaje por partición)	43
4.5. Validación cruzada II: dropout = 0.5 (Curva ROC media + Matriz de confusión final)	43
4.6. Validación cruzada III: Batch Norm. (curvas de aprendizaje por partición)	47
4.7. Validación cruzada III: Batch Norm. (Curva ROC media + Matriz de confusión final)	47
4.8. Transfer Learning I: VGG16 extracción de características (cruvas de aprendizaje).	49
4.9. Transfer Learning II: VGG16 Fine Tuning (cruvas de aprendizaje).	50
4.10. Transfer Learning III: InceptionV3 extracción de características (cruvas de aprendizaje).	51
4.11. Porcentajes de error por clase en cada partición de la validación cruzada.	55
4.12. Algunos ejemplos sobre la localización de los falsos negativos en validación.	56
4.13. El algoritmo confunde un vaso sanguíneo con una glándula. .	59
4.14. El algoritmo no llega a cubrir el 70 % de estas dos glándulas, las cuales presentan una morfología diferente: sin pliegues y con pocas células secretoras en la frontera.	59
4.15. Detección del algoritmo en muestras malignas de grado 3. .	60
5.1. Calendario con la planificación.	63
5.2. Estructura del sistema.	64

Índice de cuadros

3.1. Sumario sobre el tamaño del Dataset.	30
4.1. Resultados de la elección del LR inicial.	35
4.2. Resultados sin penalización L2.	36
4.3. Resultados con penalización L2.	36
4.4. Validación cruzada I: dropout = 0.25 (media \pm std)	37
4.5. Validación cruzada II: dropout = 0.5 (media \pm std)	44
4.6. Validación cruzada III: Batch Norm. (media \pm std)	48
4.7. Transfer Learning I: VGG16 extracción de características (métricas).	49
4.8. Transfer Learning I: VGG16 extracción de características (matriz de confusión).	49
4.9. Transfer Learning II: VGG16 Fine Tuning (métricas).	50
4.10. Transfer Learning II: VGG16 Fine Tuning (matriz de confusión).	50
4.11. Transfer Learning III: InceptionV3 extracción de características (métricas).	51
4.12. Transfer Learning III: InceptionV3 extracción de características (matriz de confusión).	51

Capítulo 1

Introducción: problema y motivación

Nuestro organismo está constituido por un conjunto de células sólo visibles a través de un microscopio. Estas células se dividen periódicamente y de forma regular con el fin de reemplazar a las ya envejecidas o muertas, y mantener así la integridad y el correcto funcionamiento de los distintos órganos.

El proceso de división de las células está regulado por una serie de mecanismos de control que indican a la célula cuándo comenzar a dividirse y cuándo permanecer estática. Cuando se produce un daño celular que no puede ser reparado se produce una autodestrucción celular que impide que el daño sea heredado por las células descendientes. Cuando estos mecanismos de control se alteran en una célula, ésta y sus descendientes inician una división incontrolada, que con el tiempo dará lugar a un tumor o nódulo.

Cuando las células que constituyen dicho tumor no poseen la capacidad de invadir y destruir otros órganos, hablamos de tumores benignos. Pero cuando estas células además de crecer sin control sufren nuevas alteraciones y adquieren la facultad de invadir tejidos y órganos de alrededor (infiltración), y de trasladarse y proliferar en otras partes del organismo (metástasis), hablamos de tumor maligno, que es a lo que llamamos cáncer.

A día de hoy, el cáncer constituye la segunda causa de muerte global [18], estando superada únicamente por las enfermedades cardiovasculares. Esto se debe a que actualmente no existe un tratamiento 100 % eficaz para combatirlo y la efectividad de los principales tratamientos (cirugía, quimioterapia o radioterapia) depende, en gran medida, de que la enfermedad sea detectada en fases tempranas. El diagnóstico precoz conlleva ventajas importantes para el paciente: recibe tratamientos menos agresivos y más eficaces; los efectos secundarios del tratamiento son menores y si se diagnostica la lesión

premaligna a tiempo, se puede evitar que esta progrese.

En la actualidad, uno de los programas de diagnóstico que se ha establecido como “el estándar de oro” para la detección de cáncer es la anatomía patológica, la rama de la medicina que se ocupa del estudio, por medio de técnicas morfológicas, de las causas, el desarrollo y las consecuencias de las enfermedades, a través del análisis de los tejidos extraídos quirúrgicamente de los pacientes (biopsias) [17]. De esta forma, un patólogo oncólogo (un anatomoatólogo especializado en el análisis de tejidos cancerosos) puede estudiar una pequeña muestra de tejido de un determinado paciente para determinar si el mismo presenta o no lesiones premalignas.

Tradicionalmente, las biopsias han implicado realizar el análisis del tejido a través de un microscopio. Sin embargo, los avances significativos en el ámbito tecnológico, han llevado a la patología a adoptar soluciones innovadoras proporcionadas por el campo del procesamiento de imágenes digitales. Cada vez son más los hospitales que están sustituyendo los microscopios por potentes escáneres de alta resolución capaces de generar imágenes de preparaciones histológicas con un gran nivel de detalle. Ahora, el patólogo realiza su trabajo analizando, marcando y etiquetando esas imágenes de alta resolución (también conocidas como Whole Slide Imaging, WSI [4]), a través de un programa informático (Figura 1.1). A todo este proceso que ha revolucionado el campo en los últimos años se le conoce con el nombre de Patología Digital.

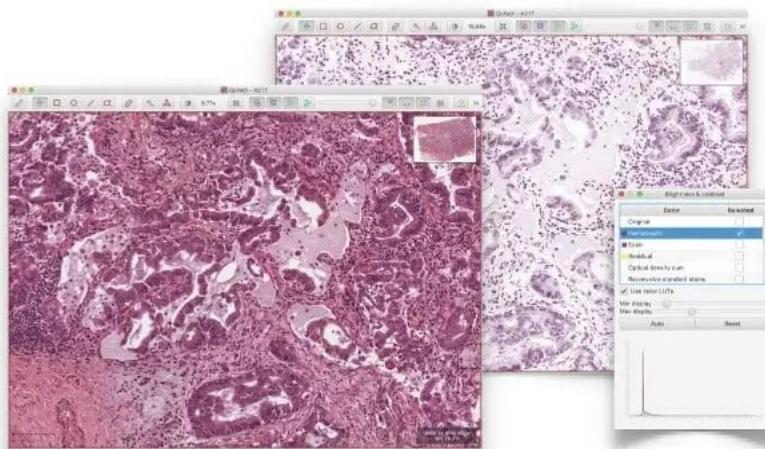


Figura 1.1: Ejemplo de QuPath, un software para el análisis patológico digital.

Además, el que puedan realizarse estudios histológicos a través de un ordenador ha propiciado el surgimiento de un nuevo campo: la patología computacional, una disciplina dedicada al desarrollo de algoritmos y procedimientos automáticos que ayuden a los patólogos a desempeñar su trabajo. Gracias a la patología digital y computacional, se han podido desarrollar sistemas informáticos de ayuda al diagnóstico que facilitan la automatización de las muestras y su interpretación visual.

Dentro de este campo es donde nace el proyecto. Disponemos de una base de datos de imágenes WSI de biopsias de aguja, extraídas para realizar un diagnóstico sobre el cáncer de próstata en distintos pacientes.

Según la Sociedad Española de Oncología Médica (SEOM), los cánceres de colon, próstata, pulmón y mama constituyeron los 4 tipos de cáncer más comunes en España en el año 2017. El claro ganador entre la población masculina es el cáncer de próstata, con un total de 30 mil nuevos pacientes ese mismo año (casi 10 mil afectados más que el segundo cáncer más común entre los hombres) [21].

Aunque la próstata está formada por muchos tipos de células diferentes, más del 99 % de los cánceres de próstata se desarrollan sobre células glandulares. Estas células son las encargadas de producir el líquido prostático, que forma parte del semen [2]. Cuando las glándulas están sanas, suelen presentar una morfología estrellada, con una serie de pliegues y una pequeña fila de células secretoras situadas en su frontera (véase la Figura 1.2). Son fácilmente distinguibles. En la Figura 1.3 puede verse un ejemplo de una muestra de tejido prostático sana donde aparecen marcadas las regiones que contienen células glandulares.

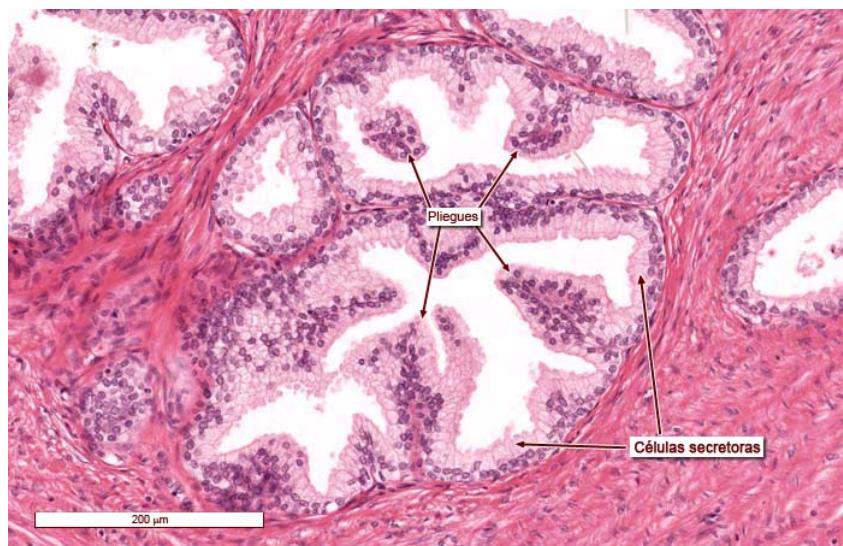


Figura 1.2: Patología de una célula glandular sana.

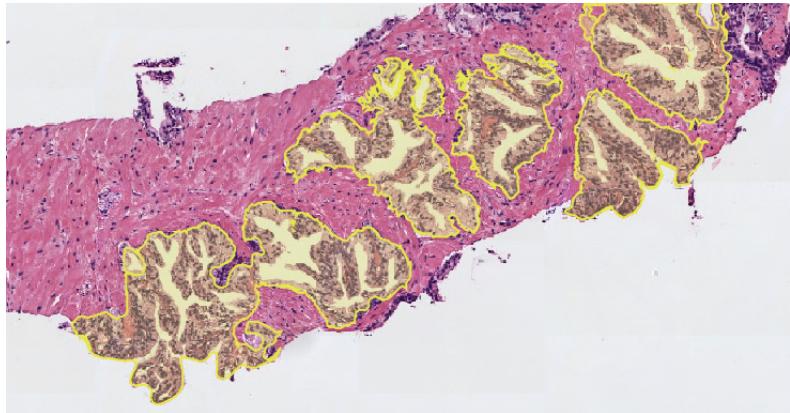


Figura 1.3: Regiones con células glandulares sanas.

Sin embargo, cuando el cáncer comienza a afectarlas, dichas glándulas se degradan paulatinamente, apareciendo formas extrañas (normalmente, los pliegues se pierden), aumentando considerablemente el número de células de la frontera e incluso las estructuras pueden llegar a romperse y desaparecer (véase la Figura 1.4). En función de cuánto se han degenerado las glándulas, podemos distinguir distintos grados de cáncer de próstata. Los niveles pueden ir de 1 a 5, siendo a partir de 2 (o 3 según algunos expertos) cuando comienza a ser preocupante (cáncer en estadio medio) [15].

En este trabajo pretendemos asentar las bases y establecer una primera aproximación en el desarrollo de un algoritmo de patología computacional capaz de reconocer regiones que contienen células glandulares, aquellas zonas de vital importancia para el patólogo durante el diagnóstico del cáncer de próstata. A la vez, estudiaremos el desempeño que tienen los algoritmos en imágenes de alta resolución que, debido a su gran tamaño (más de 10^{10} píxeles), son difíciles de procesar. En la base de datos disponemos de imágenes tanto benignas como malignas. Como primera aproximación, nos centraremos en las benignas (pues su morfología es más clara y presentan una menor variabilidad) y posteriormente realizaremos un análisis de su comportamiento en muestras malignas de tercer grado (que como ya hemos dicho anteriormente, es cuando comienza a ser preocupante).

El problema presenta otra dificultad: para determinar si un paciente tiene o no cáncer de próstata, el patólogo debe estudiar la morfología de las glándulas presentes en la muestra. Para la extracción de la misma, se realiza una punción (de ahí que se les conozca como biopsias de aguja), extrayendo un tubo de tejido que posteriormente será laminado. Las formas que se observen en dichas glándulas dependerán de la inclinación y la localización en la que se haya realizado el punzado. Esto dificulta el reconocimiento de las estructuras y será uno de los principales problemas a los que se enfrenta.

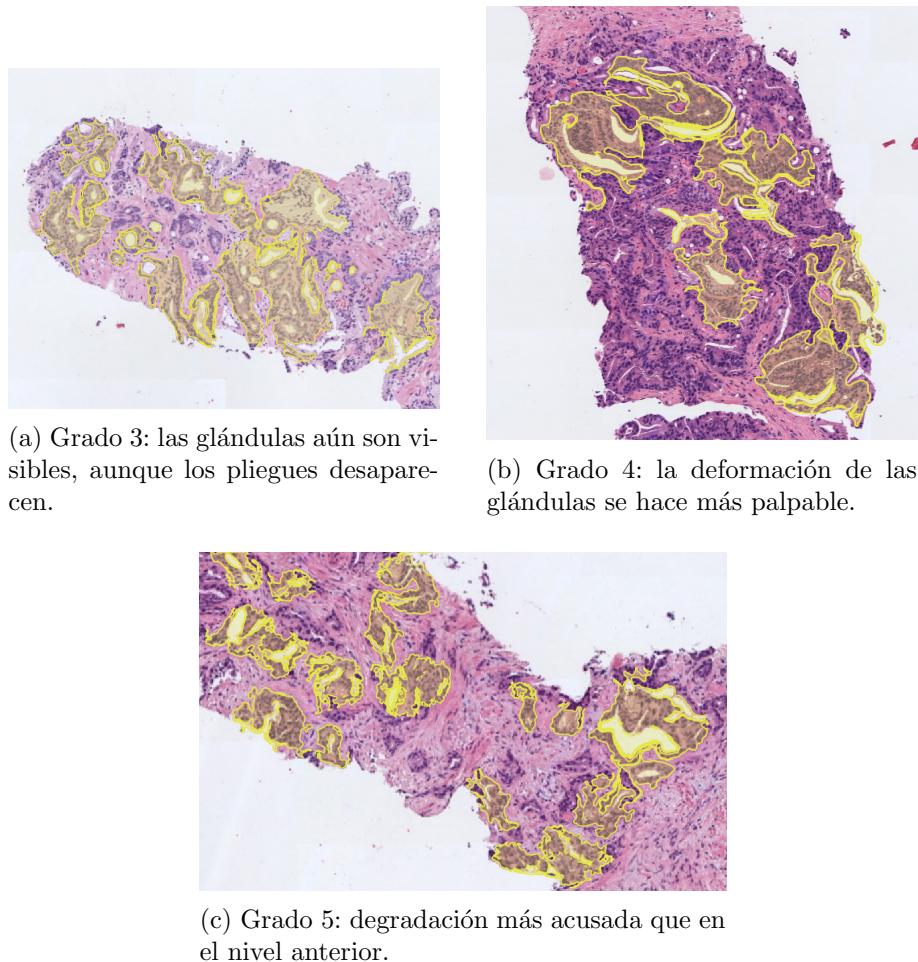


Figura 1.4: Ejemplos de distintos grados de cáncer en células glandulares.

el algoritmo.

En definitiva, el cáncer de próstata está postulado como el segundo tipo de cáncer más común entre la población española y el más común entre los hombres. Gracias a las nuevas técnicas que han surgido con la patología digital y computacional podemos desarrollar herramientas que nos faciliten el diagnóstico temprano de la enfermedad, lo cual será clave para combatirla y aumentar las probabilidades de supervivencia.

Capítulo 2

Análisis y enfoque adoptado

El análisis microscópico de secciones de tejido teñidas con hematoxilina y eosina ha sido y sigue siendo en la actualidad la base para el diagnóstico y la clasificación del cáncer durante el último siglo [6], aunque nuevas técnicas basadas en el uso de inmunomarcadores han sido introducidas en los últimos años. Todas las imágenes usadas en este trabajo han sido teñidas con hematoxilina y eosina.

Concretaremos el problema descrito en el capítulo anterior de la siguiente manera: *el desarrollo un algoritmo capaz de detectar, en imágenes de alta resolución de biopsias de aguja de la próstata, aquellas regiones que contienen células glandulares*. Comenzaremos, con las imágenes de casos benignos en donde las glándulas presentan formas estándares.

Queremos localizar zonas específicas en las imágenes que son relevantes al problema. Esto encaja con la definición de un problema típico de Visión por Computador, una rama de la informática que proporciona un conjunto de herramientas y métodos que permiten obtener, procesar y analizar imágenes del mundo real con la finalidad de que puedan ser tratadas por un ordenador.

De todos los problemas que resuelve la Visión por Computador, este en concreto se trata de un problema de detección de objetos a través de la clasificación. Dada una imagen, queremos localizar la ubicación exacta de todas las instancias de la clase “grupos de células glandulares” y dibujar una caja alrededor suya.

Dicho esto, y para resolverlo, tenemos dos grandes alternativas:

- a) Aplicar técnicas de detección clásicas de Visión por Computador. Normalmente, consisten en un algoritmo de ventana deslizante junto con un método de extracción de características relevantes de las imágenes (descriptores SIFT, histogramas de gradientes orientados... O un procedimiento específicamente diseñado para el problema) y un clasi-

fificador como por ejemplo un SVM.

- b) Utilizar técnicas de aprendizaje profundo, más en concreto, una red neuronal convolucional (también denominadas CNNs o ConvNets). Una red convolucional (LeCun 1989) es un tipo de red neuronal especializada en el procesamiento de datos que tienen una topología de cuadrícula. Algunos ejemplos de estos datos son: series de tiempo, que pueden ser consideradas como una cuadrícula unidimensional que toma muestras en intervalos de tiempo regulares, o las imágenes, que pueden entenderse como una cuadrícula bidimensional de píxeles. Su nombre proviene de que hacen uso de una operación matemática conocida como “convolución”, un tipo especial de operación lineal. Las redes convolucionales son simplemente redes neuronales en las que la transferencia de información entre capas utiliza la convolución en lugar de la multiplicación de matrices al menos una vez [8] (véase la Figura 2.1).

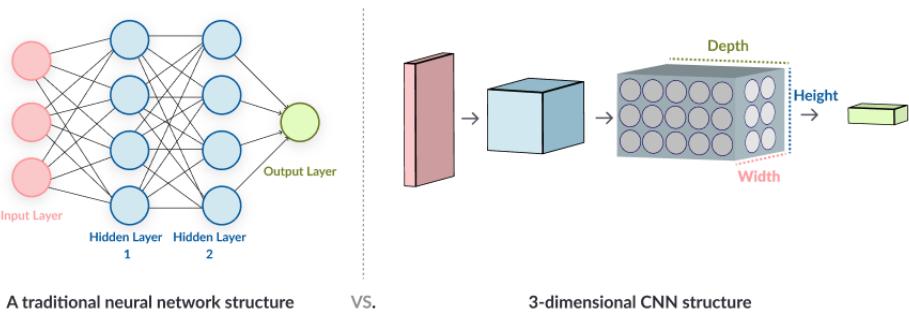


Figura 2.1: Red neuronal tradicional frente a una CNN.

No obstante, hay una de las alternativas expuestas que desde hace relativamente poco tiempo (2012) ha eclipsado completamente a la otra. Se trata de las redes neuronales convolucionales. A parte de su excelente desempeño en una amplia variedad de problemas, gran parte de su éxito radica en la capacidad de generalización que presentan. En los algoritmos clásicos de Visión por Computador hay que diseñar minuciosamente la forma en la que representamos una imagen de cara al algoritmo (extracción de características) para cada problema. Las CNNs no necesitan de este diseño, sino que son capaces de aprender de los datos cuáles son las características más importantes, por lo que pueden aplicarse de forma generalizada a una gran variedad de problemas. Sin embargo, su mayor ventaja es también su principal inconveniente. El hecho de que la red decida de forma autónoma cuáles son las características que le permiten resolver el problema, hace que seamos incapaces de justificar o explicar el porqué de la decisión tomada. Muchos expertos tienen cierta desconfianza en esto y rechazan su aplicación en la medicina.

Sin embargo, y a pesar de la polémica, las ConvNets se aplican cada vez más en el campo de la patología digital y con resultados bastante esperanzadores. A continuación se presenta una lista con algunos de los artículos más relevantes en este campo de investigación:

- En el 2017, el diario Annual Review of Biomedical Engineering, publicó una revisión sobre el impacto que estaba teniendo el aprendizaje profundo en el ámbito de la imagen médica, postulándolo en el estado del arte gracias a su capacidad de explotar representaciones jerárquicas de características aprendidas únicamente de los datos. Destaca sus éxitos en el registro de imágenes, la detección de estructuras anatómicas y celulares, la segmentación de tejidos y el diagnóstico de enfermedades asistidas por computadora [20].
- En el mismo año, la universidad de Cornell publica el artículo “A Survey on Deep Learning in Medical Image Analysis”, donde nuevamente se revisan los principales conceptos de aprendizaje profundo pertinentes al análisis de imágenes médicas y resume más de 300 contribuciones al campo. También examina su uso para la clasificación de imágenes, la detección de objetos, la segmentación, el registro y otras tareas [13].
- El artículo de 2016 titulado “Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis” publicado en Scientific Reports, estudió cómo el aprendizaje profundo constituye una herramienta indispensable para mejorar la objetividad y eficiencia del análisis histopatológico, reduciendo considerablemente el flujo de trabajo de los patólogos en casos como la identificación del cáncer de mama y de próstata [12].
- La universidad de Cornell publicó en mayo de 2019 el trabajo titulado “Prostate Cancer Detection using Deep Convolutional Neural Networks” en el que desarrollaron e implementaron un sistema automatizado basado en CNNs para la detección de cáncer de próstata clínicamente significativo en imágenes DWI axiales, obteniendo un área bajo la curva ROC (AUC) de 0.87. Constituye actualmente el estado de la cuestión sobre este problema [23].
- En el año 2018, se publicó en el diario Journal of Imaging el paper “Glomerulus Classification and Detection Based on Convolutional Neural Networks” que utilizó estos modelos de aprendizaje profundo para la clasificación y detección de glomérulos en segmentos de tejido renal, los cuales son de vital importancia para el correcto diagnóstico de enfermedades. En cierto modo, se trata de un problema similar al nuestro [7].

En definitiva, teniendo en cuenta la trayectoria de las redes neuronales convolucionales y los resultados que están demostrando en el campo, se convierten en las mejores candidatas para este proyecto.

2.1. Elección del modelo

Una vez escogida la técnica (red neuronal convolucional), hay que determinar la arquitectura de la misma, así como sus distintos parámetros: número de capas (de convolución, pooling y totalmente conectadas), ordenación de las mismas, tamaño de los filtros de convolución y pooling, número de filtros por capa de convolución, stride de filtrado, funciones de activación, tipo de pooling, número de neuronas por capa totalmente conectada, padding y dropout. Se ha optado por estudiar el desempeño de tres arquitecturas diferentes, para posteriormente seleccionar aquella que evidencie el mejor comportamiento:

- Una arquitectura de diseño propio, ya empleada anteriormente en un problema médico. Incorpora tres convoluciones con filtros pequeños, Max Pooling y Dropout. Además, procesa las imágenes utilizando una doble escala (véase la Subsección 2.2.3).
- Dos arquitecturas haciendo uso de técnicas de Transfer Learning:
 - VGG16: desarrollada por la universidad de Oxford. Entrenada tanto congelando las capas de convolución como realizando un ajuste fino (Fine Tuning).
 - InceptionV3: desarrollada por Google. Se ha entrenado únicamente el clasificador final, manteniendo los pesos de la parte de extracción de características.

