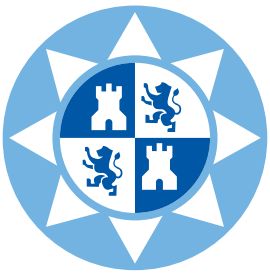


Universidad Politécnica de Cartagena

Trabajo Fin de Grado

Graduado en Ingeniería de Sistemas de Telecomunicación

Técnicas de segmentación semántica basadas en aprendizaje profundo aplicadas a retinografías

Carlos García Navarro

Director: Juan Morales Sánchez

Departamento de Tecnologías de la Información y las Comunicaciones

8 de diciembre de 2020

Agradecimientos

Gracias a Ángela, por ser siempre un apoyo incondicional y un ejemplo a seguir en todo lo que hace. Gracias por formar parte de mí.

Gracias a Juan Morales, mi tutor, por prestarme ayuda cuandoquiera que lo necesitara y por permitirme investigar en un tema que he disfrutado de inicio a fin.

Gracias a mis padres, mi hermano y el resto de mi familia, por apoyarme a lo largo de todo este proyecto. Gracias a mis abuelos, por estar siempre conmigo.

Gracias a mis compañeros Ángel, Paula, Alfredo, Joserra, Nabil y Juanjo, por regalarme algunos de los años más felices en la universidad y, sobre todo, por tener la capacidad de hacerme reír en cualquier momento.

Gracias a José Mari y Emilio, por demostrarme que las personas buenas siempre van a existir.

Resumen

Los avances de la última década han brindado al Deep Learning de una versatilidad que permite aplicarlo a problemas de cualquier índole. Esta característica ha propiciado la creación de diversos modelos de aprendizaje máquina por parte de la comunidad científica, dedicados a resolver un determinado problema. En el caso que compete, segmentación semántica, se ha entrenado una red neuronal profunda con el objetivo de identificar dos zonas en la retinografía, que permiten hacerse una idea del estado del glaucoma. Estas son el disco óptico y la excavación, con las cuales se obtiene un parámetro conocido como *Cup-to-Disk ratio*, muy útil para diagnosticar esta enfermedad.

En el presente trabajo se busca segmentar retinografías utilizando dos bases de datos de libre acceso: RIMONEr3 y RIGA. Gracias a su gran extensión, la red ha logrado aprender las características necesarias para segmentar adecuadamente las imágenes, que se ven reflejadas en una media de índice de Dice de 0.960 y una distancia Hausdorff de 1.649 % para el disco óptico. Por otro lado, para la excavación, se ha obtenido un índice de Dice medio de 0.863, y una distancia Hausdorff media de 2.213 % . Estos resultados demuestran que se trata de un modelo perfectamente válido para la segmentación de retinografías, capaz de aportar ayuda al sanitario en el diagnóstico de esta enfermedad.

Palabras clave: Deep Learning; segmentación; retinografía; excavación; disco óptico; glaucoma.

Abstract

The advances of algorithms in the last decade have given Deep Learning a flexibility that allows it to be applied to different problems. This feature has encouraged the creation of several machine learning models by the scientific community, aimed at solving a particular problem. In the case of semantic segmentation, a deep neural network was trained with the intention of identifying two areas of a retinography, which provide an overview of the state of glaucoma. These are the optic disk and the cup, which allow to obtain a parameter known as *Cup-to-Disk* ratio, which is very useful to diagnose this disease.

In the current project we are looking for segmenting retinographies using two free access databases: RIMONEr3 and RIGA. Due to its wide extension, the network has been able to learn the necessary features to adequately segment the images, which are represented by an average Dice index of 0.960 and a Hausdorff distance of 1.649 % for the optic disk. On the other hand, for the cup, an average Dice index of 0.863 has been obtained, and an average Hausdorff distance of 2.213 % . These results show a perfectly valid model for the segmentation of retinographies, which is also capable of helping the health professional in the diagnosis of this disease.

Keywords: Deep Learning; segmentation; retinography; cup; optic disk; glaucoma.