La definición de todos los modelos puede consultarse en el script “models/cnn.py”.

2.1.1. Arquitectura propia

Partiremos de una arquitectura creada por nosotros, que ya ha sido utilizada previamente en un problema similar y que evidenció buenos resultados. La red está formada por un conjunto de tres bloques, donde cada uno de ellos contiene las siguientes tres capas:

1. Convolución, con Leaky ReLU como función de activación.
2. Max Pooling.

3. Dropout con probabilidad de 0,25.

Tras estos tres bloques, se ha añadido una capa totalmente conectada de 150 neuronas también con Leaky ReLU como función de activación y dropout. Finalmente, se colocó un clasificador softmax que nos proporcionará la probabilidad para cada una de las clases.



Figura 2.2: Diagrama con la arquitectura implantada.

La Figura 2.2 muestra que no es una arquitectura muy profunda, pues únicamente incorpora 12 capas, 5 de ellas con parámetros entrenables. Cabe destacar que no son necesarias un gran número de capas para obtener buenos resultados, pues como se ha mencionado anteriormente, esta arquitectura ya ha sido utilizada con éxito en otro problema de índole médica. Además, las imágenes de entrada tienen un total de 6 canales, en lugar de 3 que es lo habitual. Esto se debe al procesamiento a dos escalas que realiza. Los tres primeros canales corresponden a la primera escala (32x32) y los tres últimos, a la segunda (128x128 reescalada a 32x32).

Experimentaremos con distintas regularizaciones y estudiaremos también el impacto que tiene el uso de Batch Normalization en los resultados.

2.1.2. Transfer Learning

Transfer Learning es una técnica de aprendizaje automático donde un modelo desarrollado y entrenado para una determinada tarea, es reusado para otra. Primero, debemos entrenar una red en un conjunto de datos y una tarea base y a continuación, reutilizamos las características aprendidas, o las transferimos, a una segunda red que será entrenada en el conjunto de datos y tarea objetivo. Este proceso funcionará si las características aprendidas en primer lugar son lo suficientemente genéricas.

En el campo del deep learning, cuando se aplica esta técnica para redes neuronales convolucionales, lo más habitual es conservar únicamente las capas de convolución (extracción de características) y modificar las capas totalmente conectadas del final (clasificación) para adaptarlas a nuestro

problema. Por tanto, a la hora de aplicar esta técnica, se nos plantean dos grandes alternativas:

1. Extracción de características: usamos lo aprendido por la red anterior para extraer características significativas de nuevas muestras. Simplemente agregamos un nuevo clasificador, que se entrenará desde cero, sobre el modelo pre-entrenado para que podamos reutilizar los mapas de características aprendidos previamente en nuestro conjunto de datos.
2. Ajuste fino (Fine-Tunning): descongelamos algunas de las capas superiores del modelo base utilizadas para la extracción de características, y entrenamos conjuntamente las capas de clasificación recién agregadas, así como las últimas capas del modelo congelado. Esto nos permite “afinar” la representación de características, además de nuestro clasificador final, para hacerlas más relevantes para nuestro problema.

Para la red base se suelen utilizar redes profundas pre-entrenadas para tareas de clasificación de imágenes. Sobre todo, es bastante común utilizar redes desarrolladas para la competición de ImageNet. Las organizaciones de investigación que participan en esta competición a menudo lanzan sus modelos bajo una licencia abierta para su reutilización. Debemos descargarlos e incorporarlos a nuestro problema.

Esta técnica proporciona las siguientes ventajas:

- Nos permite ahorrar tiempo de entrenamiento. Las redes neuronales profundas pueden tardar semanas en entrenarse cuando los conjuntos de datos son de gran tamaño. Con Transfer Learning solo se entrena unas pocas capas, por lo que el proceso es mucho más rápido.
- Nos permite utilizar técnicas de aprendizaje profundo incluso si no disponemos de una gran cantidad de datos. Las redes neuronales convolucionales tienen una enorme cantidad de parámetros a aprender para los que se requieren ingentes cantidades de muestras. Con Transfer Learning, puesto que solo entrenamos unas pocas capas, el número de parámetros se reduce drásticamente y en consecuencia, también el número de datos necesarios.

Debido especialmente al segundo punto, puede resultar útil aplicar esta técnica a nuestro problema. Disponemos de aproximadamente 15000 muestras etiquetadas (lo que es una cantidad relativamente baja para un problema de deep learning). Por otro lado, el uso de Transfer Learning en el campo de la medicina y la patología digital está ampliamente extendido, por lo que puede ser interesante ver qué resultados se obtienen.

De todas las arquitecturas pre-entrenadas que podemos encontrar, nos hemos decantado por dos:

Red VGG16

Desarrollada por la universidad de Oxford y entrenada en ImageNet. Se trata de un modelo que destaca por ser una red convolucional pura con una potente extracción de características gracias a sus repetidas y múltiples convoluciones con filtros pequeños (Figura 2.3).



Figura 2.3: Diagrama con la arquitectura de la red VGG16.

Las 4 capas superiores son completamente desechadas y sustituidas por una capa totalmente conectada de 128 neuronas (con ReLU como función de activación, dropout y penalización L2), seguida de un nuevo clasificador softmax (véase la Figura 2.4). Tras la última convolución, para una entrada de tamaño 32x32, el vector de características tiene un tamaño de 512. Por este motivo se introduce una capa de 128 neuronas, para reducir su dimensionalidad de cara al softmax. Esto nos deja una red con 20 capas, 8 más que nuestra arquitectura propia. También cabe destacar que la red original está pensada para trabajar con imágenes de 3 canales, por tanto, resulta imposible utilizar la segunda escala.

Por último, para esta arquitectura se aplicarán las dos técnicas de Transfer Learning disponibles: extracción de características y ajuste fino.

Red InceptionV3

InceptionV3 es un modelo de reconocimiento de imágenes ampliamente utilizado, el cual alcanzó un accuracy superior al 78.1% en el conjunto de datos de ImageNet. Está formado por bloques de construcción simétricos y asimétricos que incluyen convoluciones, reducción promedio, reducción máxima, concatenaciones, retirados y capas completamente conectadas. La normalización por lotes se usa con frecuencia en todo el modelo (aplicada a las entradas de activación) y las pérdidas se calculan a través de Softmax.

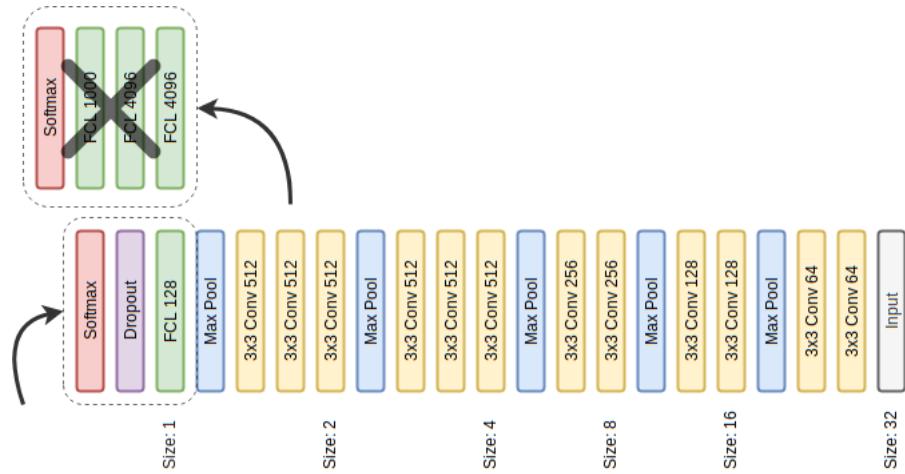


Figura 2.4: Adaptación de la red VGG16 al problema. Las capas más altas se sustituyen por otras. Los tamaños se ajustan a la entrada.

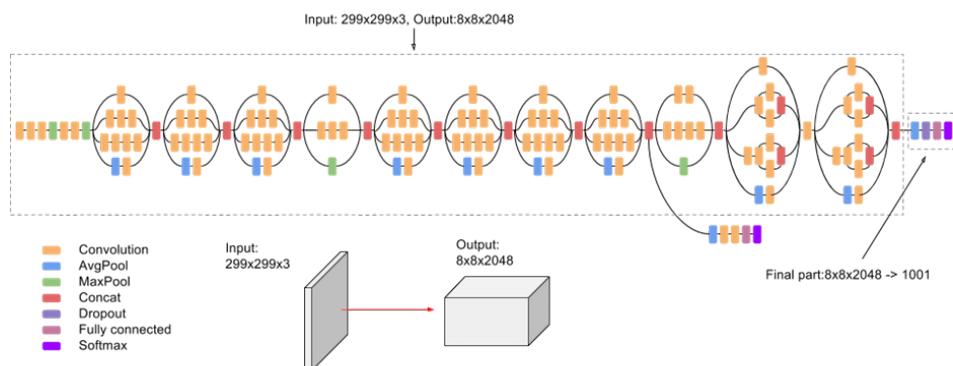


Figura 2.5: Diagrama con la arquitectura de la red InceptionV3.

Como puede observarse en la Figura 2.5, se trata de una red de una profundidad y complejidad mucho mayores que el resto de modelos descritos. Al igual que con VGG, las 4 capas nombradas como “Final part” en la figura son descartadas y sustituidas por una capa totalmente conectada, dropout, penalización L2 y un nuevo Softmax. En este caso, el nuevo FCL incorpora un total de 512 neuronas, pues tras la última concatenación, el vector de características presenta un tamaño de 2048.

Con respecto a la entrada, el tamaño de imagen mínimo admitido por la red es de 75x75 con 3 canales, por lo que nuevamente se prescindirá de la segunda escala y será necesario realizar un reescalado de los parches. Finalmente, con respecto a las técnicas de transfer learning, únicamente se entrenarán las capas del nuevo clasificador final.

2.2. Tratamiento de las imágenes

Para cualquier problema de aprendizaje (ya sea automático o profundo), los datos son de vital importancia y en la mayoría de los casos, la disponibilidad y el etiquetado de los mismos determinan las técnicas que podemos emplear.

En nuestro caso, disponemos de una base de datos de preparaciones histológicas (digitalizadas mediante un escáner) de biopsias de aguja de la próstata, cedidas por un hospital de Valencia a la Universidad de Granada para el proyecto *Sistema de interpretación de imágenes histopatológicas para la detección de cáncer de próstata* (SICAP) [22], y extraídas con el objetivo de determinar mediante técnicas de patología digital si una persona padece o no cáncer de próstata. Las imágenes están divididas en benignas (no padece síntomas preocupantes) y malignas (de los grados 3 al 5). Como ya adelantamos anteriormente, nosotros nos centraremos en la benignas. Las imágenes están etiquetadas en función de los distintos grados de cáncer, pero no están etiquetadas para nuestro problema, es decir, ninguna contiene marcadas aquellas regiones que son células glandulares. Por tanto, tendremos que realizar nosotros mismos el etiquetado.

Dado el tiempo que tenemos para esta fase según la planificación establecida, debemos elegir un método de etiquetado que sea lo más rápido y efectivo posible. De entre las distintas técnicas que existen, una de las que se ajusta a este criterio es aquella consistente en trocear la imagen en un conjunto de parches cuadrados (con o sin solapamiento) y asignarle a todos los píxeles contenidos en el parche la misma etiqueta. Por tanto, aparecen dos parámetros que debemos fijar: el tamaño del parche y el solapamiento o separación entre ellos (Figura 2.6).

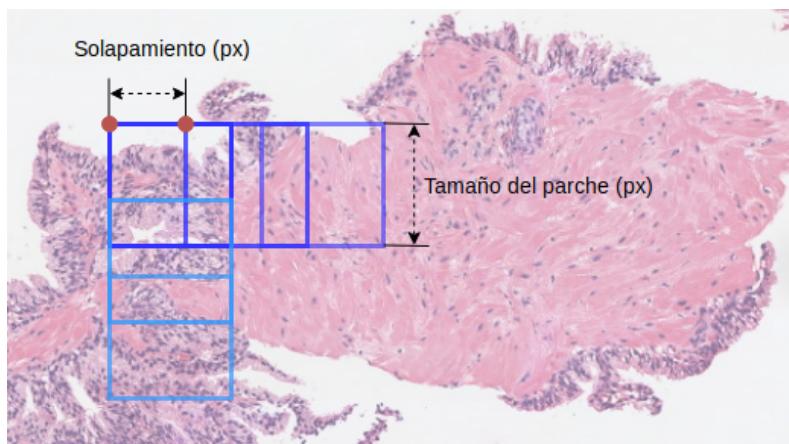


Figura 2.6: Visualización de los conceptos de tamaño de parche y solapamiento.

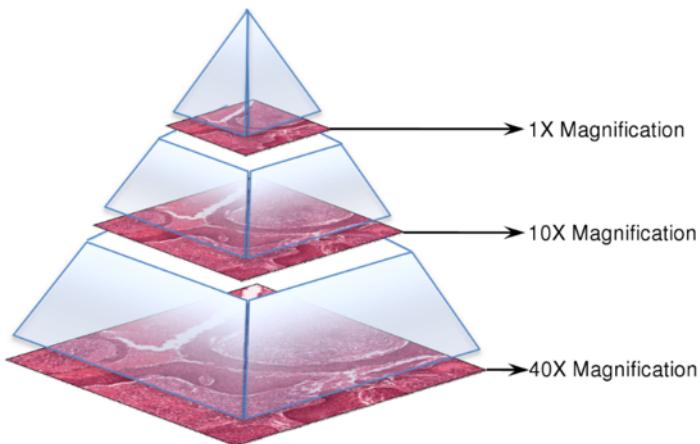


Figura 2.7: Pirámide de resoluciones. Incorpora distintas escalas de una misma imagen.

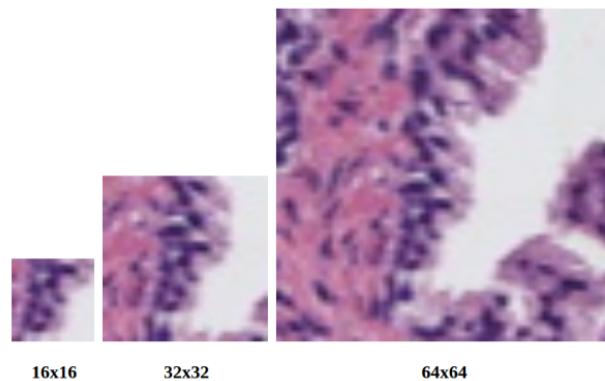
Por otro lado, no debemos olvidar que las imágenes son WSI, es decir, contienen varias escalas de una misma imagen conformando una pirámide de resoluciones (Figura 2.7). Concretamente, presentan una magnificación de 40x y el alto y ancho de cada píxel equivale a $0.25 \mu m$ en la muestra real. Esto nos lleva a que la imagen más grande del conjunto, en la mayor escala, tiene unas dimensiones de 64752×141984 píxeles (lo que nos deja un total de aproximadamente $9 \cdot 10^9$ píxeles). Es prácticamente imposible que la red pueda procesar en un tiempo razonable imágenes tan grandes, además de que el tamaño de los parches tendría que ser bastante elevado para que contengan información relevante. Es necesario llevar a cabo un submuestreo de las imágenes (escoger una escala más pequeña) para reducir su dimensionalidad y hacerlas más manejables. En resumen, tenemos tres parámetros que fijar:

- El factor de submuestreo para las imágenes, es decir, la escala a la que las procesaremos.
- El tamaño de los parches.
- El solapamiento (stride) entre parches consecutivos.

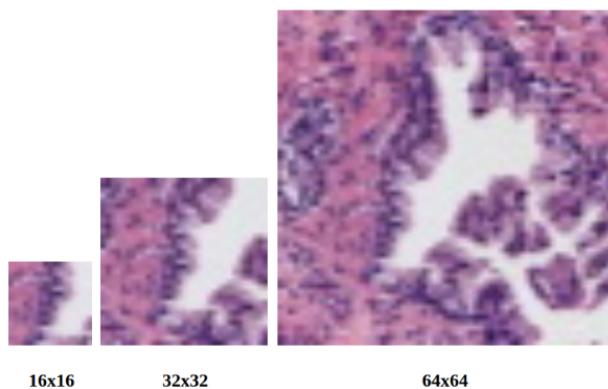
Para escoger sus valores, se probaron una serie de combinaciones. Se resumen en la Figura 2.8.

En principio, es deseable que los parches se ajusten lo máximo posible a los bordes de las glándulas, por lo que no nos interesan tamaños ni demasiado grandes, ni demasiado pequeños. De las combinaciones probadas, las que más se ajustan a estos requerimientos son 32x32 con un submuestreo de 8 (5x) y 16x16 con un submuestreo de 16 (2.5x). Finalmente, nos hemos decantado

por utilizar parches de 32x32 con un submuestreo de 8. Con respecto a la región que ambos representan no existe mucha diferencia, pero la pérdida de información es mucho menor en los parches de 32 (véase la diferencia de tamaño entre uno y otro en la Figura 2.8).



(a) Distintos tamaños de parche con un factor de submuestreo de 8 (escala 5x).



(b) Distintos tamaños de parche con un factor de submuestreo de 16 (escala 2.5x).

Figura 2.8: Combinaciones probadas para el tamaño de parche y la escala a la que se procesarán las imágenes.

Por último, debemos determinar el solapamiento que existirá entre los parches. Este será fijado a 16 píxeles, de esta forma cada parche estará solapado por la mitad con sus adyacentes, o dicho de otra forma, cada píxel pertenecerá a 4 parches diferentes (Figura 2.9), lo cuál será útil a la hora de determinar qué clase le corresponde.

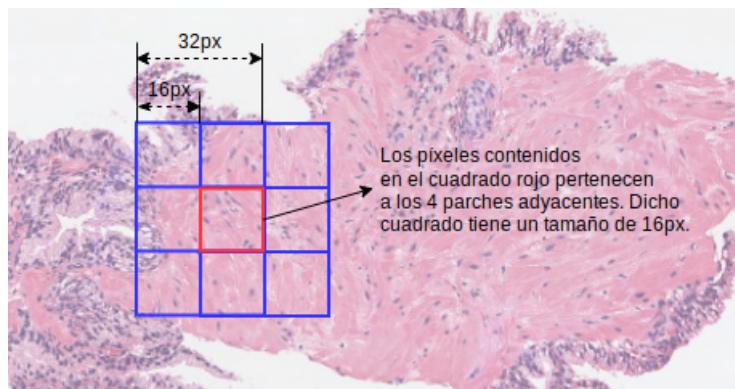


Figura 2.9: Efecto del solapamiento escogido. En la imagen se muestran 4 parches solapados.

2.2.1. Eliminación del fondo

Todas las preparaciones histológicas escaneadas presentan un fondo blanco/grisáceo (Figura 2.10). Dicho fondo es completamente irrelevante para el problema, ya que procesar cientos de parches blanquecinos que a priori se conoce con certeza que no son glándula, afectaría de forma negativa a la eficiencia. En definitiva, es necesario idear un mecanismo para eliminarlo y teniendo en cuenta el contraste existente entre el tejido y el fondo, podemos abordar esta supresión en una etapa de preprocesado y utilizando técnicas de filtrado de imágenes.

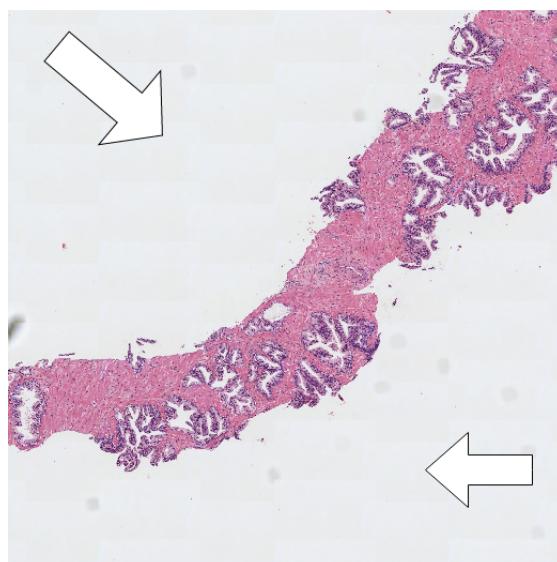


Figura 2.10: Fondo grisáceo de las imágenes.

Concretamente, se utilizará una máscara binaria del mismo tamaño que la imagen. Los píxeles del fondo se mapearan al negro, mientras que los pertenecientes al tejido, se asignarán al blanco. El proceso seguido para la creación de esta máscara es el siguiente (Figura 2.11):

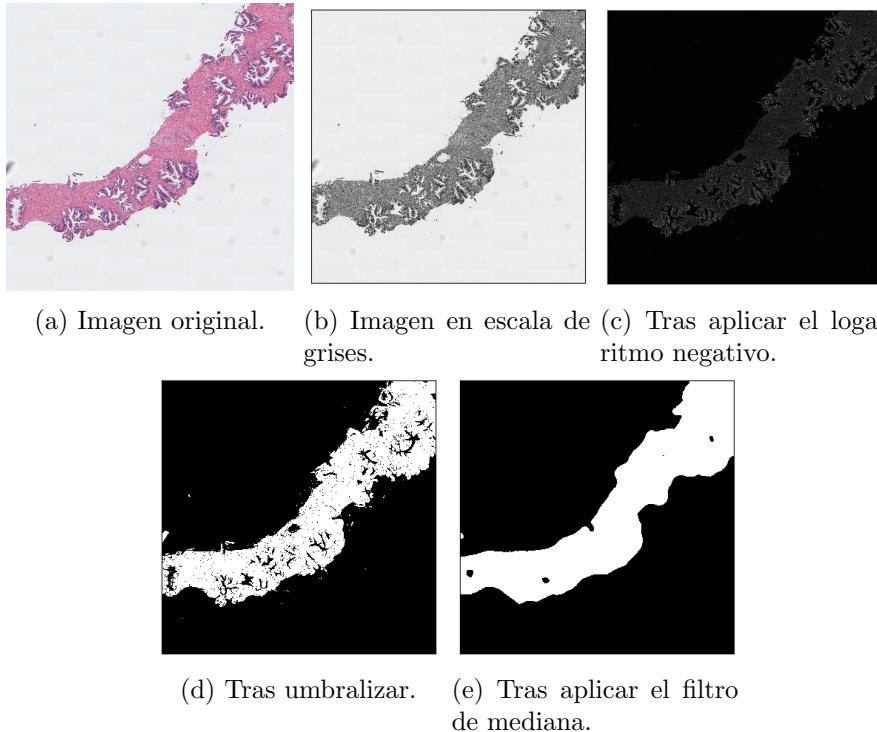


Figura 2.11: Proceso de construcción de la máscara.

1. Convertir la imagen de RGB a escala de grises, quedándonos únicamente con el canal verde.
2. Convertir la imagen de entero sin signo de 8 bits a flotante normalizado en el intervalo [0,1]. Para ello basta con dividir la intensidad de cada píxel entre 255 y realizar una conversión de tipos.
3. Tomar $-\log(x)$, donde x es la intensidad de cada uno de los píxeles. De esta forma conseguimos que los valores de intensidad cercanos a 1 (blancos, como el fondo) se conviertan en valores cercanos a 0 (negro) y viceversa. En definitiva, invierte las intensidades.
4. Umbralizar: tras el paso anterior seguimos teniendo una imagen en escala de grises. Es necesario convertirla a binaria (solo blanco y negro). Para ello, todos los valores inferiores a un umbral serán considerados como 0 y los mayores como 1. Como valor de corte se ha escogido 0.25 mediante ensayo y error.

5. Eliminar el ruido sal y pimienta resultante mediante un filtro de mediana.

Finalmente, y una vez construida la máscara, en el proceso de extracción de parches solo se almacenarán aquellos que estén ubicados dentro de la parte blanca de la máscara. Los demás serán ignorados.

2.2.2. Etiquetas

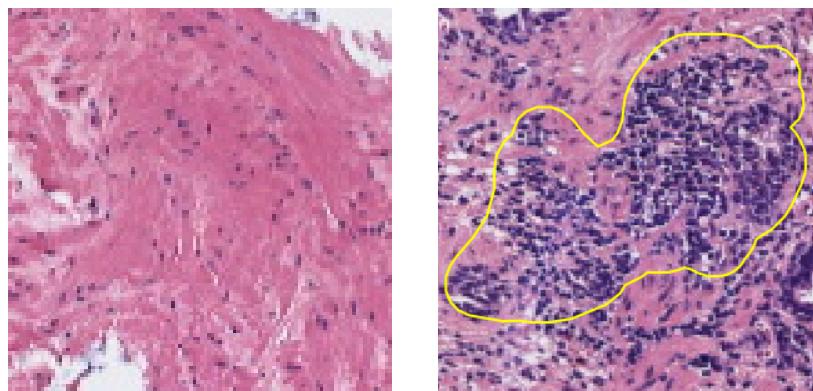
Resulta necesario establecer cuáles serán las distintas etiquetas que se asignarán a los parches durante el proceso de etiquetado. En un principio, parece que está bastante claro que se trata de un problema binario:

- 1 → el parche pertenece a una célula glandular.
- 0 → el parche no pertenece a una célula glandular (normalmente, al tejido que rodea a las células de un órgano se le conoce como estroma, por lo que de ahora en adelante nos referiremos a esta clase como estroma).

Ahora bien, identificar un parche que contiene una célula glandular no presenta demasiado problema, identificar uno de estroma tampoco. El verdadero problema surge cuando tenemos un parche que está a caballo entre ambos, que no es ni estroma “puro” ni llega a ser una célula glandular. A este tipo de parches le asignaremos informalmente la etiqueta “estroma contaminado”. En la Figura 2.12 puede observarse un ejemplo del mismo: no presenta la morfología propia de una glándula, pero tampoco es una zona de la que se pueda decir con total seguridad que es estroma. En definitiva, se trata de una etiqueta comodín y situaremos en ella todos aquellos parches de los que no estemos completamente seguros de su naturaleza.

En resumen, nuestro dataset no será binario, sino que tendremos parches correspondientes a tres etiquetas:

- 0 → Estroma puro.
- 1 → Célula glandular.
- 2 → “Estroma Contaminado”



(a) Estroma “puro”. Prácticamente no presenta células.

(b) Sección de “estroma contaminado”.

Figura 2.12: Comparativa entre estroma “puro” y “contaminado”.

2.2.3. Procesamiento a dos escalas

Las células glandulares presentan tamaños muy variados, por lo que en principio, no existe un tamaño de parche/escala que sea válido para todas ellas (Figura 2.13). En consecuencia, parece razonable no limitarnos a una sola escala, y realizar un procesamiento empleando varias de ellas.

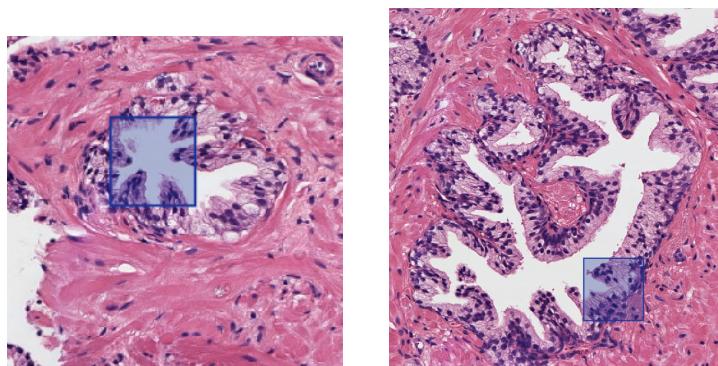


Figura 2.13: Mismo parche, distintos tamaños de glándula.

Para tal fin, se nos plantean dos alternativas:

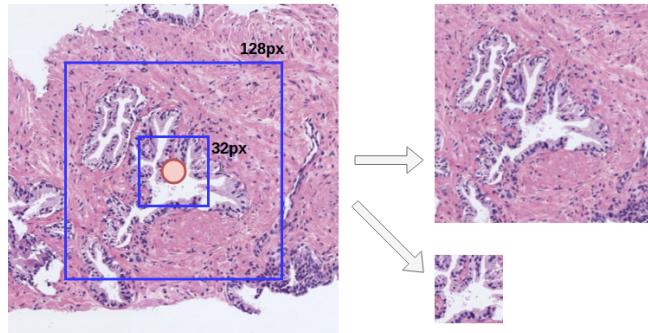
1. Utilizar otros niveles de magnificación de las imágenes WSI.
2. Utilizar otros tamaños de parche.

Por mayor comodidad, hemos optado por la segunda. Concretamente, además de los parches de 32x32 ya explicados antes, se plantea añadir un

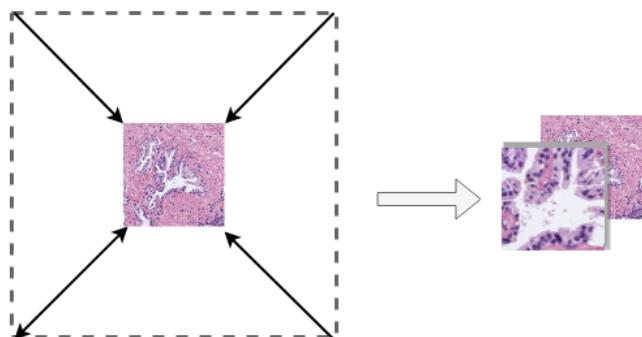
segundo tamaño de 128x128 (equivalente a una resolución más pequeña de la imagen). De esta forma, el modelo cubre un amplio espectro de proporciones.

Ahora, la arquitectura recibirá como entrada parches de tamaño 32x32x6. Los tres primeros canales pertenecen a la escala de 32x32, mientras que los tres últimos pertenecen a la escala de 128x128 (reescalados a 32 para que la concatenación sea posible). Por último, el procedimiento seguido para obtener el parche de tamaño 128x128 a partir de uno de tamaño 32x32 es el siguiente (Figura 2.14):

- Calcular el centro del parche de tamaño 32x32.
- Partiendo de dicho centro, extraer el parche de tamaño 128x128 de tal manera que los centros de ambos coincidan.
- Reescalar el parche de tamaño 128 a 32 píxeles.
- Concatenar ambos parches en la dimensión de los canales: $(32,32,3) + (32,32,3) = (32,32,6)$.



(a) Se extraen ambos parches con el mismo centro.



(b) El más grande se reescala a 32 y ambos se concatenan en la tercera dimensión.

Figura 2.14: Proceso de extracción de las dos escalas.

2.3. El problema del desbalanceo

Tas efectuar el proceso de etiquetado (el cual se describe con detalle en el siguiente capítulo) el número de parches pertenecientes a la clase estroma es prácticamente el doble que la suma de las otras dos clases (Figura 2.15). Evidentemente, no se trata de un problema donde la cantidad de muestras de cada clase sea más o menos la misma. Decimos entonces que los datos están desbalanceados.

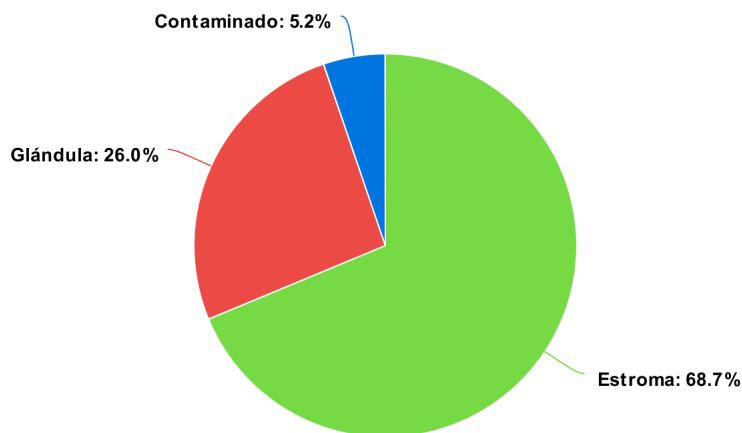


Figura 2.15: Comparativa de la cantidad de parches de cada clase.

Para solucionar este problema, asignaremos pesos a cada una de las clases e incorporaremos dichos pesos a la función de error. Esta es una técnica que nos permite afrontar el desbalanceo sin la necesidad de remuestrear los datos. La idea es que la clase con menos representación tenga un mayor peso que el resto. Concretamente, utilizaremos la siguiente fórmula para calcular el peso de cada una de ellas:

$$\text{ClassWeight}(A) = \frac{N}{N_{\text{classes}} \cdot \text{Count}(A)} \quad (2.1)$$

Por otro lado, el desbalanceo también nos influirá a la hora de decidir cuales serán las métricas que utilizaremos para estimar el error de nuestro modelo.

Las métricas más utilizadas en el deep learning son la entropía cruzada (Ecuación 2.6) y el accuracy (Ecuación 2.4). Sin embargo, esta última no es apropiada para datos desbalanceados. Supongamos que tenemos un problema de clasificación binaria, donde hay 80 muestras de la clase 0 y 20 de la clase 1, esto nos deja un desbalanceo de 80:20, que es lo mismo que 4:1. Si nuestro clasificador predice siempre la clase 0, acertará en un 80 % de las

veces, es decir, tendrá un accuracy del 0.8. Puede parecer un buen resultado, pero realmente el clasificador no está comportandose como debería, pues esta obviando por completo la clase 1.

Por este motivo no es recomendable utilizar la métrica Accuracy en casos de desbalanceo. En su lugar utilizaremos la métrica F1 Score para estimar los distintos hiperparámetros. Esta métrica puede interpretarse como la media ponderada entre precision (la habilidad del clasificador de no etiquetar como positiva una muestra negativa) y recall (la capacidad del clasificador de encontrar todas las muestras positivas), por lo que tiene en cuenta todas las clases. Alcanza su mejor valor en 1 y el peor en 0.

$$Precision = \frac{TP}{TP + FP} \quad (2.2)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.3)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.4)$$

$$F1\ Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (2.5)$$

Donde TP son los verdaderos positivos, TN verdaderos negativos, FP falsos positivos y FN falsos negativos.

Entropía cruzada (cross-entropy) para el caso binario:

$$-(y \log(p) + (1 - y) \log(1 - p)) \quad (2.6)$$

donde:

\log = logaritmo natural.

p = probabilidad predicha de que una muestra m sea de una clase c .

y = indicador binario (0 o 1) si la clase c es la clasificación correcta para la muestra m .

En caso de tener un problema con múltiples clases ($M > 2$), se calcula el error por separado para cada una de ellas y se suman los resultados:

$$\sum_{c=1}^M y_{m,c} \log(p_{m,c}) \quad (2.7)$$

2.4. Detección de glándulas

El procedimiento que efectuará el algoritmo final es el siguiente: dividir la imagen de entrada en parches de tamaño 32x32 con un solapamiento de 16 píxeles, hacer que el modelo le asigne una etiqueta de las indicadas en la sección 2.2.2 a cada uno de ellos y por último, dibujar un contorno alrededor de cada célula glandular a partir de los resultados de la clasificación. En esta sección se describe este último paso.

Recordemos que el clasificador le asigna la misma probabilidad de ser célula glandular a todos los píxeles contenidos en un parche. Como ya se ha explicado anteriormente, si tenemos parches de 32x32 con un solapamiento de 16 píxeles significa que cada píxel pertenece a 4 parches diferentes, por lo que tendrá 4 probabilidades distintas (Figura 2.9). Por tanto, parece claro que lo primero que necesitamos determinar es la forma en la que combinaremos las 4 probabilidades en una sola. Para ello, nos hemos decantado por realizar la media aritmética.

Una vez hecho esto, nuestra imagen es un mapa de probabilidades: cada píxel tiene asociado un único valor que determina la probabilidad de ser o no célula glandular. Ahora solamente necesitamos un umbral de corte, un valor a partir del cual podemos considerar que la probabilidad de ser glándula es lo suficientemente elevada.

Precisamente por este motivo, calcularemos la curva ROC durante la validación cruzada del modelo. Dicha curva presenta el ratio de verdaderos positivos (tpr) en el eje Y, y el ratio de falsos positivos (fpr) en el eje X. Esto significa que la esquina superior izquierda de la gráfica es el punto “ideal”: un ratio de falsos positivos de cero y un ratio de verdaderos positivos de uno. Este punto no es demasiado realista, pero sí que podemos utilizar como umbral de corte el umbral asociado al punto de la curva más cercano a la esquina superior izquierda.

La función de Scikit-learn que calcula la curva ROC nos devuelve un array con todos los umbrales probados, junto con el tpr y fpr para cada uno de ellos. Por tanto, lo único que necesitamos es una fórmula con la que poder calcular cuál es la pareja [fpr,tpr] más cercana a la esquina superior izquierda y devolver el umbral asociado. Dicha fórmula la podemos encontrar en el siguiente artículo de Medical Biostatistics [14]:

$$\sqrt{(1 - tpr)^2 + (fpr)^2} \quad (2.8)$$

Seleccionaríamos el umbral asociado a la pareja [fpr, tpr] que minimiza dicha ecuación.

Para que los resultados sean más robustos, hemos calculado el umbral

óptimo para la curva ROC de cada una de las particiones de la validación cruzada. Finalmente, el valor de corte final será la media de todos ellos: 0.081414 ± 0.053300 .

Tras umbralizar tendremos una máscara binaria: 1 el píxel es glándula, 0 no lo es. Con esta máscara y aplicando funciones de OpenCV [16], podemos obtener el contorno de cada una de las componentes conexas.

Para una mayor claridad en la explicación, en la Figura 2.16 puede observarse un diagrama con los distintos pasos (y sus resultados) del procedimiento descrito en esta sección.

2.5. Algoritmo final

Llegados a este punto, ya se han explicado con detalle todas las consideraciones a tener en cuenta para construir el algoritmo final que efectuará la detección de las glándulas. A continuación se describe el mismo utilizando un lenguaje pseudoalgorítmico.

Algoritmo 1: Detección de glándulas.

Entrada: Una imagen *img* submuestreada con un factor de 8.

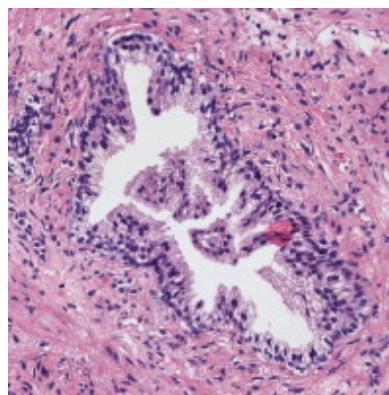
Salida: Una nueva imagen *out* que contiene dibujada la detección.

inicio

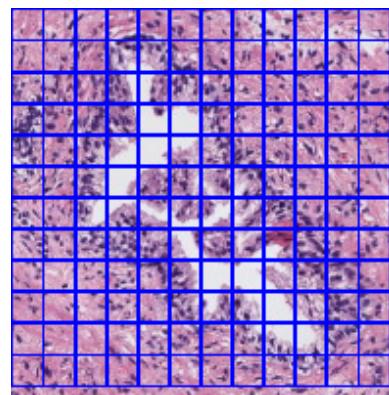
```

    modelo ← CargarModelo();
    /* Crear la máscara para eliminar el fondo */ 
    máscaraFondo ← CrearMáscaraFondo(img);
    /* Tamaño = 32, solapamiento = 16: */
    parches ← ExtraerParches(img, 32, 16, máscaraFondo);
    /* La segunda escala de tamaño 128 */
    parches' ← AñadirEscalaExtra(parches, img, 128);
    /* Restando la media y dividiendo entre la std */
    parches" ← Normalizar(parches');
    probParches ← modelo.Predecir(parches");
    /* Ahora se dibujan los contornos */
    probPíxeles ← CombinarPromediando(probParches);
    máscara ← probPíxeles > UMBRAL;
    out ← DibujarContornos(img, máscara);
    devolver out
fin

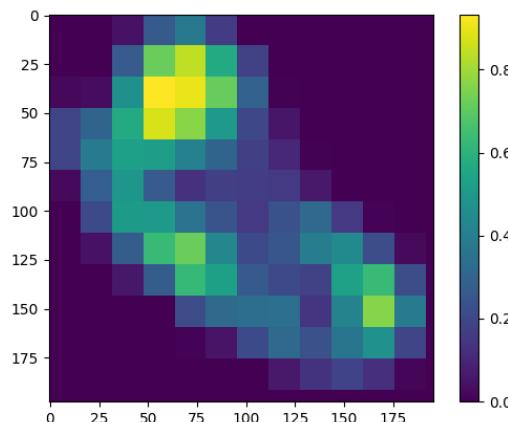
```



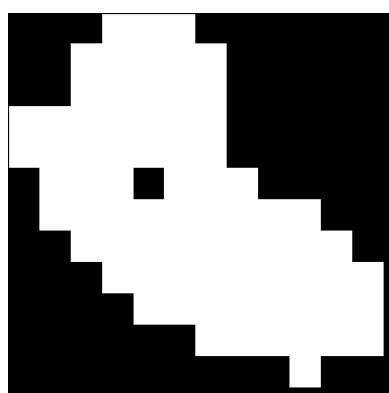
(a) Imagen de entrada. Contiene una célula glandular.



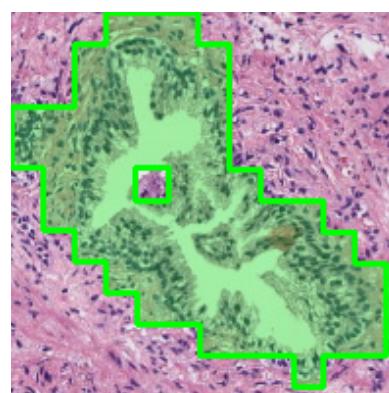
(b) La imagen se divide en parches de tamaño 32x32 con un sołapamiento de 16px. También se extraen los parches de 128x128.



(c) El modelo le asigna una probabilidad de ser glándula a cada parche. Se combinan utilizando la media aritmética, dando lugar a este mapa de probabilidades.



(d) Todos los píxeles con una probabilidad superior al umbral son considerados glándula (en blanco).



(e) Por último, se dibujan los contornos a partir de la máscara binaria.

Figura 2.16: Ilustración del proceso de detección.

Capítulo 3

Proceso de etiquetado

Con el fin de hacer el proceso de etiquetado más eficiente y menos tedioso, nos ayudaremos en el mismo tanto del software de patología digital de código abierto QuPath [3], como de la arquitectura propia descrita en el capítulo anterior (subsección 2.1.1). De forma esquemática, la metodología seguida es la siguiente:

1. Dividir la imagen a etiquetar en parches de 32x32 sin solapamiento (es decir, con un stride igual al tamaño del parche). El motivo por el que se eliminó el solapamiento en este paso es porque, en caso de haberlo, no resultaba posible importar el resultado en QuPath (la imagen contenía demasiados parches y el programa se cerraba inesperadamente). Esto obviamente tendrá su consecuencia negativa a la hora del entrenamiento, pues estamos perdiendo un número bastante elevado de parches.
2. Hacer que el clasificador le asigne una etiqueta a cada parche.
3. Importar los resultados de la clasificación en QuPath.
4. Corregir manualmente en QuPath los parches mal etiquetados. Tras este paso ya tenemos etiquetada una imagen más.
5. Exportar los parches corregidos a un fichero de texto que pueda ser procesado por Python y alimentar al clasificador con los nuevos datos corregidos (para refinarlo).
6. Repetir desde el paso 1 para la siguiente imagen.

Este proceso implica que al menos una imagen debe ser etiquetada manualmente (ya que para poder utilizar el clasificador deberá entrenarse previamente con al menos una de ellas). Para este etiquetado manual sí se

Imagen	Estroma	Glándula	Contaminado
1	899	487	113
2	1103	451	42
3	1092	338	63
4	770	266	64
5	628	414	72
6	1276	518	116
7	997	306	17
8	366	290	23
9	1025	287	130
10	1003	298	99
11	1406	347	64
Total por clase	10565	4002	803
Total global	15370		

Cuadro 3.1: Sumario sobre el tamaño del Dataset.

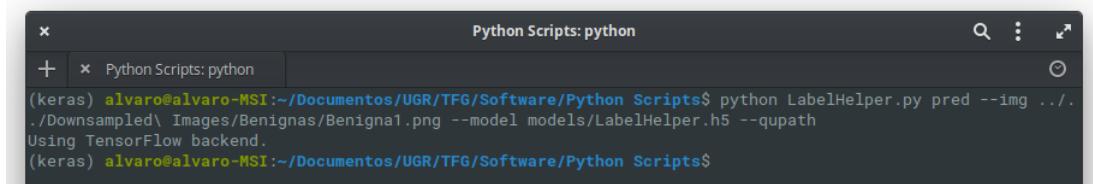
empleó el solapamiento. Simplemente se visualizaron los parches uno por uno y se escribió su correspondiente etiqueta.

Es muy importante aclarar que el objetivo de esta fase no es ajustar el clasificador, sino mostrarle unas cuantas imágenes para ayudarnos en el etiquetado. Dicho de otro modo, el clasificador entrenado no será el que se emplee en el algoritmo final, ni tampoco se ha puesto mucho esfuerzo en su entrenamiento. Concretamente, para cada alimentación del clasificador, este se entrenó con los parámetros por defecto de Keras, 25 épocas, aplicando data augmentation a todos los datos y sin utilizar la doble escala.

El script que lleva a cabo esta tarea es LabelHelper. Puede utilizarse desde la línea de comandos y nos permite tanto realizar las predicciones de una imagen y escribirlas en un fichero de texto (que será procesado por QuPath) como alimentar un clasificador previamente entrenado con nuevas imágenes, actualizando sus pesos. También fue necesario desarrollar dos scripts para QuPath: uno para importar las predicciones del clasificador y otro para exportar las etiquetas corregidas como fichero de texto.

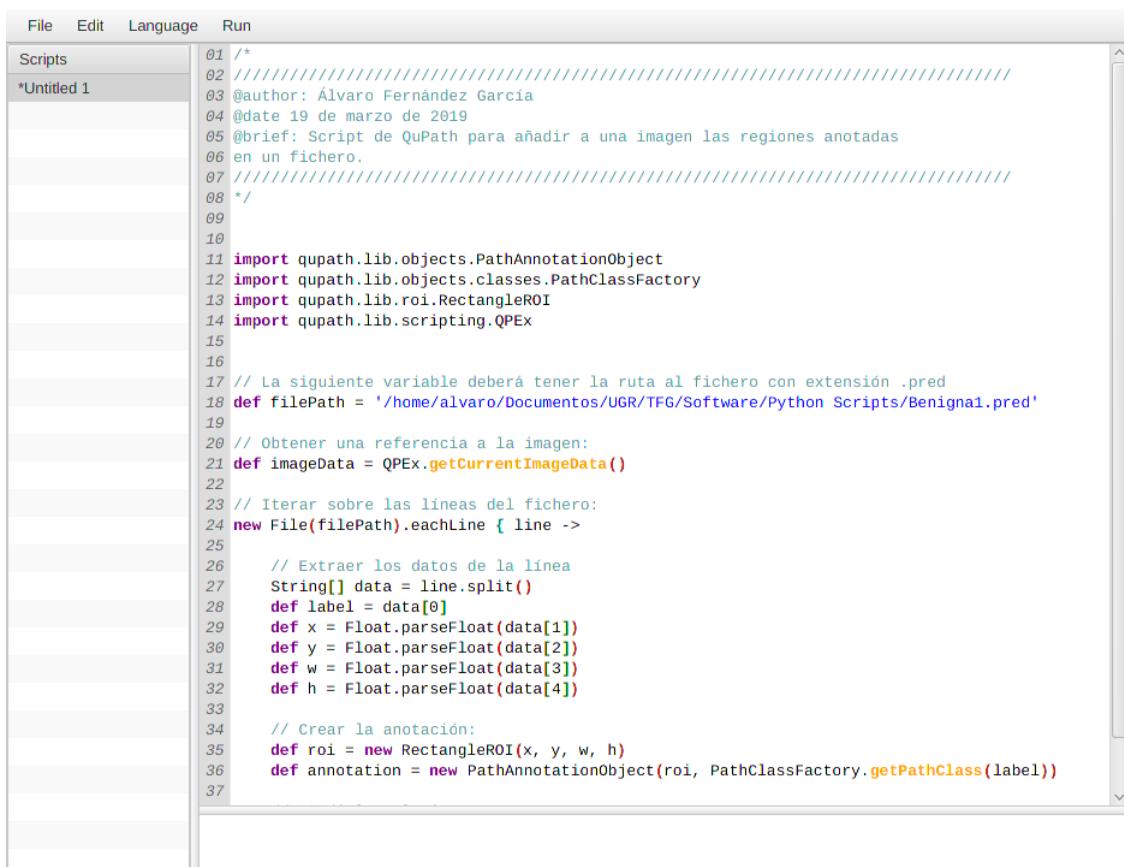
Gracias a este procedimiento, se etiquetaron un total de 11 imágenes benignas. En el Cuadro 3.1 se muestra un sumario con el total de parches extraídos por imagen, separados por clase.