Índice general

Índice de figuras

Índice de tablas

Glosario

ACC	Accuracy
AdaGrad	Adaptative Gradient Algorithm
Adam	Adaptative Moment Stimation
ASPP	Atrous Spatial Piramid Pooling
API	Application Programming Interface
AUC	Area Under the Curve
CDR	Cup to Disk Ratio
CPU	Central Processing Unit
CRFs	Conditional Random Fields
DCNNs	Deep Convolutional Neural Networks
DL	Deep Learning
FOV	Field of View
GPU	Graphics Processing Unit
IA	Inteligencia Artificial
IOU	Intersection Over Union
ML	Machine Learning
MSE	Mean Square Error
OS	Output Stride
PIO	Presión Intraocular
ReLU	Rectified Linear Unit
RMS	Root Mean Square
ROC	Receiver Operating characteristic Curve
SGD	Stochastic Gradient Descent
TPU	Tensor Processing Unit

1 Introducción

En el presente Trabajo de Fin de Grado, se busca aportar una solución alternativa a la segmentación de retinografías con el propósito de diagnosticar el glaucoma. Para este pretexto, se utilizará un enfoque basado en una Deep Convolutional Neural Network (DCNN, Red Neuronal Convolucional Profunda), que será entrenada utilizando distintas bases de datos.

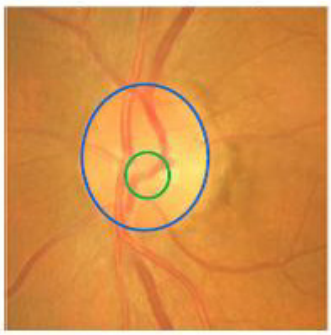
El glaucoma es la segunda causa de ceguera tras la catarata, y la primera causa mundial de ceguera irreversible (Lago y Bengoa, 2018). Es una enfermedad que daña el nervio óptico, debido a un aumento de la Presión Intraocular (PIO), consecuencia directa de una acumulación de humor acuoso en el interior del ojo, que no se puede drenar (Anónimo, 2020c).

Esta enfermedad representa un importante problema de salud pública, que se espera que este mismo 2020 alcance los 79.6 millones de enfermos, de los cuales el 13 % se quedarían ciegos (Anónimo, 2020b). El diagnóstico de esta enfermedad puede agruparse en tres categorías, en función del método utilizado (Anónimo, 2020a):

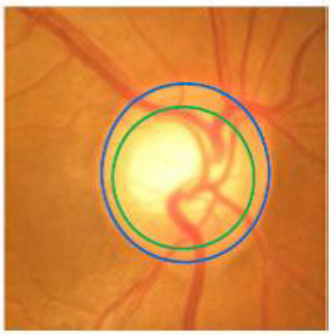
- Medidas de la PIO, mediante tonometría o algún método alternativo que permita obtener un valor de presión.
- Evaluación del campo de visión del paciente, ya que el glaucoma se caracteriza por una pérdida de la visión periférica.
- Medida de parámetros anatómicos, de manera que se haga un seguimiento de la evolución del ojo del paciente con la enfermedad. Estas pueden basarse en la inspección de las fibras nerviosas del nervio óptico, o bien en el CDR (Cup to Disk Ratio), parámetro en el que se centrará el trabajo.

Para calcular el CDR, es necesario distinguir entre dos zonas del ojo para identificar esta enfermedad: el disco óptico y la excavación. En la Figura 1.1 se puede observar una imagen ilustrativa de estas zonas tanto en un ojo sano (Figura 1.1a) como en otro enfermo (Figura 1.1b). El área rodeada por la línea azul se corresponde con el disco óptico, mientras que la que está rodeada por la línea verde se corresponde con la excavación.

Además, al comparar ambas imágenes, es resaltable el aumento del tamaño de la excavación en caso de padecer glaucoma, como se aprecia en la Figura 1.1b. Por consiguiente, retinografías con una excavación mayor tendrán un CDR mayor y, por tanto, serán posibles enfermos[1].



(a) Retinografía de un ojo sano



(b) Retinografía de un ojo enfermo de glaucoma

Figura 1.1: Comparativa de un ojo sano y otro enfermo de glaucoma. Fuente: Balan et al., 2020, p. 7

A la hora de calcular la CDR, las máquinas pueden ser de gran ayuda debido a que, una vez programadas, siempre tratarán un problema de la misma manera, sin verse afectados por factores externos, como puede ser la fatiga o la atención. Por tanto, en tareas en las que se disponga de una gran cantidad de retinografías, las máquinas podrían separar aquellos casos claramente sanos del resto.