Para finalizar, a continuación se muestran algunas capturas de pantalla que ilustran el procedimiento.



```
(keras) alvaro@alvaro-MSI:~/Documentos/UGR/TFG/Software/Python Scripts$ python LabelHelper.py pred --img ../../Downsampled\Images/Benignas/Benigna1.png --model models/LabelHelper.h5 --qupath
Using TensorFlow backend.
(keras) alvaro@alvaro-MSI:~/Documentos/UGR/TFG/Software/Python Scripts$
```

Figura 3.1: Se divide la imagen de entrada en parches, el clasificador les asigna una etiqueta y se exportan los resultados a un fichero de texto con extensión .pred.



```
File Edit Language Run
Scripts Untitled 1
01 /*
02 //////////////////////////////////////////////////////////////////
03 @author: Álvaro Fernández García
04 @date 19 de marzo de 2019
05 @brief: Script de QuPath para añadir a una imagen las regiones anotadas
06 en un fichero.
07 //////////////////////////////////////////////////////////////////
08 */
09
10
11 import quupath.lib.objects.PathAnnotationObject
12 import quupath.lib.objects.classes.PathClassFactory
13 import quupath.lib.roi.RectangleROI
14 import quupath.lib.scripting.QPEx
15
16
17 // La siguiente variable deberá tener la ruta al fichero con extensión .pred
18 def filePath = '/home/alvaro/Documentos/UGR/TFG/Software/Python Scripts/Benigna1.pred'
19
20 // Obtener una referencia a la imagen:
21 def imageData = QPEx.getCurrentImageData()
22
23 // Iterar sobre las líneas del fichero:
24 new File(filePath).eachLine { line ->
25
26     // Extraer los datos de la línea
27     String[] data = line.split()
28     def label = data[0]
29     def x = Float.parseFloat(data[1])
30     def y = Float.parseFloat(data[2])
31     def w = Float.parseFloat(data[3])
32     def h = Float.parseFloat(data[4])
33
34     // Crear la anotación:
35     def roi = new RectangleROI(x, y, w, h)
36     def annotation = new PathAnnotationObject(roi, PathClassFactory.getPathClass(label))
37 }
```

Figura 3.2: Este script de QuPath lee el archivo con extensión .pred (indicado en la variable filePath) e importa los parches y sus correspondientes etiquetas al programa.

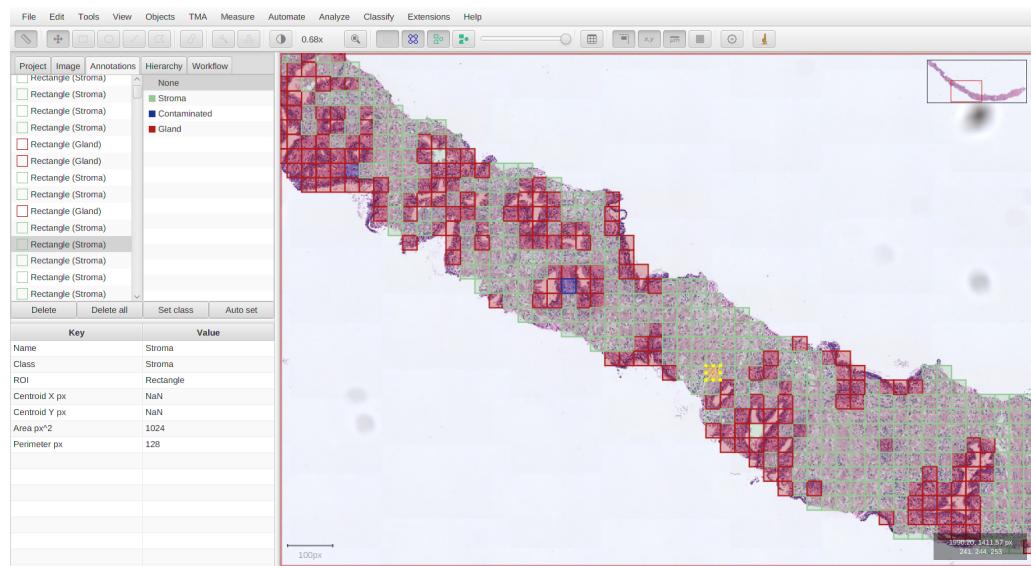


Figura 3.3: Así se ven los parches importados en QuPath. A continuación, se corrigen las etiquetas erróneas y con un segundo script se exportan a un fichero de texto con extensión .qpdat.

```
* python LabelHelper.py feed --imgs ../../Downsampled\ Images/Benignas/ --qpf QupathFiles/ --model models/LabelHelper.h5
+ ...dels/LabelHelper.h5
(keras) alvaro@alvaro-MSI:~/Documentos/UGR/TFG/Software/Python Scripts$ python LabelHelper.py feed --imgs ../../Downsampled\ Images/Benignas/ --qpf QupathFiles/ --model models/LabelHelper.h5
Using TensorFlow backend.
Epoch 1/25
3082/3082 [=====] - 8s 3ms/step - loss: 0.2152 - categorical_accuracy: 0.9611
Epoch 2/25
3082/3082 [=====] - 7s 2ms/step - loss: 0.1667 - categorical_accuracy: 0.9718
Epoch 3/25
3082/3082 [=====] - 7s 2ms/step - loss: 0.1574 - categorical_accuracy: 0.9643
Epoch 4/25
3082/3082 [=====] - 7s 2ms/step - loss: 0.1540 - categorical_accuracy: 0.9708
Epoch 5/25
1920/3082 [=====>.....] - ETA: 2s - loss: 0.1398 - categorical_accuracy: 0.9714
```

Figura 3.4: A partir de la imagen y el fichero .qpdat, podemos refinar los pesos de un clasificador cualquiera pasado como argumento.

Capítulo 4

Experimentos

En este capítulo se describirán con detalle los distintos experimentos realizados junto a sus resultados. El objetivo de los mismos es la obtención y validación del modelo de aprendizaje profundo que será empleado en el algoritmo final para efectuar la detección de glándulas, tal y como se explicó en el capítulo 2. De forma abstracta, los experimentos efectuados son los siguientes:

- Sobre la arquitectura propia: en primer lugar, se han realizado un par de experimentos con 7 imágenes y una partición aleatoria de validación para fijar hiperparámetros (learning rate inicial y penalización L2 para las capas totalmente conectadas). Hecho esto, a continuación se ha llevado a cabo validación cruzada con las 11 imágenes y 5 particiones, aplicando la doble escala y learning rate decay durante 100 épocas en una versión binaria del problema. Este último experimento se ha realizado con dos valores diferentes para la probabilidad de dropout (0.25 y 0.5) y también empleando Batch Normalization en las capas de convolución.
- Transfer Learning: con la arquitectura VGG16 se han realizado dos experimentos: entrenar únicamente las últimas capas, manteniendo lo aprendido para la convolución (durante 50 épocas y aplicando learning rate decay) y realizar Fine Tuning de las tres últimas capas de convolución durante 25 épocas. Con la red InceptionV3, solamente se han entrenado las últimas capas bajo las mismas condiciones que para VGG.

De todos los modelos probados, el que mostró un mejor desempeño fue la arquitectura de diseño propio con dropout de 0.25 y sin Batch Normalization (0.89 de F1 Score en validación). Finalmente, con este mismo modelo, se realizó un último experimento para evaluar su rendimiento en la detección

de glándulas: de 4 imágenes benignas distintas a las empleadas en el entrenamiento, se extrajeron 15 regiones con una alta concentración de células glandulares y se ejecutó el algoritmo para estudiar sus resultados. También se ha probado sobre regiones de imágenes malignas de grado 3.

4.1. Experimentos sobre la red propia

Todos y cada uno de los experimentos descritos a continuación están realizados sobre la arquitectura diseñada por nosotros mismos, explicada en la sección 2.1.1.

4.1.1. Elección de hiperparámetros

Learning Rate inicial

En todos los experimentos descritos en este capítulo, se aplicará como optimizador de la función de error el método Adam (ampliamente recomendado por la literatura [5]). Dicho método ajusta dinámicamente el Learning Rate, aunque puede resultar interesante encontrar el valor inicial óptimo. El procedimiento seguido para tal fin ha sido el siguiente:

- Se emplearon 7 de las 11 imágenes disponibles para este experimento (concretamente las 7 primeras del Cuadro 3.1, un total de 10.032 parches). De ellos, se muestran aleatoriamente el 25 % para ser utilizados como única partición de validación. Esto nos deja un total de 2.508 parches para validar y 7.524 para entrenar.
- Con dichos datos, el optimizador Adam y distintos valores iniciales para el Learning Rate, se entrenaron varios modelos durante 5 épocas. Se escogerá aquel que presente el mejor F1 Score.

Los valores estudiados fueron desde 0.0001 hasta 100 multiplicando por 10 en cada paso (conformando un espacio lineal). El Cuadro 4.1 muestra los resultados obtenidos por cada uno de ellos.

A partir de 0.01, los pasos dados por el optimizador son demasiado amplios, saltándose el mínimo cercano (además, con la precisión escogida, no se aprecia ninguna modificación ni en el Loss ni en el Accuracy). Por otro lado, el claro ganador es 1e-3, siendo este el valor de learning rate inicial que se utilizará en el resto de experimentos.

Learning Rate	F1 Score	Loss	Accuracy
1e-4	0.8515	0.3447	0.8698
1e-3	0.8706	0.2951	0.8826
1e-2	0.6248	6.0484	0.6247
0.1	0.6248	6.0484	0.6247
10	0.6241	6.0484	0.6247
100	0.5414	6.0484	0.6247

Cuadro 4.1: Resultados de la elección del LR inicial.

Penalización L2

Se trata de un tipo de regularización que, como su propio nombre indica, impone una penalización (L1, L2...) a la función de coste. Con estas penalizaciones obligamos a que se seleccione el modelo más simple de todos los existentes.

$$\text{Loss}(w) = \text{LossFunction}(X_i, y_i, w) + \lambda \text{Penalty}(w) \quad (4.1)$$

donde:

λ = hiperparámetro que determina la influencia de la regularización en la función de coste.

w = conjunto de pesos actuales de la red.

X_i = muestra iésima de los datos de entrenamiento.

y_i = etiqueta de la muestra iésima de los datos de entrenamiento.

Concretamente, añadiremos penalización L2 ($\|w\|_2^2$) con $\lambda = 0.01$ (tal y como recomienda la literatura [5]) a la capa totalmente conectada y al clasificador softmax del final.

Se realizaron dos experimentos: uno con este tipo de regularización y otro sin ella. Gracias a esto, podremos discutir mejor la influencia de la penalización en el modelo. Ambos se entrenaron durante 25 épocas, aplicando Adam con el Learning Rate inicial elegido previamente y exactamente con los mismos datos de entrenamiento y validación que en el experimento anterior.

En la Figura 4.1 (a), el error en validación comienza a crecer rápidamente mientras que en train continúa disminuyendo. El claro hueco que se observa entre las dos líneas es síntoma inequívoco de que se está produciendo sobreajuste. Este ocurre cuando el modelo aprende demasiado bien las muestras de aprendizaje (error en train bajo), pero no es capaz de generalizar ante ejemplos nunca antes vistos (error en validación alto). Sin embargo, la regu-

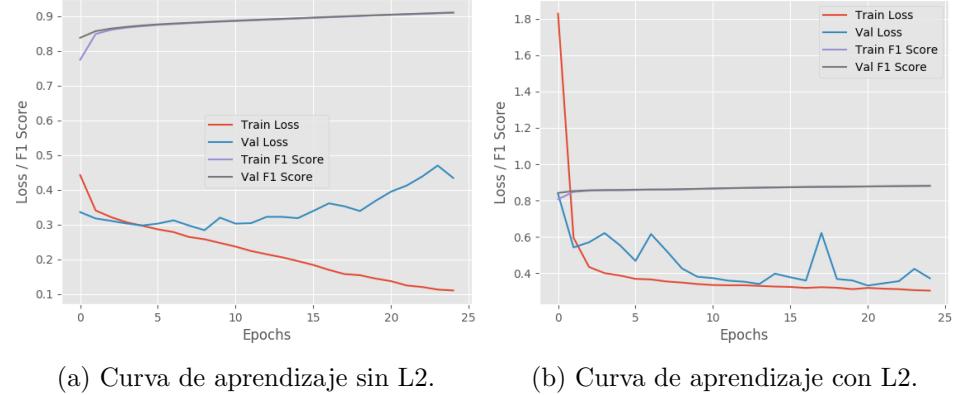


Figura 4.1: Resultados de los experimentos para estimar la penalización L2.

	Train	Validación
Loss	0.1100	0.4337
F1 Score	0.9091	0.9098
Accuracy	0.9573	0.8848

Cuadro 4.2: Resultados sin penalización L2.

	Train	Validación
Loss	0.3045	0.3727
F1 Score	0.8808	0.8812
Accuracy	0.9049	0.8829

Cuadro 4.3: Resultados con penalización L2.

larización (Figura 4.1 (b)) elimina dicho hueco, solucionando así el problema del sobreajuste. Podemos concluir que dicha regularización es relevante para los resultados, por lo que se incorporará en el resto de experimentos.

4.1.2. Experimento principal: estimación del error

El objetivo de este experimento es dar una estimación estadística robusta y fiable sobre el error del modelo a través de un conjunto de métricas, de tal forma que podamos discutir apropiadamente el desempeño de la arquitectura y compararla con el resto. Para ello, se realizará validación cruzada con las siguientes consideraciones:

- Se utilizarán las 11 imágenes disponibles y un total de 5 particiones de validación. Dichas particiones se realizarán por imagen, por ejemplo, 9 imágenes para entrenar y 2 para validar.

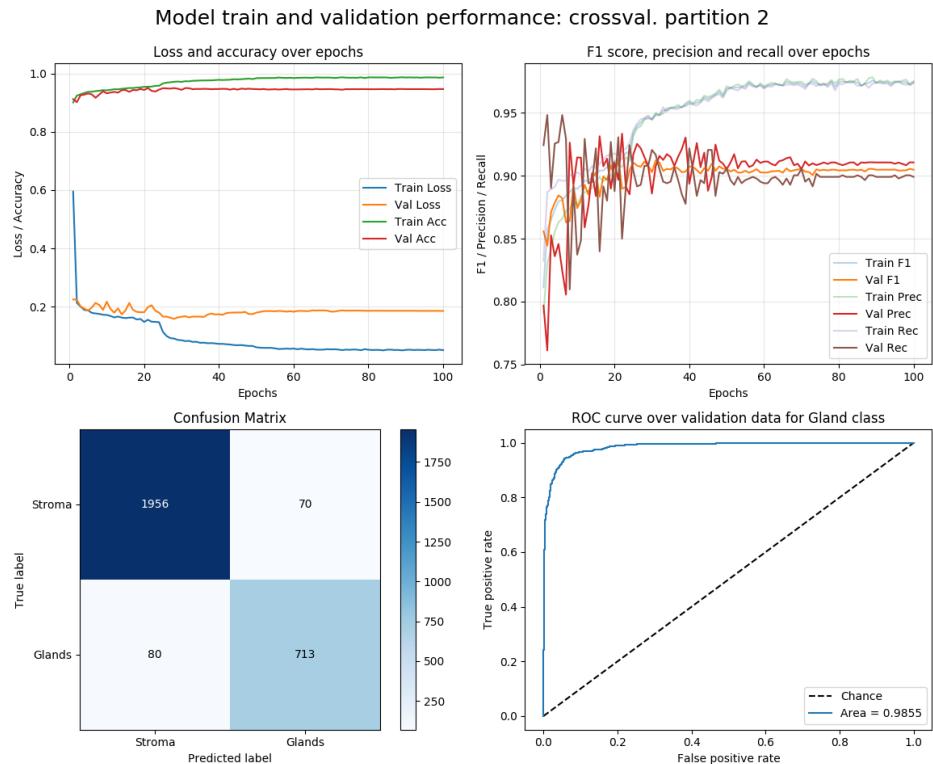
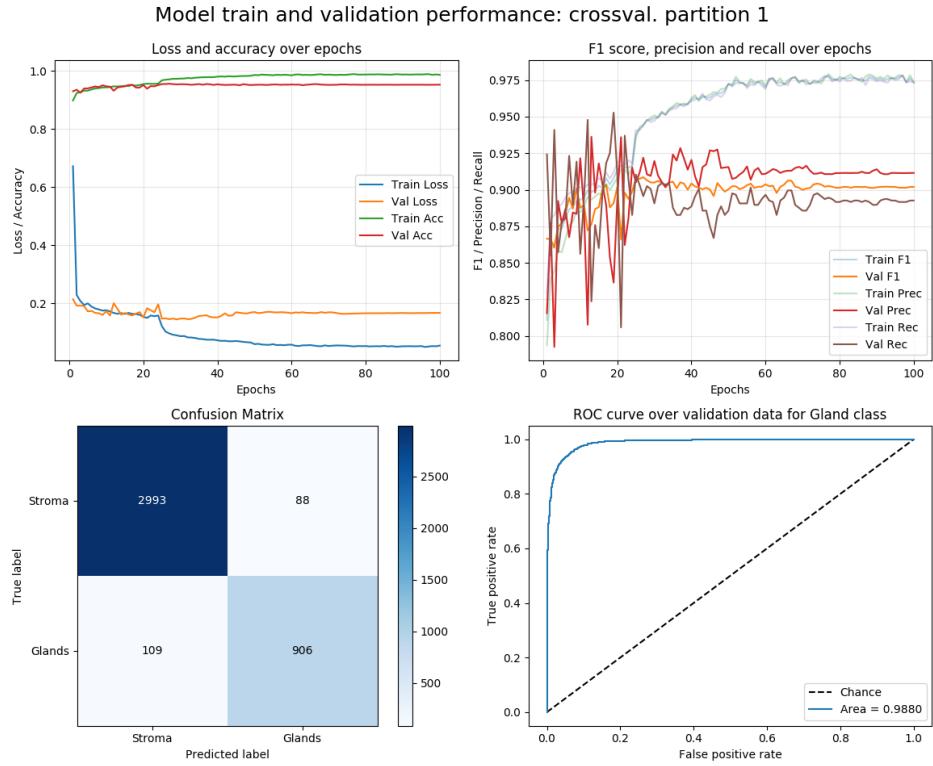
- Se utilizarán los hiperparámetros previamente fijados.
- Se empleará la doble escala descrita en la Sección 2.2.3.
- Se incorporará Learning Rate Decay: en las últimas épocas el error se estabiliza y prácticamente se mantiene invariante. Se plantea que cada 25 épocas el Learning Rate actual se divide entre 10. Esto es equivalente a aplicar Adam hasta la época 25 y después continuar empleando SGD con Learning Rates más pequeños.
- En cada partición de la validación cruzada, el modelo se entrenará durante 100 épocas.
- Se convertirá el problema en binario, prescindiendo de la clase “estroma contaminado”. El número de ejemplos de esta clase es extremadamente reducido comparado con el resto y dado que al fin y al cabo no son células glandulares, podemos considerarlos como ejemplos negativos.
- Aplicar data augmentation a los datos de train para suplir la escasez de muestras. Concretamente, cada parche se añade nuevamente al Dataset rotado 90, 180 o 270 grados de forma aleatoria. De esta forma, duplicamos los datos de entrenamiento.
- Por cada partición se calculará Precision, Recall, F1 Score, Accuracy, Cross-entropy Loss, matriz de confusión y curva ROC.
- Finalmente, se calculará la curva ROC media de todas las particiones y la matriz de confusión final.

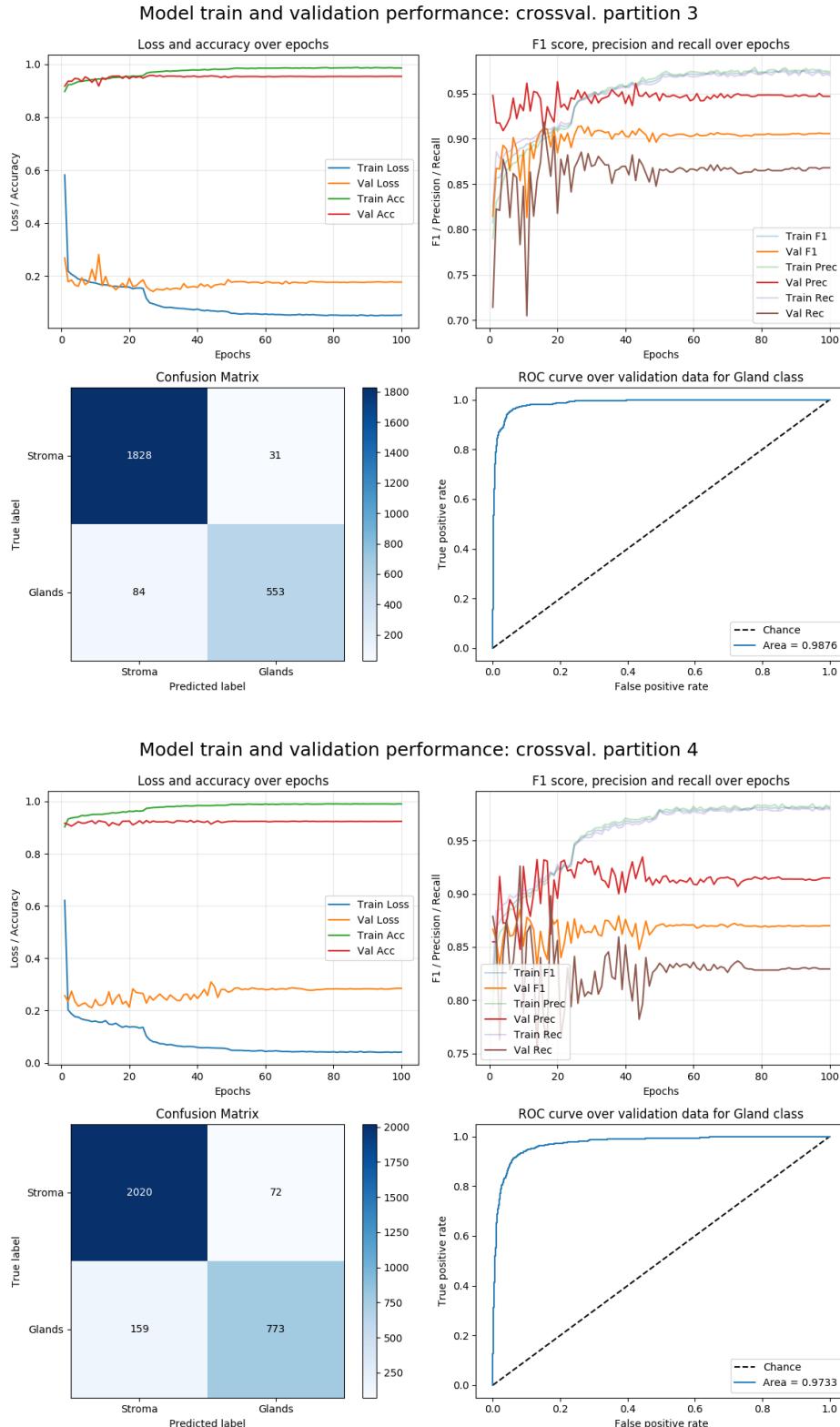
Los pasos seguidos durante la implementación del proceso de validación cruzada se describen con mayor detalle en el Apéndice A. Además, este experimento se ha repetido para dos valores distintos de la probabilidad de Dropout. Con esto se pretende estudiar qué nivel de regularización es suficiente para la red: una regularización baja ($p_d = 0.25$) o media ($p_d = 0.5$).

Dropout de 0.25

	Train	Validación
Loss	0.0300 ± 0.0031	0.1975 ± 0.0443
Accuracy	0.9973 ± 0.0008	0.9460 ± 0.0115
Precision	0.9968 ± 0.0012	0.9147 ± 0.0183
Recall	0.9928 ± 0.0018	0.8764 ± 0.0258
F1 Score	0.9948 ± 0.0013	0.8948 ± 0.0134

Cuadro 4.4: Validación cruzada I: dropout = 0.25 (media \pm std)





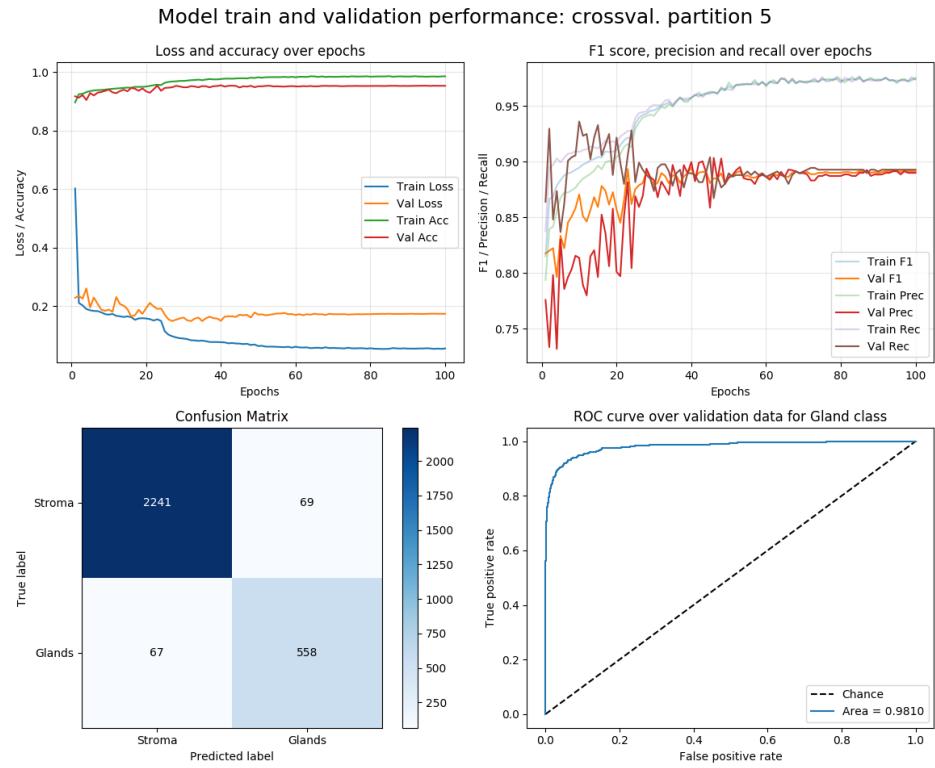


Figura 4.2: Validación cruzada I: dropout = 0.25 (curvas de aprendizaje por partición)

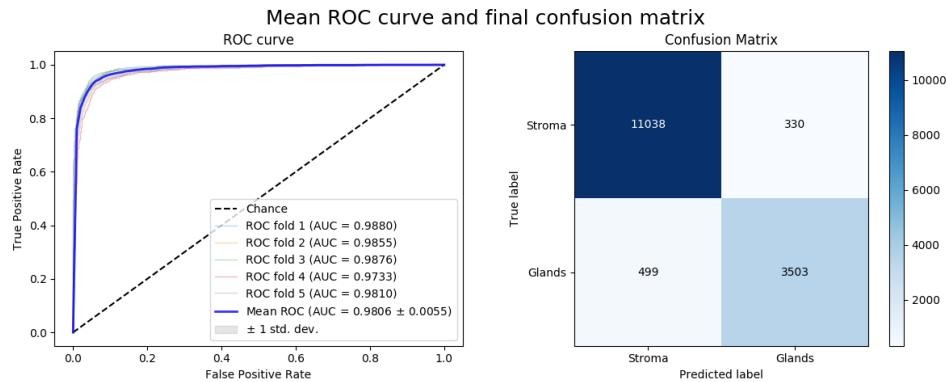
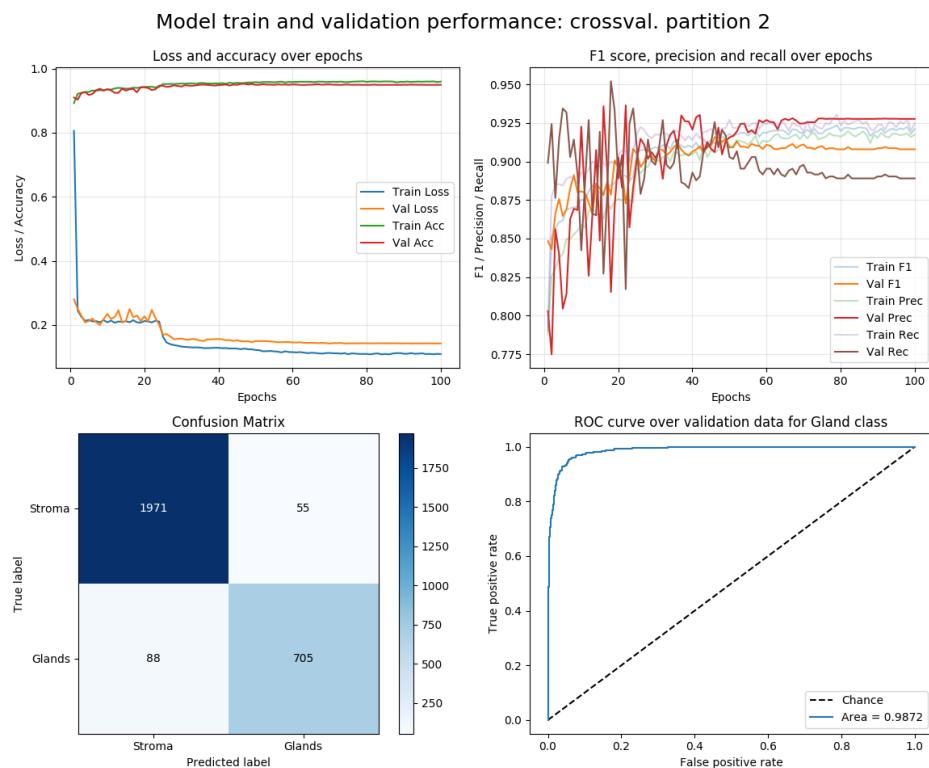
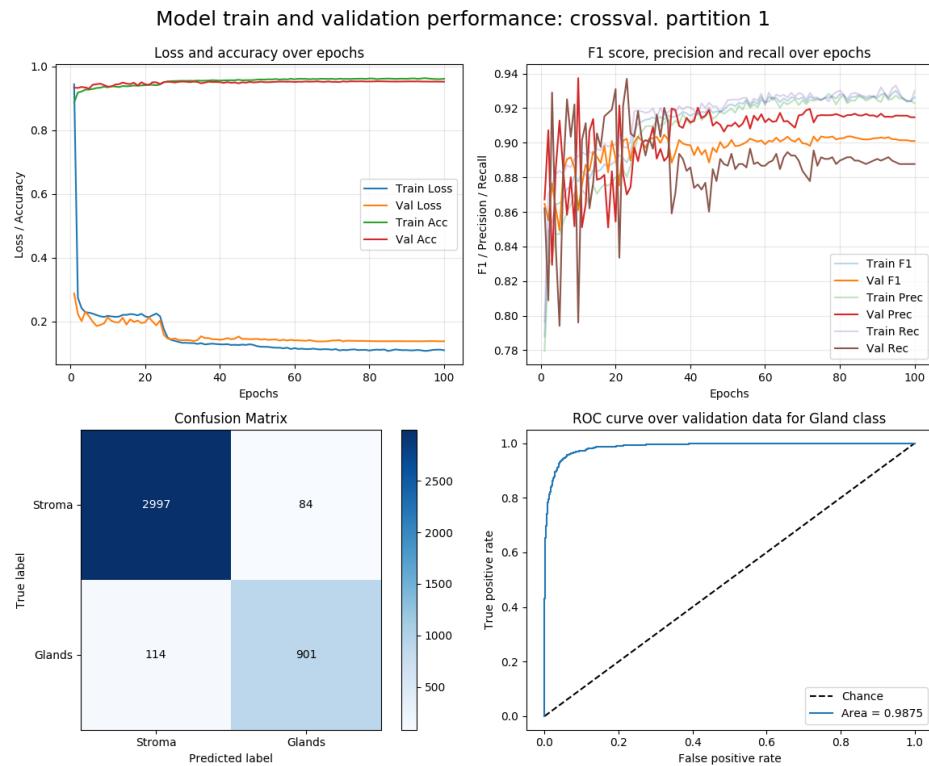
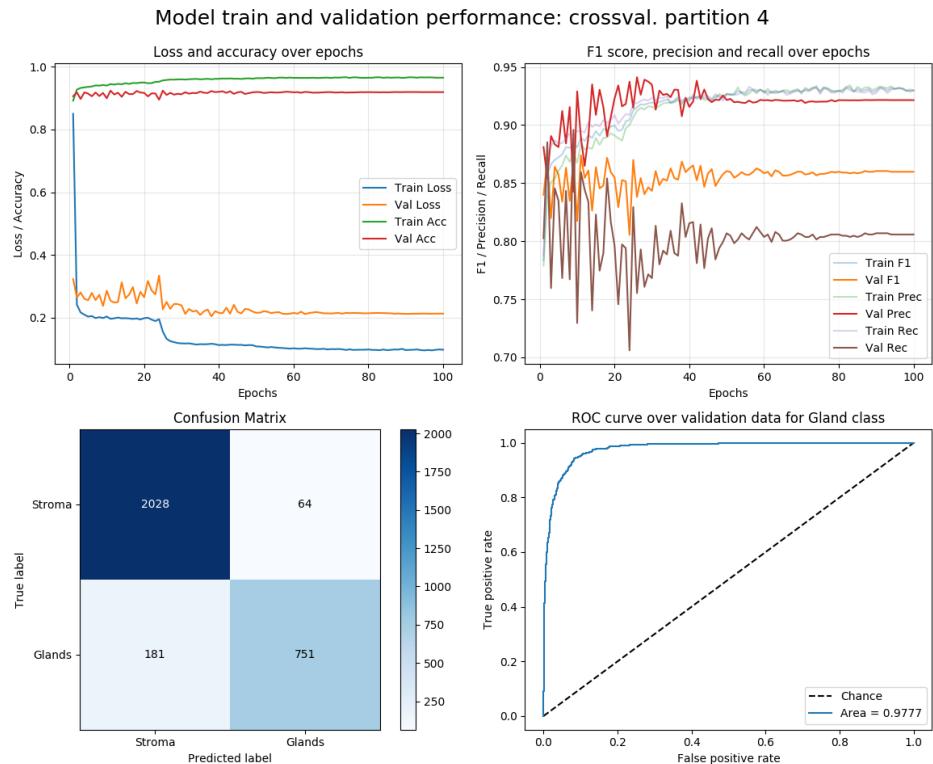
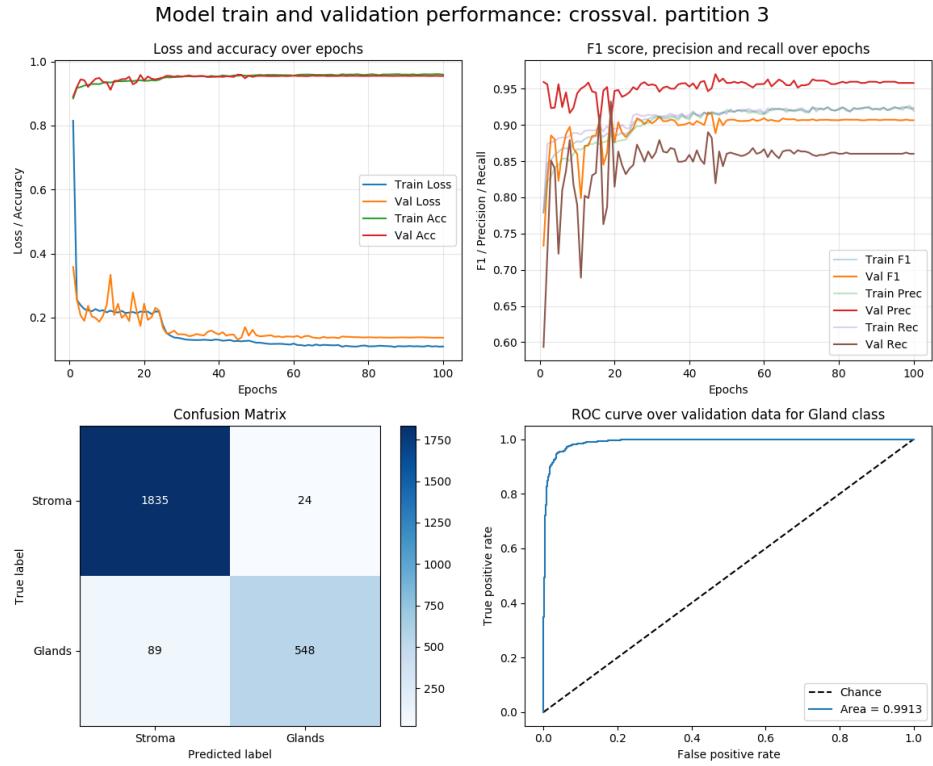


Figura 4.3: Validación cruzada I: dropout = 0.25 (Curva ROC media + Matriz de confusión final)

Dropout de 0.5





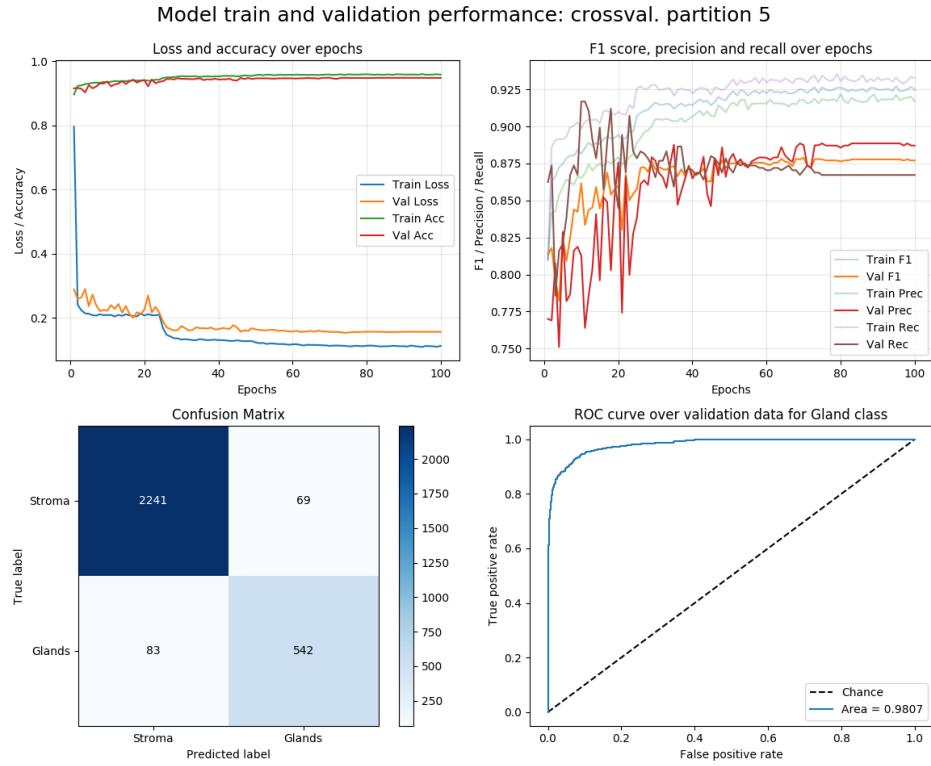


Figura 4.4: Validación cruzada II: dropout = 0.5 (curvas de aprendizaje por partición)

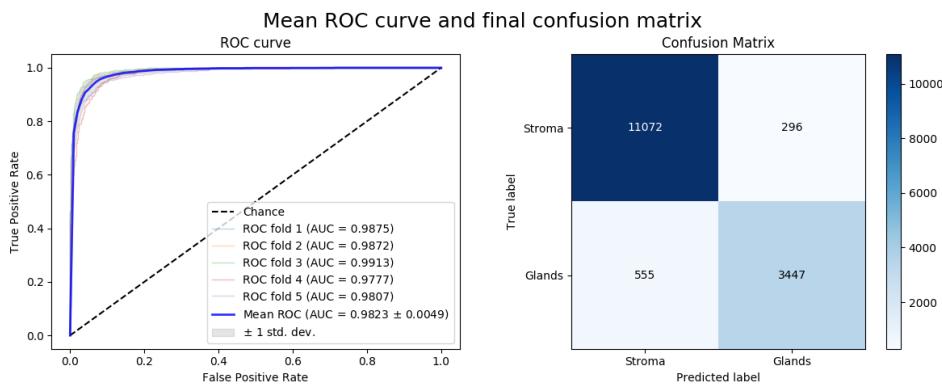


Figura 4.5: Validación cruzada II: dropout = 0.5 (Curva ROC media + Matriz de confusión final)

	Train	Validación
Loss	0.0964 ± 0.0077	0.1574 ± 0.0288
Accuracy	0.9661 ± 0.0037	0.9446 ± 0.0130
Precision	0.9548 ± 0.0022	0.9218 ± 0.0228
Recall	0.9134 ± 0.0105	0.8620 ± 0.0303
F1 Score	0.9336 ± 0.0063	0.8904 ± 0.0190

Cuadro 4.5: Validación cruzada II: dropout = 0.5 (media \pm std)

En ambos experimentos, llaman la atención las grandes oscilaciones que se observan en las curvas de aprendizaje durante las primeras etapas del entrenamiento (sobre todo en el F1 Score, Precision y Recall). Esto evidencia que el haber elegido un número de épocas alto ha sido una decisión acertada, pues el clasificador no comienza a estabilizarse, de media, hasta la época 20, tardando 50 épocas más en asentarse completamente.

Con respecto al valor de probabilidad para el dropout, si observamos el Cuadro 4.4, podemos notar que con 0.25 el modelo claramente sobreajusta. Todas las métricas en train alcanzan el 0.99, mientras que por el contrario en validación, son menores. Por otro lado, si aumentamos la probabilidad hasta 0.5 (Cuadro 4.5) el modelo reduce levemente su desempeño en train, pero esto no se traduce en una mejor generalización (pues las métricas en validación de los dos experimentos se diferencian en unas centésimas). Esto evidencia que la probabilidad de dropout no resulta relevante a la hora de determinar como de bien generaliza el modelo, por lo que deben de existir otras causas (probablemente relacionadas con los datos) que le impiden mejorar (véase la sección 4.3). No obstante, tomaremos como valor provisional para el dropout 0.25, pues aunque sobreajuste ligeramente, es el que presenta el mejor F1 Score en validación.

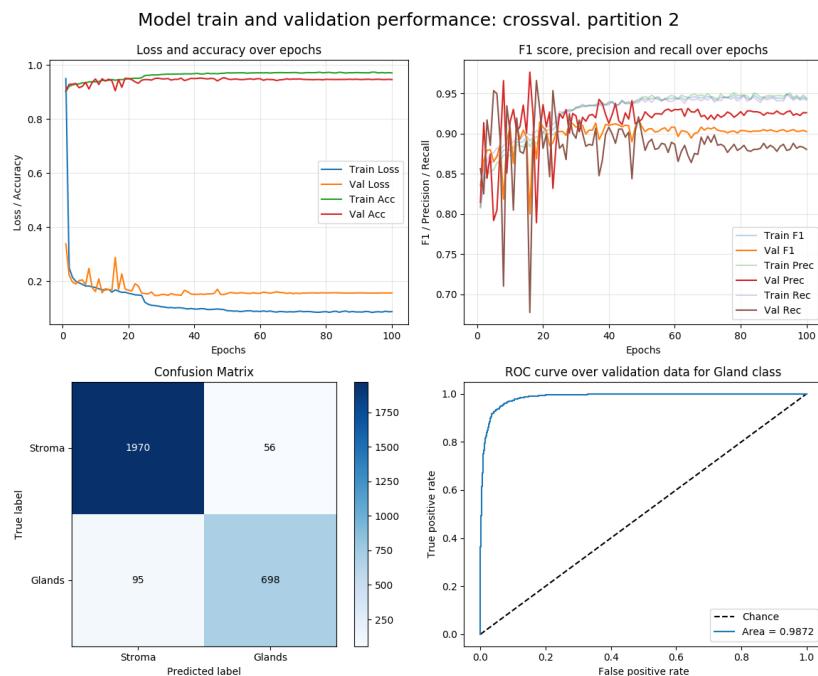
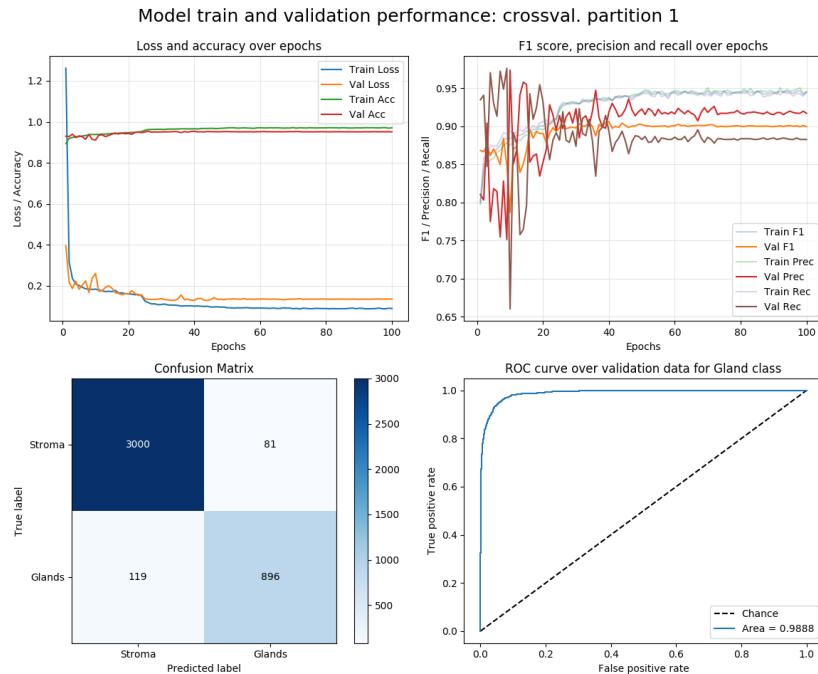
4.1.3. Batch Normalization

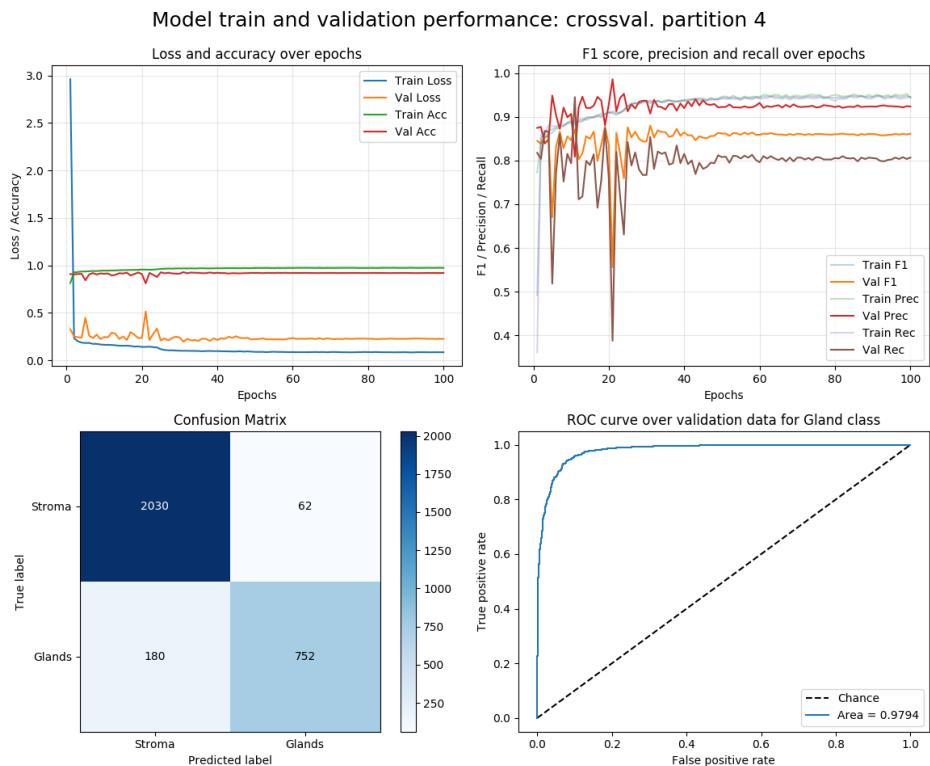
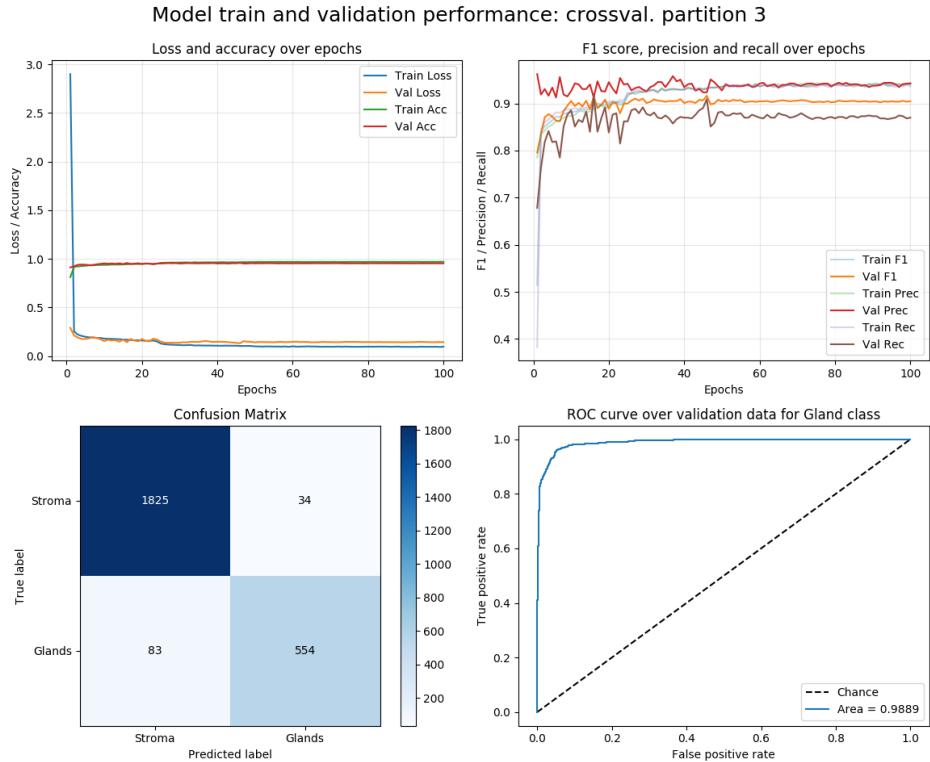
Como su propio nombre indica, la normalización por lotes (Batch Normalization) es una capa que coge el batch con el que se está entrenando y lo normaliza, restándole la media y dividiendo entre la desviación estándar. Teóricamente, la normalización por lotes ayuda en el aprendizaje, reduciendo el tiempo que se necesita para el mismo y aumentando tanto el desempeño como la estabilidad de la red neuronal. No obstante, existe bastante controversia respecto a los resultados prácticos, pues hay casos en los que efectivamente los mejora y casos en los que no. Nosotros ya tenemos resultados de como funciona la red sin Batch Normalization. Ahora probaremos a añadírsela y estudiaremos como afecta al desempeño final de la red.

Concretamente, añadiremos una capa de normalización del batch entre

la salida de las convoluciones y las no-linealidades (Leaky ReLU) tal y como se recomienda en el paper de los propios autores de Batch Normalization [9].

Dicho esto, repetimos nuevamente el experimento de validación cruzada:





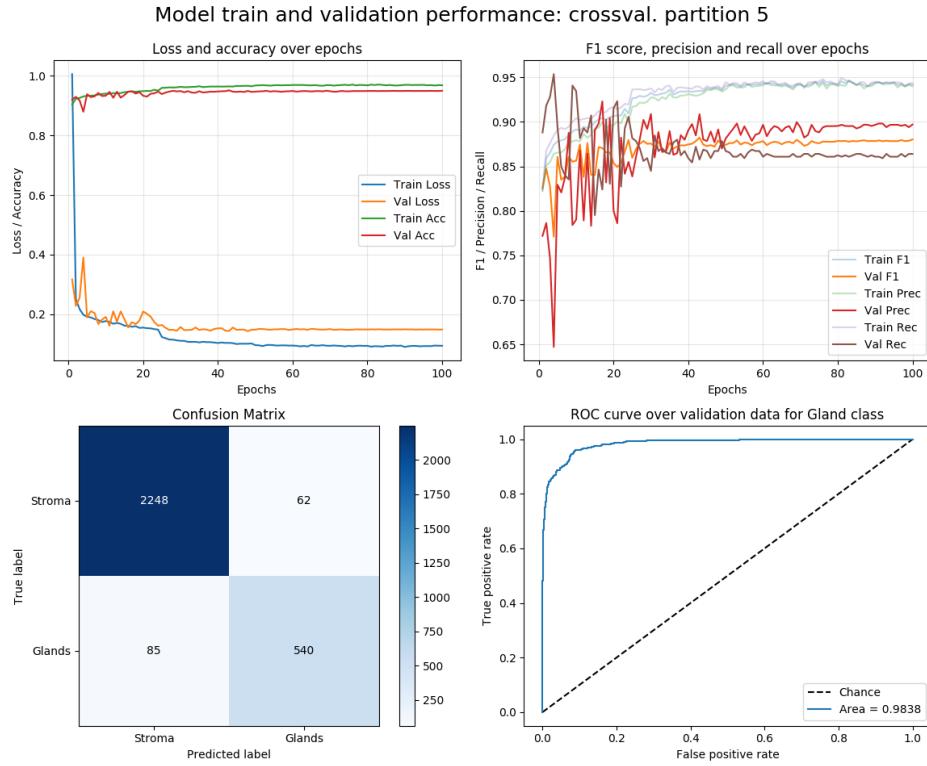


Figura 4.6: Validación cruzada III: Batch Norm. (curvas de aprendizaje por partición)

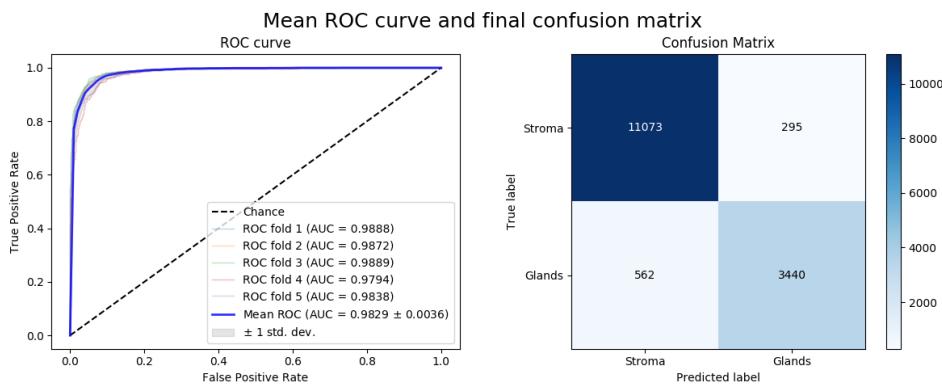


Figura 4.7: Validación cruzada III: Batch Norm. (Curva ROC media + Matriz de confusión final)

	Train	Validación
Loss	0.0661 ± 0.0040	0.1615 ± 0.0322
Accuracy	0.9817 ± 0.0021	0.9441 ± 0.0123
Precision	0.9746 ± 0.0067	0.9212 ± 0.0146
Recall	0.9547 ± 0.0054	0.8607 ± 0.0278
F1 Score	0.9645 ± 0.0033	0.8896 ± 0.0165

Cuadro 4.6: Validación cruzada III: Batch Norm. (media \pm std)

4.2. Experimentos de Transfer Learning

En esta sección se describen los distintos experimentos de Transfer Learning ya explicados en la sección 2.1.2, junto a sus resultados. Todos ellos se llevaron a cabo bajo las siguientes condiciones:

- Para agilizar el proceso, no se realizará validación cruzada. En su lugar, se utilizarán 2 imágenes aleatorias para validar (concretamente para estos experimentos, la 2 y 5 del Cuadro 3.1) y las 9 restantes para entrenar. Esto nos deja un total de 12.435 parches para entrenar y 2.935 para validar.
- Se aplicará data augmentation a los datos de train (el mismo que en experimentos anteriores).
- Se incorporará el mismo Learning Rate Decay que en los experimentos de validación cruzada.
- Se utilizará la versión binaria del problema.
- No se aplicará la doble escala.
- En los experimentos en los que se mantiene la extracción de características del modelo original, las arquitecturas se entrenarán durante 50 épocas.
- En el caso del ajuste fino de la red VGG16, se descongelarán únicamente las tres últimas capas de convolución (pues no se disponen de datos suficientes como para descongelar más) y se entrenarán junto al nuevo clasificador final durante 25 épocas.
- Se calcularán las metricas de Cross-entropy loss, Accuracy, Precision, Recall, F1 Score y la matriz de confusión para el conjunto de validación.

VGG16: extracción de características

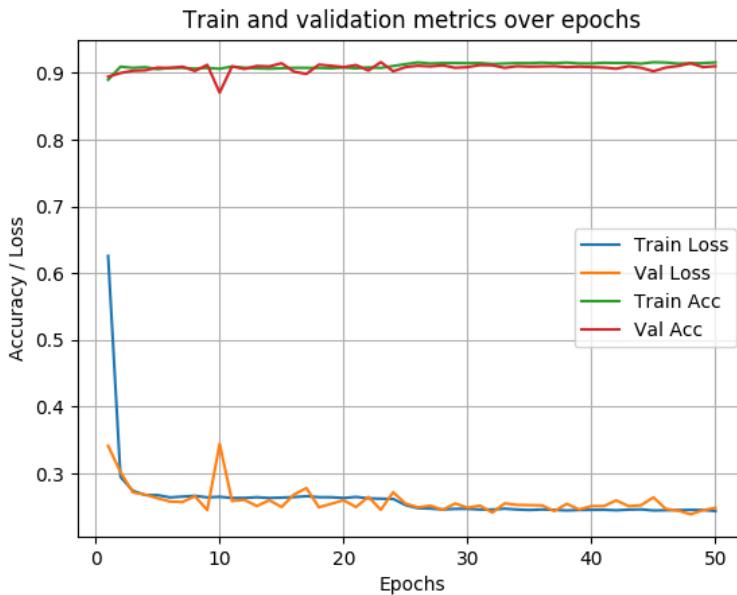


Figura 4.8: Transfer Learning I: VGG16 extracción de características (cruvas de aprendizaje).

Validation Loss	0.2486
Validation Accuracy	0.9101
Validation Precision	0.7568
Validation Recall	0.8512
Validation F1 Score	0.8012

Cuadro 4.7: Transfer Learning I: VGG16 extracción de características (métricas).

		Predicted Label	
		Stroma	Gland
True Label	Stroma	2139	171
	Gland	93	532

Cuadro 4.8: Transfer Learning I: VGG16 extracción de características (matriz de confusión).

VGG16: Fine Tunning

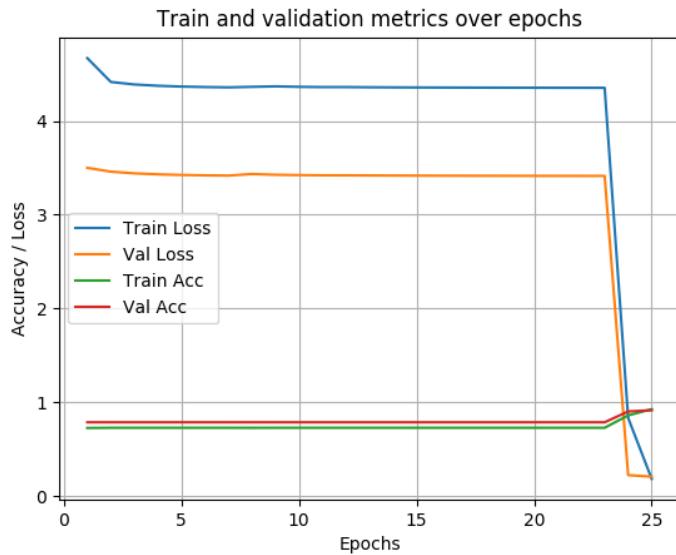


Figura 4.9: Transfer Learning II: VGG16 Fine Tuning (cruvas de aprendizaje).

Validation Loss	0.2066
Validation Accuracy	0.9155
Validation Precision	0.7697
Validation Recall	0.8608
Validation F1 Score	0.8127

Cuadro 4.9: Transfer Learning II: VGG16 Fine Tuning (métricas).

		Predicted Label	
		Stroma	Gland
True Label	Stroma	2149	161
	Gland	87	538

Cuadro 4.10: Transfer Learning II: VGG16 Fine Tuning (matriz de confusión).

En la curva de aprendizaje de este experimento (Figura 4.9) llama la atención que tanto el Accuracy como la Entropía Cruzada se mantienen constantes hasta la época 23-24, donde el error baja drásticamente y el Accuracy mejora. Justo en ese momento los pesos se orientan correctamente hacia el reconocimiento de glándulas.

InceptionV3: extracción de características

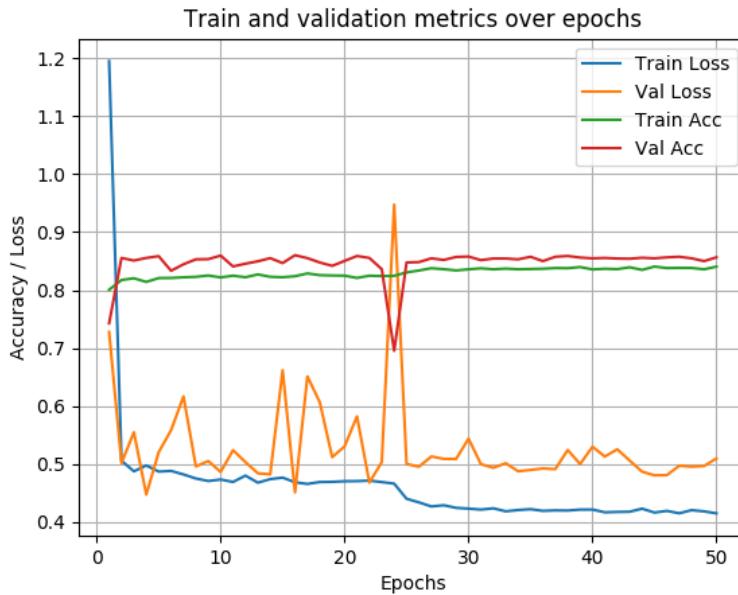


Figura 4.10: Transfer Learning III: InceptionV3 extracción de características (cruvas de aprendizaje).

Validation Loss	0.5094
Validation Accuracy	0.8566
Validation Precision	0.6828
Validation Recall	0.6096
Validation F1 Score	0.6441

Cuadro 4.11: Transfer Learning III: InceptionV3 extracción de características (métricas).

		Predicted Label	
		Stroma	Gland
True Label	Stroma	2133	177
	Gland	244	381

Cuadro 4.12: Transfer Learning III: InceptionV3 extracción de características (matriz de confusión).

4.3. Discusión de los experimentos

Disponemos de cuatro posibles candidatos a ser utilizados como clasificador del algoritmo final:

- (a) Nuestra propia arquitectura ($p_d = 0.25$).
- (b) Nuestra propia arquitectura ($p_d = 0.25$) con Batch Normalization.
- (c) La red VGG16 (con y sin Fine Tuning).
- (d) La red InceptionV3 (solo extracción de características).

A continuación se discutirá el desempeño de cada uno de ellos a partir de los experimentos y con el objetivo de escoger uno sobre el resto.

Transfer Learning: modelos (c) y (d) frente a (a) y (b)

Con respecto a los experimentos de Transfer Learning, en general, los resultados no han sido demasiado prometedores. Por un lado, la red Inception (Cuadro 4.11) se queda muy por detrás al resto para este problema, con un F1 Score de 0.6 y en general unas métricas bajas. Por otro lado, los resultados de la red VGG con únicamente extracción de características (Cuadro 4.7), son relativamente mejores pero no llegan a ser satisfactorios. El recall muestra un valor muy similar al de los experimentos (a) y (b), 0.86, aunque precision sufre una caída relevante (de 0.91/0.92 en (a) y (b) a 0.75). Esto provoca que el F1 Score final sea de 0.8 en validación. Una ventaja destacable de este entrenamiento es que no presenta ningún tipo de sobreajuste, pues en la Figura 4.8 puede observarse que las curvas de aprendizaje y validación están prácticamente solapadas.

Por otro lado, el realizar un ajuste fino de las 3 últimas capas de convolución de la red VGG (Cuadro 4.9) provoca que todas las métricas mejoren ligeramente (concretamente, se incrementan en unas centésimas), pues los pesos de las mismas se acercan un poco más a nuestro problema, aunque no constituye un avance remarcable. En definitiva, aún eligiendo el mejor modelo de Transfer Learning (en este caso, VGG16 con ajuste fino), el F1 Score alcanzado es inferior a cualquiera de los obtenidos en los experimentos (a) y (b), 0.81 frente a 0.89. Podemos concluir que las características aprendidas por estos dos modelos en el dataset de ImageNet no son demasiado extensibles a nuestro problema, algo perfectamente esperable, pues no se asemeja en nada un problema de reconocer más de mil categorías de objetos cotidianos, a un problema de medicina. Esto es especialmente palpable en la red Inception, ya que las características aprendidas por la red VGG si son más generales. Los modelos (c) y (d) no se emplearán en el algoritmo final.

Batch Normalization: modelo (a) frente a (b)

Con respecto al uso de Batch Normalization en nuestra arquitectura, sus resultados no distan mucho de los que no la emplean. Las diferencias más remarcables son las siguientes:

- En la validación cruzada del modelo con Batch Normalization (Figura 4.6), puede observarse que el hueco entre las curvas de train y validation es menor que en el caso sin Batch Normalization (Figura 4.2). Esto indica que la regularización introducida por la normalización del batch ayuda a paliar levemente el sobreajuste del modelo, haciendo que las métricas en train se aproximen más a los resultados en validación.
- A grandes rasgos, las métricas en validación del modelo con Batch Normalization (Cuadro 4.6) son muy parecidas a las que no la emplean (Cuadro 4.4) salvo ligeras variaciones. La entropía cruzada y precision mejoran al aplicar normalización por lotes. El F1 Score y el recall empeoran en una centésima y el accuracy permanece invariante.
- Comparando las matrices de confusión finales (Figuras 4.3 y 4.7) podemos observar que el modelo con Batch Normalization se equivoca en más ejemplos de la clase glándula que su contrapartida: 562 errores frente a los 499 del modelo que no la incorpora, aunque no es una diferencia abrumadora. Por otro lado, la AUC media del modelo con normalización por lotes es ligeramente superior a la otra (0.9829 frente a 0.9806).

Batch Normalization parece mejorar algunos aspectos del modelo y empeorar otros. Por un lado, la versión que lo emplea, presenta un menor sobreajuste, lo que se traduce en un error de validación más bajo, una mejor precision (respuesta ante falsos positivos) y un AUC mayor. Sin embargo, afecta negativamente a la habilidad del clasificador de encontrar todos los parches pertenecientes a glándulas (recall) y al F1 Score, nuestra métrica guía. Puesto que tal y como se menciona en los requisitos funcionales (Sección 5.1) lo que más interesa es no descartar ninguna glándula (sensibilidad ante falsos negativos, marcada especialmente por el recall) y porque las ventajas que proporciona la normalización por lotes al entrenamiento no son demasiado remarcables, prescindiremos de su uso. La arquitectura de diseño propio sin batch normalization y con un dropout de 0.25 será la que se emplee en el algoritmo final.

Discusión del modelo final (a)

A continuación, se llevará a cabo la interpretación con más detalle de sus métricas (Figuras 4.2, 4.3 y Cuadro 4.4).

Comenzaremos por la curva ROC media. Esta representa el ratio de verdaderos positivos en el eje Y, y el ratio de falsos positivos en el eje X. Esto significa que la esquina superior izquierda de la gráfica es el punto “ideal”: un ratio de falsos positivos de cero y un ratio de verdaderos positivos de uno. Por este motivo, cuanto mayor sea el área bajo dicha curva (AUC), mejor desempeño tendrá el clasificador. Si observamos la Figura 4.3, podemos ver que todas las particiones de la validación cruzada presentan un AUC nunca inferior a 0.97, siendo la media 0.98. Además, prácticamente no existe variación entre unos folds y otros, lo cual es indicio de que los resultados son bastante sólidos. Un área bajo la curva ROC de 0.98 es un comportamiento excelente.

Debido al desbalanceo, no le prestaremos especial atención al Accuracy, ya que, aunque en validación acierta en el 94 % de las veces, puede no ser un resultado del todo fiable. En su lugar, las que sí tienen relevancia, son las métricas de precision, recall y sobre todo, F1 Score. Teniendo en cuenta que estas tres métricas toman su mejor valor en 1 y el peor en 0, podemos concluir lo siguiente:

- Precision: intuitivamente, es la habilidad del clasificador de no etiquetar como positiva una muestra que es negativa. Traducido a nuestro problema, sería la capacidad del clasificador de no etiquetar como glándula algo que en realidad es estroma (falsos positivos). Su valor de 0.91 en validación denota una respuesta bastante buena ante los falsos positivos.
- Recall: intuitivamente, es la habilidad del clasificador de encontrar todas las muestras positivas, es decir, todos los parches pertenecientes a células glandulares. Esto es lo que a nosotros más nos interesa, no perdernos ni una sola glándula, por lo que un resultado de 0.87 en validación, teniendo en cuenta la dificultad del problema causada por la gran variabilidad morfológica de las glándulas, constituye un resultado satisfactorio.
- F1 Score: se trata de una media ponderada entre las dos anteriores. Esta métrica ha sido la guía de todos estos experimentos. Finalmente, hemos logrado alcanzar un valor de 0.89 en validación, lo cual resume el buen desempeño general de nuestra arquitectura.

Además, si miramos el Cuadro 4.4, podemos ver que la desviación estándar de todas las métricas es minúscula, por lo que podemos estar tranquilos de que los resultados son bastante representativos.

Por último, hemos visto en apartados anteriores que por mucho que se aumente la probabilidad de dropout del modelo, esto no se traduce en una

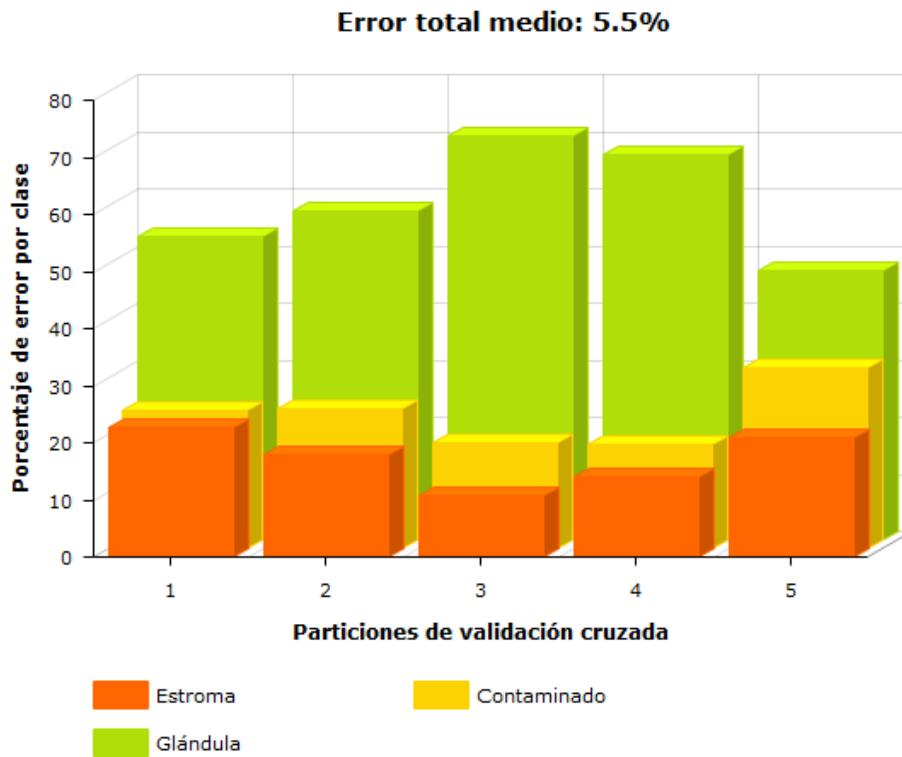


Figura 4.11: Porcentajes de error por clase en cada partición de la validación cruzada.

mejor generalización, es decir, los resultados en validación no avanzan de lo ya descrito en esta sección. Esto indica que probablemente, no sea problema del modelo, sino de los datos. Para encontrar la causa, visualizaremos los parches concretos en los que se está equivocando el clasificador en cada una de las particiones de validación cruzada, en búsqueda de algún patrón destacable, casos específicos con poca presencia que cuando no son vistos en training afectan negativamente a la validación. Al mismo tiempo, calcularemos un sumario que nos muestre el porcentaje de error cometido para cada una de las clases respecto del error total (Ecuación 4.2), manteniendo las tres categorías originales: glándula, estroma y estroma contaminado. Esto puede proporcionarnos pistas sobre los parches más costosos de identificar por el modelo.

$$\text{PorcentajeErrorClase}(A) = \frac{\text{NúmeroErrores}(A)}{\text{TotalErrores}} \cdot 100 \quad (4.2)$$

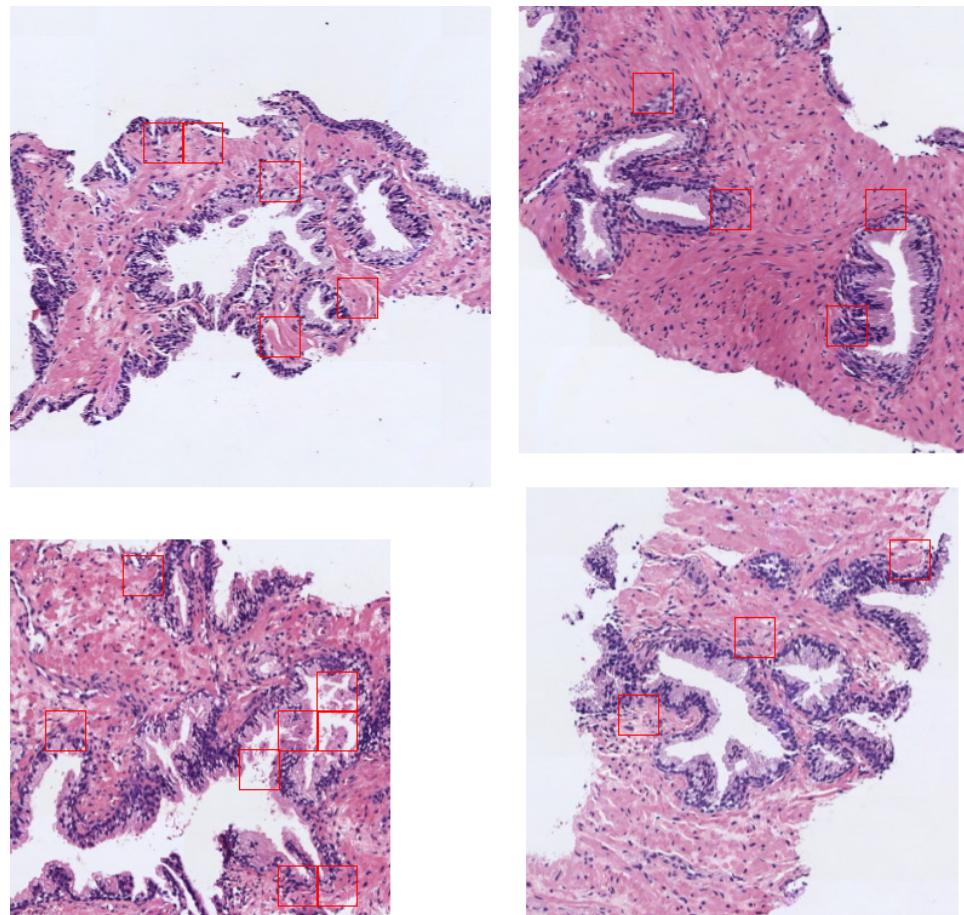


Figura 4.12: Algunos ejemplos sobre la localización de los falsos negativos en validación.

Como se puede observar en la Figura 4.11, paradójicamente, la clase en la que se comete un mayor número de errores es en la clase glándula, aquella que más nos interesa aprender. No obstante, tras inspeccionar cuidadosamente los falsos negativos dados en cada una de las imágenes de validación de las particiones, podemos reconocer que los parches erróneos se presentan aislados, es decir, que uno de ellos sea etiquetado como estroma no va a influir en que esa glándula no sea detectada. Para que una célula glandular fuese completamente ignorada, todos los parches asociados a ella deberían ser ignorados, cosa que no sucede en ningún momento. En la Figura 4.12 se muestran algunos fragmentos de estas imágenes de validación.

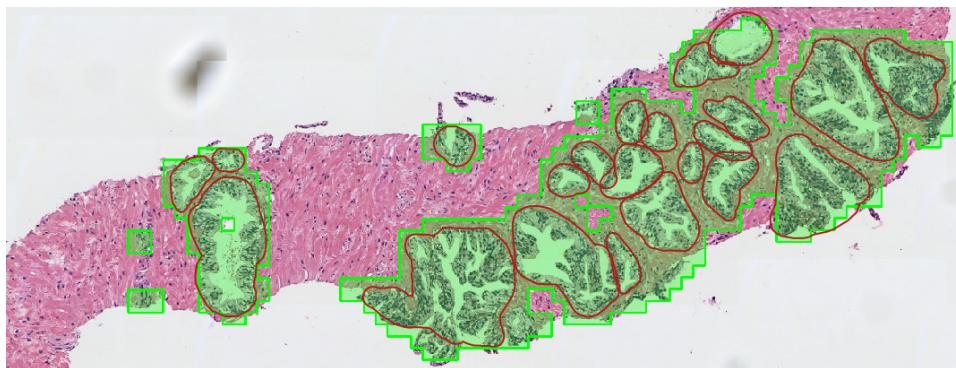
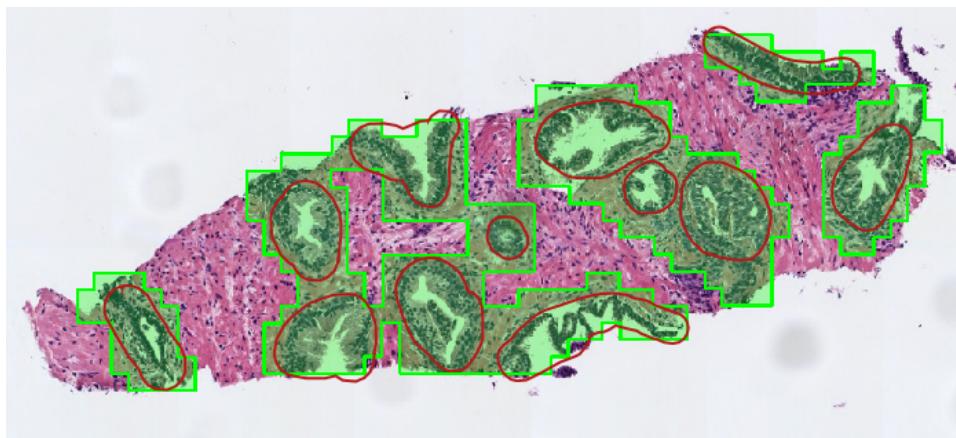
Por otro lado, no parece que exista un patrón distinguible en los errores cometidos (ni siquiera en las particiones 3 y 4, donde peor se comporta el clasificador). Simplemente y como hemos mencionado en el párrafo anterior, algunos de los parches de cada una de las glándulas son detectados como

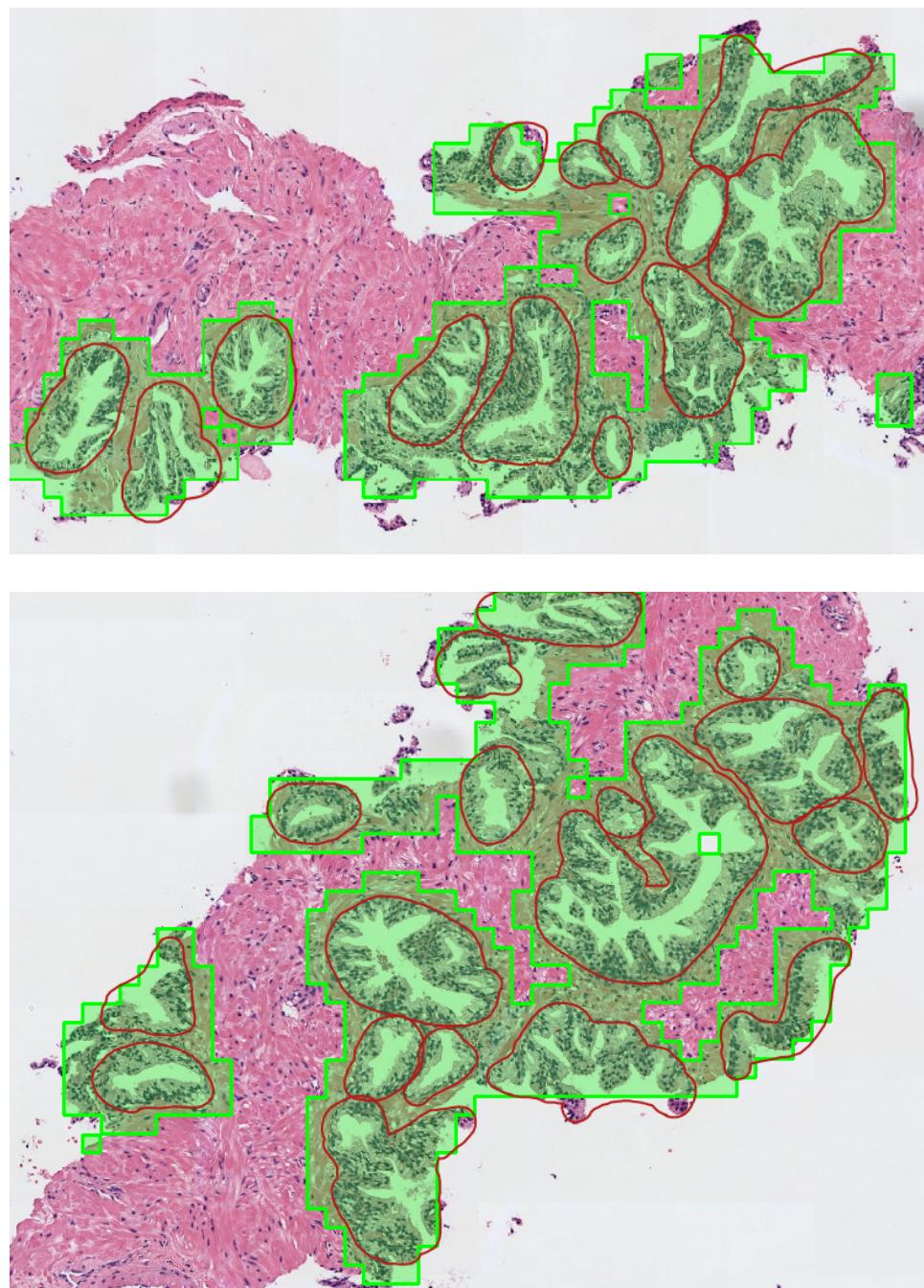
estroma. Esto es lo que está causando que el F1 Score no mejore de 0.89, pero reiterando lo dicho, son errores a nivel de parche y no afectan a la detección de la glándula en su conjunto. La escasez de datos y la dificultad asociada a la variabilidad en la morfología de las glándulas impiden al clasificador aprender más. Posiblemente, etiquetando muchas más imágenes y más variadas, este problema se subsanaría.

4.4. Test de detección

Elegido el modelo final, es hora de comprobar como funciona el proceso de detección descrito en la sección 2.4.

Para este experimento, se trajeron un total de 15 regiones de 4 imágenes benignas diferentes (y por supuesto, distintas a las utilizadas en el entrenamiento) que contienen altas concentraciones de células glandulares. Hemos considerado que una glándula ha sido detectada con éxito cuando el contorno de la misma cubre aproximadamente el 70 % de su superficie. A continuación se muestran algunos de los resultados: en verde, la detección del algoritmo, en rojo, las glándulas marcadas a mano.





De las 15 regiones extraídas, en 12 de ellas se detectan el 100 % de las glándulas presentes y en 1 se detecta el 90 % de las mismas. Las dos que restan son las únicas en las que el algoritmo no ha tenido tanto éxito: una de ellas no presenta ninguna glándula (elegida intencionadamente para estudiar su comportamiento en este tipo de imágenes. Figura 4.13) y aún así detecta

una estructura de vaso sanguíneo como célula glandular. En la otra, hay dos glándulas con una morfología un tanto diferente a lo habitual y no es capaz de detectarlas (Figura 4.14).

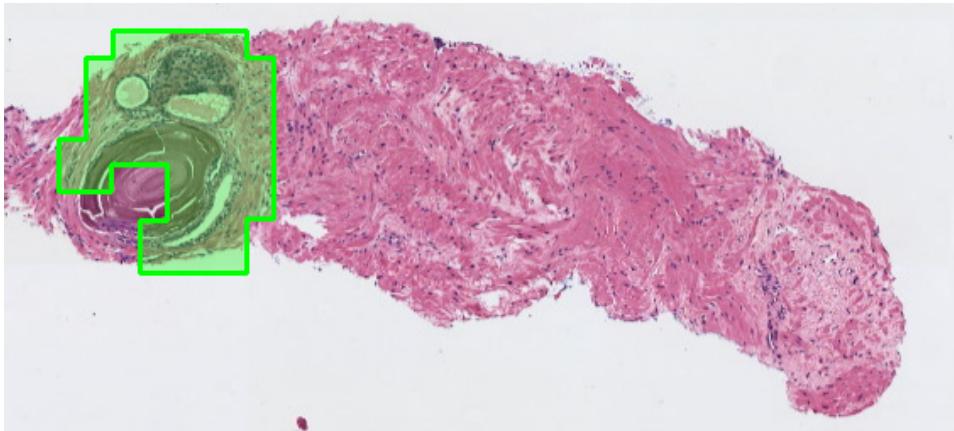


Figura 4.13: El algoritmo confunde un vaso sanguíneo con una glándula.

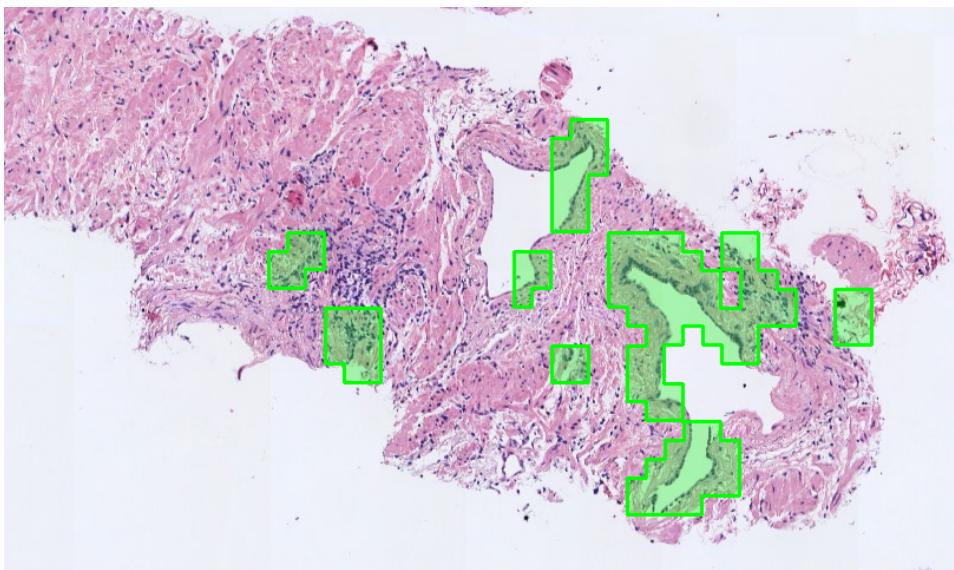


Figura 4.14: El algoritmo no llega a cubrir el 70 % de estas dos glándulas, las cuales presentan una morfología diferente: sin pliegues y con pocas células secretoras en la frontera.

A grandes rasgos, el algoritmo presenta un desempeño excelente en 13 de las 15 imágenes de test disponibles. Esto justifica lo descrito en la sección anterior: el que se falle en parches de glándula aislados no tiene un efecto remarcable en la detección de las mismas. No obstante, y siendo justos, el principal problema que tiene el procedimiento está en los falsos positivos:

detecta algunas zonas pequeñas en las que no se encuentran presentes células glandulares. Además, por el tamaño de parche, la detección no es muy ajustada a las glándulas. Aún así, marca correctamente las regiones de interés y no tiene problemas con falsos negativos (que es lo que más atención requería).

Finalmente, probaremos si el algoritmo es extensible al caso de muestras malignas de grado 3. Se han extraído 11 regiones de 4 imágenes diferentes. A continuación se muestran algunos de los resultados:

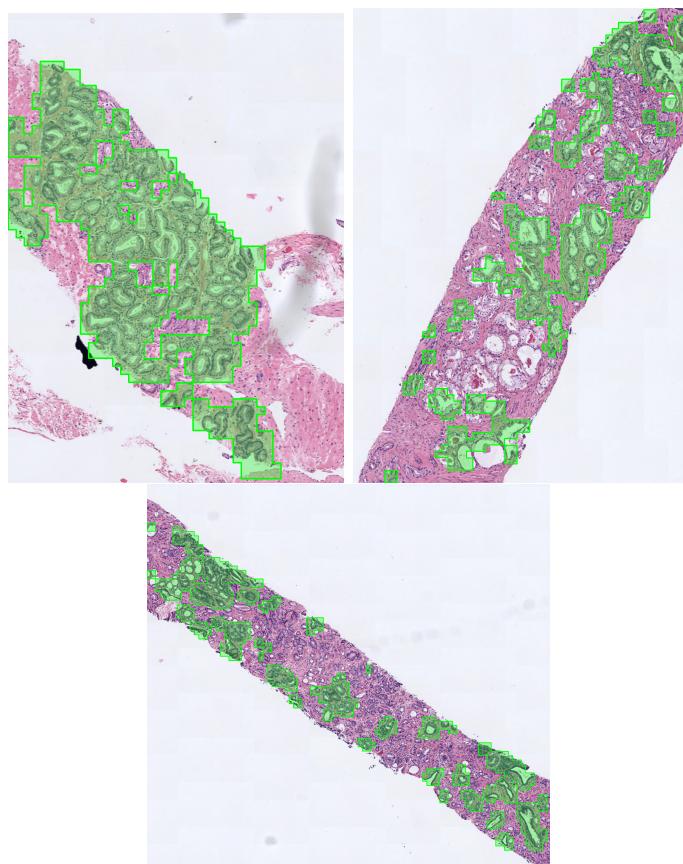


Figura 4.15: Detección del algoritmo en muestras malignas de grado 3.

En 8 de las 11 regiones, se detectan el 95 % de las glándulas, aunque también es cierto que su morfología no está demasiado degenerada (son similares a la imagen de arriba a la izquierda de la Figura 4.15). Sin embargo, cuando las estructuras sí muestran signos evidentes de deterioro (como en las dos imágenes restantes de la Figura 4.15) el algoritmo es incapaz de detectarlas. Habría que entrenar el modelo con imágenes de células glandulares de grado 3 para lograr mejorar su desempeño.

Capítulo 5

Desarrollo del Software

En este capítulo se describirán todos aquellos aspectos del proyecto relacionados con el diseño y desarrollo del software final.

5.1. Especificación de requisitos

Es innegable que el grueso del problema se centra en entrenar apropiadamente un modelo de aprendizaje. Para ello, habrá que fijar la arquitectura de red, ajustar correctamente los hiperparámetros de aprendizaje y preparar y preprocesar los datos de acuerdo a las restricciones del modelo.

Por otro lado, si hablamos desde la perspectiva de la funcionalidad del sistema, esta se centrará en recibir como entrada la digitalización de una preparación histológica, procesarla, y por último, escribir el resultado de la detección en una nueva imagen de salida.

Teniendo en cuenta lo mencionado anteriormente, la lista de requisitos es la siguiente:

5.1.1. Requisitos funcionales

- RF1: Dada una imagen de entrada, el sistema debe generar una nueva imagen de salida con la detección de las regiones que contienen células glandulares.
- RF2: Las regiones detectadas se marcarán en la imagen de salida mediante su contorno.
- RF3: El sistema debe tener una especial sensibilidad ante los falsos negativos. Dada la naturaleza del problema, debemos evitar en la medida

de lo posible que se obvien regiones que sí contienen células glandulares.

5.1.2. Requisitos no funcionales

- RNF1: Las imágenes de entrada y salida estarán en formato PNG.
- RNF2: Las imágenes deben ser procesadas por el modelo en un tiempo razonable, a ser posible, no superior a un minuto.

5.2. Planificación

En esta sección se describe con detalle la planificación adoptada para el proyecto, junto con la metodología de trabajo aplicada:

- El proyecto comenzó el día 15 de Febrero de 2019. Durante 10 días únicamente se realizó trabajo de investigación para profundizar en el conocimiento de redes neuronales, Deep Learning y su aplicación práctica en el campo de la Visión por Computador para el procesamiento de imágenes. También se estudiaron los distintos frameworks de aprendizaje profundo, junto a sus ventajas e inconvenientes y se asentaron los objetivos principales.
- El 25 de Febrero de 2019, se comenzó a trabajar en el entrenamiento y todo lo que ello conlleva: etiquetado de los datos, implementación de módulos para el preprocesamiento de las imágenes WSI, ajuste del modelo y obtención de resultados estadísticos. Se estableció como fecha límite el día 13 de Mayo de 2019. Durante este periodo (el cual ha sido el más costoso de todo el proceso de desarrollo) la forma de trabajar se ha basado en pequeñas iteraciones de una semana en la que se fijaron objetivos claros y abarcables.
- Durante las dos semanas siguientes, hasta el 26 de Mayo de 2019, se implementó el software que hace uso del modelo y se realizaron pruebas sobre su funcionamiento.
- El día 24 de Junio de 2019 se comenzó con la redacción de esta memoria y se estableció como fecha límite para terminarla el día 31 de Julio de 2019.

En total se llevaron a cabo 10 días de investigación, 77 días para el entrenamiento del modelo (aproximadamente 2 meses y medio), 14 días para el software final y 38 días para la memoria.

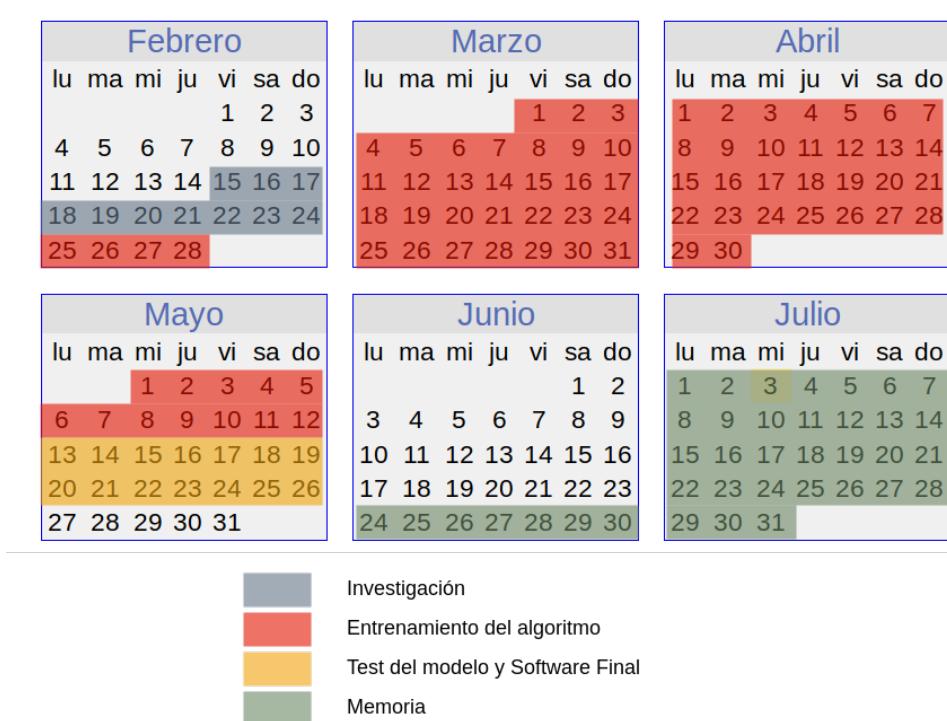


Figura 5.1: Calendario con la planificación.

5.3. Diseño del Software

Teniendo en cuenta los requisitos especificados en la sección 5.1, no se prevé que la magnitud del sistema en términos de módulos y clases sea muy elevada. Por tanto, no creemos necesario aplicar una metodología sofisticada de diseño del software. No obstante, esto no quiere decir que no se hayan tenido en cuenta una serie de consideraciones a la hora de escribir los diferentes scripts:

1. Se han desarrollado módulos que contienen todas aquellas funciones que serán reutilizadas en varios puntos del sistema. Concretamente, existen dos módulos fundamentales:
 - libs/DataUtils: contiene todas aquellas funciones necesarias para preprocessar las imágenes de alta resolución de cara al modelo. También incluye procedimientos para dibujar los contornos de

las detecciones del algoritmo. Así mismo, como procesar imágenes tan grandes puede ser extremadamente ineficiente, está muy optimizado.

El módulo se ha diseñado para proporcionar una capa de abstracción en el preprocesamiento y con la intención de que sea lo más general posible. Utilizando sus funciones y proporcionando diferentes parámetros, conseguimos una gran flexibilidad, la cual será clave para encontrar los valores apropiados de cara a la red.

- `models/cnn`: como su propio nombre indica, contiene la definición de las distintas arquitecturas de red utilizadas durante el proceso de entrenamiento.
2. Se han separado cada una de las fases del proceso de obtención del modelo (etiquetado, entrenamiento, validación y test) en scripts diferentes. De esta forma, cada una de ellas está perfectamente diferenciada, facilitando así tanto las modificaciones como la depuración.

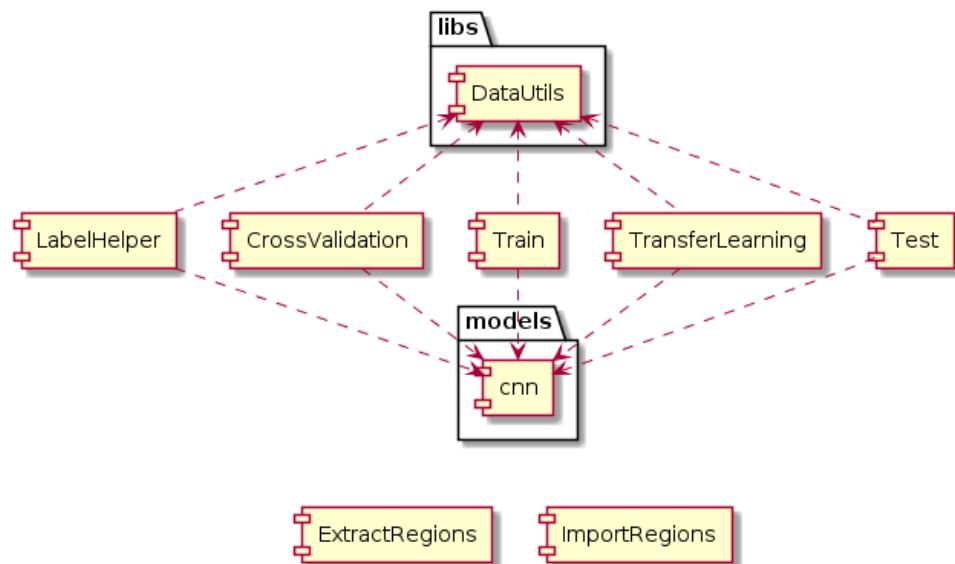


Figura 5.2: Estructura del sistema.

A continuación se describe de forma resumida la funcionalidad de cada componente del sistema:

- `libs/DataUtils`: módulo que contiene todas aquellas funciones utilizadas en el preprocesamiento de las imágenes: división en parches, data augmentation, creación de máscaras para la eliminación del fondo, etcétera.

- models/cnn: módulo que contiene la definición de las distintas arquitecturas de redes neuronales con las que se han realizado experimentos.
- LabelHelper: script auxiliar para el proceso de etiquetado de los datos.
- CrossValidation: script que realiza la validación cruzada de un determinado modelo. Calcula información estadística y gráficas.
- Train: script para entrenar un modelo utilizando todos los datos disponibles. Empleado en la obtención del modelo final.
- TransferLearning: script utilizado para entrenar y validar modelos construidos utilizando la técnica de Transfer Learning.
- Test: script equivalente al software final. Dada una imagen de entrada, la preprocesa y efectúa la detección.
- ImportRegions: script de Groovy utilizado desde QuPath. Permite añadir a una imagen los parches etiquetados por el clasificador como anotaciones, previamente escritos en un fichero de texto con extensión .pred.
- ExtractRegions: script de Groovy utilizado desde QuPath. Permite exportar los parches anotados en una imagen a un fichero de texto con extensión .qpdat, que podrá ser procesado desde Python.

5.3.1. Lenguaje y bibliotecas

Un aspecto clave del diseño es decidir que herramientas van a utilizarse de cara a la implementación. Como lenguaje de programación para el sistema, se ha elegido Python, el cual en los últimos años ha adquirido una gran relevancia en el campo de la programación científica y el análisis computacional de datos.

Con respecto a los frameworks de aprendizaje profundo, se tuvieron en cuenta dos grandes alternativas:

- Pytorch: es una API de bajo nivel centrada en trabajar directamente con expresiones de arrays. Ampliamente utilizada en el campo de la investigación académica y en el desarrollo de aplicaciones de deep learning que requieren de la optimización de expresiones personalizadas. Ofrece una mayor flexibilidad a la hora de desarrollar capas propias y está orientada a usuarios con cierto conocimiento matemático.
- Keras: es un API de alto nivel capaz de correr sobre Tensorflow, CNTK o Theano. Su principal ventaja es la facilidad de uso y la sencillez de su sintaxis. Permite desarrollar rápidamente y es ideal si se desea

experimentar con las capas estándares del deep learning. Abstira al científico de datos de las complejidades del aprendizaje profundo.

Dado que pretendemos entrenar modelos que utilizan las capas clásicas y no nos planteamos realizar ninguna modificación de bajo nivel, hemos optado por utilizar Keras (con Tensorflow como backend).

Por último, para ayudarnos en el cálculo de métricas y estadísticas, utilizaremos la API de machine learning Scikit-Learn y para las operaciones propias de la Visión por Computador (filtrado, procesamiento de imágenes, etcétera) utilizaremos OpenCV.

5.4. Pruebas

A continuación, detallaremos el proceso seguido para garantizar que tanto el sistema como los distintos módulos implementados son correctos, es decir, que todos funcionan como deberían (sin comportamientos anómalos o inapropiados) y que cumplen con los requisitos establecidos. Para todo esto, se han llevado a cabo las siguientes pruebas individuales:

1. Comprobar el funcionamiento de cada uno de los procedimientos del módulo DataUtils.
2. Verificar que el proceso de entrenamiento es apropiado y asegurarnos de que no introducimos ningún tipo de sesgo en los resultados estadísticos finales.

5.4.1. Pruebas del módulo DataUtils

Este módulo incorpora funciones para el procesamiento de los recursos que necesita el algoritmo (ya sean imágenes, ficheros de texto con el etiquetado, rutas del sistema, etcétera). Por tanto, las pruebas realizadas se basan en comparar manualmente los resultados obtenidos con los resultados esperados, incluso siendo necesario elaborar subrutinas específicas para facilitarlo en algunos casos. A continuación se describe a grandes rasgos el procedimiento seguido:

1. Seleccionar un conjunto reducido y controlado de entrada para las pruebas.
2. Llamar a la función correspondiente con las entradas previamente seleccionadas y unos parámetros fijos.

3. Inspeccionar el resultado obtenido, comparándolo con el resultado esperado.

Para ilustrar lo expuesto, se muestran un par de ejemplos:

- Una de las funciones nos permite dividir una imagen en un conjunto de parches cuadrados de un tamaño y separación concretos. Para verificarla, elegimos una imagen de entrada, la fraccionamos utilizando dicha función y visualizamos cada uno de los parches obtenidos comparándolos con la imagen original para ver si si la división es apropiada. Además, si para la visualización de los parches diseñamos un proceso que nos muestre el número de píxeles de los parches resultado y sus coordenadas en la imagen original, podemos asegurarnos de que tienen el tamaño y la separación apropiados.
- Otra función nos permite escribir un conjunto de parches a partir de sus coordenadas y etiquetas en un fichero de texto. Para verificar esta función simplemente la utilizamos para generar un archivo de un subconjunto de los parches y a continuación, leyendo el fichero, comprobamos que el número de parches es el mismo, que las etiquetas y las coordenadas escritas coinciden y que la separación y tamaños de los parches son correctos.

En definitiva son pruebas bastante tediosas (puesto que no pueden automatizarse) pero necesarias. Cada vez que se implementa una función nueva, esta es probada exhaustivamente.

5.4.2. Pruebas del proceso de entrenamiento y validación

Resulta de vital importancia que este paso esté correctamente implementado, pues el modelo es el corazón del proyecto.

Las funciones que llevan a cabo el entrenamiento de la red están proporcionadas por las APIs empleadas, por lo que en principio, deben ser correctas. Lo que a nosotros nos incumbe es asegurarnos de que las llamadas se realizan según el flujo de trabajo de Keras, su semántica y que reciben los parámetros pertinentes (por ejemplo, que recibe los parches correctamente preprocesados o de cara al cálculo de estadísticas, asegurarnos de que ninguno de los datos utilizados para validar haya sido visto por el modelo durante proceso de entrenamiento).

Por tanto, con revisar el código repasando las llamadas a los frameworks, asegurándonos de que son correctas respecto a la documentación y comprobando qué parámetros están recibiendo debería ser más que suficiente.

5.5. Análisis DAFO

Al fin y al cabo esto es un producto de ingeniería y como tal, para conocer mejor cuáles son sus puntos fuertes y débiles, resulta interesante realizar su análisis DAFO.

Debilidades

- Sólo útil para el caso benigno, siendo el maligno el verdaderamente interesante.
- Las imágenes requieren de mucho preprocesado.

Amenazas

- Los especialistas detractores del uso de redes neuronales en medicina.
- La gran cantidad de trabajos existentes al respecto.

Fortalezas

- Resultados buenos y robustos.
- Eficiente y fácil de utilizar.
- Puede ser utilizado en cualquier equipo.

Oportunidades

- Los hospitales utilizan cada vez más técnicas de patología digital como mecanismo de detección precoz del cáncer.

Capítulo 6

Conclusión

El principal objetivo de este proyecto es la elaboración de un algoritmo capaz de detectar, en biopsias de próstata digitalizadas, las regiones que resultan de interés a la hora de identificar si un paciente presenta o no cáncer de próstata (concretamente, las regiones con células glandulares). Podemos concluir que hemos alcanzado dicho objetivo con éxito. La red neuronal empleada presenta un F1 Score de 0.89 que, teniendo en cuenta la dificultad del problema asociada a la variabilidad en la morfología de las glándulas, es un resultado bastante bondadoso. Además, hemos determinado que, aunque el F1 Score no sea todo lo alto que podría desearse, esto no influye negativamente en la detección de glándulas. Se ha comprobado que el clasificador falla en parches aislados y que gracias a la combinación de probabilidades entre parches adyacentes y la umbralización óptima, la detección continúa siendo exitosa, obteniéndose más de un 90 % de glándulas reconocidas en el caso benigno.

También se ha testeado cuál es el desempeño del algoritmo en preparaciones histológicas con cáncer de tercer grado. Los resultados evidencian que el procedimiento sigue comportándose excelentemente en casos en los que las glándulas continúan presentando una morfología similar a las benignas, es decir, cuando estas aún no han llegado a degenerarse en exceso. Esto es un indicio de la buena capacidad de generalización que presenta. Hay que tener en cuenta que el modelo no ha visto glándulas de tercer grado durante el proceso de entrenamiento y aún así es capaz de identificar algunas similitudes con las benignas. Si por el contrario, el clasificador se entrenara con parches de glándulas pertenecientes a distintos grados de cáncer (ya sea segundo, tercero, cuarto o quinto), en principio debería ser capaz de reconocerlas sin ningún problema. De hecho, hemos identificado que el motivo por el cual el modelo no es capaz de aprender más es por la falta de datos. Con más imágenes etiquetadas las métricas deberían mejorar. Esto justifica lo previamente mencionado: su buena generalización y su extensibilidad a

otros casos.

Por otro lado, el haber utilizado con éxito una red neuronal convolucional para resolver este problema, no hace más que reafirmar lo potentes y útiles que resultan este tipo de modelos en el campo de la detección de objetos, quedando muy por encima de las técnicas clásicas de visión por computador. En lugar de invertir nuestro esfuerzo en diseñar un sofistificado procedimiento de extracción de características tal y como requieren los métodos tradicionales, hemos empleado un modelo capaz de aprender por sí mismo esas características relevantes de los datos, permitiéndonos así concentrarnos en intentar ajustarlo al máximo. Además, con respecto a los resultados obtenidos, estos seguramente son mucho mejores a los podríamos haber conseguido con una técnica clásica, pues se trata de un problema bastante complejo incluso para las redes neuronales. Con respecto a la dificultad que supone para el modelo el procesar imágenes de alta resolución, hay que destacar que, gracias al factor de submuestreo, la eliminación del fondo y la división en parches, el proceso es bastante eficiente y no consume demasiado tiempo de cómpupto.

Por último, con respecto a la utilidad del producto final, si el algoritmo y el modelo se integran en un software de patología digital (como por ejemplo QuPath, el que se ha utilizado en este proyecto), podría facilitarse enormemente la tarea de los anatomopatólogos, aumentando así la productividad. Si además se desarrollara un segundo modelo capaz de tomar las regiones detectadas por nuestro algoritmo y clasificarlas en los distintos grados de cáncer, podría automatizarse completamente el proceso de diagnóstico. En definitiva, el algoritmo desarrollado en este proyecto, constituye una primera aproximación necesaria que allana el camino para futuras implementaciones y sobre todo, para el futuro de la medicina.

Apéndice A

La validación cruzada en detalle

Dado que la validación cruzada es de vital importancia para obtener resultados estadísticos robustos sobre el error de nuestro clasificador, aquí se explica la misma con total transparencia:

1. Se realizan las particiones por imágenes: como hay un total de 11 y se han hecho 5 divisiones para validación cruzada, en la primera partición se entrena con 8 imágenes y se valida con 3. En el resto, se entrena con 9 y se valida con 2.
2. Una vez separadas las imágenes en train y validación, en cada partición se realiza lo siguiente:
 - a) Se construye el conjunto de train. Se extraen los parches de 32x32x6 (las dos resoluciones) de las imágenes y se aplica data augmentation.¹
 - b) Se construye el conjunto de validación. Se extraen los parches de 32x32x6 (las dos resoluciones) de las imágenes y no se aplica data augmentation.
 - c) Se pasa el problema a binario, sustituyendo las etiquetas 2 de ambos conjuntos por 0 (el estroma contaminado pasa a ser estroma).
 - d) Se calcula la media y desviación estándar del conjunto de train y con ella, se normalizan tanto el conjunto de train como el de validación.

¹Cada imagen se añade de nuevo al dataset rotada aleatoriamente 90, 180 o 270°.

- e) Se entrena un modelo recién inicializado con los criterios descritos a lo largo de la Subsección 4.1.2: 100 épocas y Learning Rate Decay.
 - f) Una vez finalizado el entrenamiento, se evalúa el modelo para los datos de train y validación y se almacenan los resultados. Concretamente, para cada partición se calcula:
 - Loss, Accuracy, F1 Score, Precision y Recall para train y validación.
 - Matriz de confusión y Curva ROC solo para validación.
 - g) Se pasa a la siguiente partición.
3. Finalizada la validación cruzada, se calcula:
- a) La curva ROC media siguiendo el siguiente ejemplo de scikit-learn [19].
 - b) La matriz de confusión final realizando la sumatoria de las matrices de confusión de cada partición.
 - c) La media y desviación estándar de cada métrica guardada en el paso f) de las particiones de la validación cruzada, tanto para train como para validación.

El código que realiza esta parte se puede consultar en el script CrossValidation.

Bibliografía

- [1] Asociación Española Contra el Cáncer. Todo sobre el cáncer, 2019. URL <https://www.aecc.es/es/todo-sobre-cancer>.
- [2] Asociación Española Contra el Cáncer. Cáncer de próstata: Toda la información, 2019. URL <https://www.aecc.es/es/todo-sobre-cancer/tipos-cancer/cancer-prostata>.
- [3] Bankhead, P. et al. QuPath: Open source software for digital pathology image analysis. *Scientific Reports*, 2017. doi: <https://doi.org/10.1038/s41598-017-17204-5>. URL <https://www.nature.com/articles/s41598-017-17204-5>.
- [4] P. L. Farahani N, Parwani A. Whole slide imaging in pathology: advantages, limitations, and emerging perspectives. *Dovepress*, 2015. doi: <https://doi.org/10.2147/PLMI.S59826>. URL <https://www.dovepress.com/whole-slide-imaging-in-pathology-advantages-limitations-and-emerging-p-peer-reviewed-article-PLMI>.
- [5] Fei-Fei Li, Justin Johnson, Serena Yeung (Stanford University). CS231n: Convolutional Neural Networks for Visual Recognition, 2017. URL <http://cs231n.stanford.edu/>.
- [6] A. H. Fischer, K. A. Jacobson, J. Rose, and R. Zeller. Hematoxylin and eosin staining of tissue and cell sections. *Cold Spring Harbor Protocols*, 2008(5):pdb.prot4986, 2008. doi: 10.1101/pdb.prot4986. URL <http://cshprotocols.cshlp.org/content/2008/5/pdb.prot4986.abstract>.
- [7] Gallego, Jaime; Pedraza, Anibal; Lopez, Samuel; Steiner, Georg; Gonzalez, Lucia; Laurinavicius, Arvydas; Bueno, Gloria. Glomerulus classification and detection based on convolutional neural networks. *Journal of Imaging*, 2018. doi: <https://doi.org/10.3390/jimaging4010020>. URL <https://www.mdpi.com/2313-433X/4/1/20>.
- [8] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.

- [9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL <http://arxiv.org/abs/1502.03167>.
- [10] Jason Brownlee. Evaluate the Performance Of Deep Learning Models in Keras, 2016. URL <https://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/>.
- [11] Jason Brownlee. A Gentle Introduction to Transfer Learning for Deep Learning, 2017. URL <https://machinelearningmastery.com/transfer-learning-for-deep-learning/>.
- [12] G. Litjens, C. I. Sánchez, N. Timofeeva, M. Hermsen, I. Nagtegaal, I. Kovacs, C. Hulsbergen van de Kaa, P. Bult, B. van Ginneken, and J. van der Laak. Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. *Scientific Reports*, 6:26286, 2016. URL <http://dx.doi.org/10.1038/srep26286>.
- [13] G. J. S. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, and C. I. Sánchez. A survey on deep learning in medical image analysis. *CoRR*, abs/1702.05747, 2017. URL <http://arxiv.org/abs/1702.05747>.
- [14] Medical Biostatistics. ROC Curve. URL <http://www.medicalbiostatistics.com/roccurve.pdf>.
- [15] MedlinePlus. Sistema de puntuación de Gleason, 2019. URL <https://medlineplus.gov/spanish/ency/patientinstructions/000920.htm>.
- [16] OpenCV. Contours : Getting Started. URL https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html.
- [17] Quirurgie: Grupo Quirúrgico. Anatomopatología, 2019. URL <https://quirurgie.com/anatomopatologia/>.
- [18] H. Ritchie and M. Roser. Causes of Death. Our World in Data, 2017. URL <https://ourworldindata.org/causes-of-death>.
- [19] Scikit-learn. Receiver Operating Characteristic (ROC) with cross validation. URL https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc_crossval.html.
- [20] D. Shen and H.-I. Wu, G. and Suk. Deep learning in medical image analysis. *Annual Review of Biomedical Engineering*, 19:221–248, 2017. URL <http://doi.org/10.1146/annurev-bioeng-071516-044442>.

- [21] Sociedad Española de Oncología Médica. Las cifras del cáncer en España, 2018. URL <https://seom.org/es/noticias/106525-las-cifras-del-cancer-en-espana-2018>.
- [22] VIP (Universidad de Granada), CVLAB (Universidad Politécnica de Valencia). Enhancement, classification and interpretation of histological images of cancer. Project SICAP. URL <http://decsai.ugr.es/pi/sicap/>.
- [23] S. Yoo, I. Gujrathi, M. A. Haider, and F. Khalvati. Prostate cancer detection using deep convolutional neural networks. *CoRR*, abs/1905.13145, 2019. URL <http://arxiv.org/abs/1905.13145>.

Glosario

algoritmo Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problema. 3

aprendizaje profundo conjunto de algoritmos de aprendizaje automático que intenta modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos expresados en forma matricial o tensorial. 62

biopsia Examen microscópico de un trozo de tejido o una parte de líquido orgánico que se extrae de un ser vivo. 2

clase En el campo de la Visión por Computador hace referencia a un conjunto de elementos que comparten una serie de características. 7

data augmentation Proceso consistente en aplicar transformaciones (rotaciones, escalados, recortes...) a los datos disponibles para aparentemente generar nuevas imágenes. 30

estroma Tejido conjuntivo que constituye la matriz o sustancia fundamental de un órgano y sostiene los elementos celulares que lo conforman. 20

framework estructura conceptual y tecnológica de asistencia definida con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. 62

glomérulo Cada uno de los diminutos ovillos de capilares situados en el riñón donde se filtra la sangre y se elabora la orina. 9

instancia En el campo de la Visión por Computador hace referencia a un objeto concreto de una determinada clase. 7

morfología Rama de la biología que estudia la forma o estructura de los seres vivos. 4

patología Parte de la medicina que estudia los trastornos anatómicos y fisiológicos de los tejidos y los órganos enfermos, así como los síntomas y signos a través de los cuales se manifiestan las enfermedades y las causas que las producen. 2

red neuronal Paradigma de aprendizaje y procesamiento automático inspirado en el funcionamiento del sistema nervioso humano. 8

software Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas. 2