Sin duda alguna, situaciones como la vivida este año, en la que la pandemia de la COVID-19 ha saturado los hospitales de enfermos, pone de manifiesto que el uso de este tipo de tecnologías puede ayudar a los profesionales a realizar su trabajo de una manera igualmente válida en un intervalo temporal inferior. Por consiguiente, se precisaría de un menor tiempo para el diagnóstico del glaucoma, permitiendo una reorganización de los recursos que ayudaría a solventar picos de afluencia en situaciones de extrema necesidad.

Debido a ello, diversos estudios a lo largo de todo el mundo han buscado poner solución a la segmentación de retinografías utilizando distintos enfoques, la mayoría centrados en el procesado de imágenes (Balan et al., 2020). Sin embargo, se trata de un tema de investigación abierto, ya que no se ha llegado a una solución clara, y aún hay muchos proyectos en desarrollo.

Por otro lado, a la hora segmentar retinografías, uno de los principales problemas es la ubicación del nervio óptico, que suele atravesar tanto el disco óptico como la excavación,

[1]Es necesario recordar que hay varios indicadores para identificar el glaucoma, y es necesario prestar atención a varios de ellos para estar seguro del diagnóstico de la enfermedad antes de comenzar con ningún tratamiento.

y que dificulta enormemente la segmentación de estas imágenes. Es por ello que, en este trabajo, se hace una propuesta centrada en la identificación de estas zonas del ojo mediante la búsqueda de patrones en la imagen. En este caso, las segmentaciones serán realizadas por un modelo de Inteligencia Artificial (IA), más concretamente de Deep Learning (DL), denominado Deeplabv3+, que "ha establecido una nueva meta en el estado del arte en lo que respecta a los conjuntos de datos PASCAL VOC 2012 y Cityscapes" (Chen, Zhu et al., 2018, p. 14).

En este sentido, los últimos avances en IA, y especialmente en el DL, son altamente remarcables, y han cambiado la forma en la que se resuelven muchos de los problemas que antes requerían un duro planteamiento por parte del ingeniero. Desde esta perspectiva, se busca aplicar una solución enmarcada dentro de este campo. En concreto, este trabajo se centra en el entrenamiento de Deeplabv3+ para el propósito de la segmentación de la excavación y el disco óptico en retinografías, que son especialmente útiles a la hora de medir el CDR. Para lograr esta meta, se utilizarán dos bases de datos, que contienen segmentaciones tanto del disco óptico como de la excavación realizadas por expertos: RIMONEr3 y RIGA.

Para comprender todo lo relativo a este proceso, se comenzará definiendo algunos conceptos necesarios para conocer el campo en el cual se enmarca este trabajo, así como la red neuronal utilizada para realizar las segmentaciones, y las métricas con las que se evaluará la bondad del modelo. Visto esto, el lector se encontrará en disposición de comprender los resultados obtenidos en el trabajo y, por tanto, las conclusiones finales a las que se llega.

2 Contextualización

2.1. Inteligencia Artificial

Para comenzar, es preciso conocer qué se entiende por Inteligencia Artificial (IA). Esta pretende imitar las funciones cognitivas del ser humano, es decir, busca automatizar tareas intelectuales que habitualmente realizamos. Se trata de un campo interdisciplinar tan amplio que puede resultar abstracto. Contiene un gran conjunto de ciencias de ámbitos muy variados, que muy difícilmente se podrían imaginar interrelacionadas: Lógica, Estadística, Informática o Psicología pueden encontrarse en este campo.

Esta ciencia ha tenido un desarrollo muy lento, que no sería posible entender sin la influencia de autores como McCulloch, Pitts y Hebb, que desarrollaron el primer modelo matemático de red neuronal y lo conectaron a la lógica proposicional, abriendo un nuevo campo de estudio. Alan Turing, unos pocos años después, realizaría su aportación más relevante en el campo: el test de Turing, que pretendía determinar cuándo se consideraba a una máquina como inteligente. Siguiendo esta línea, no es hasta la histórica conferencia de Dartmouth en el año 1956 donde John McCarthy acuñó por primera vez el término de «Inteligencia Artificial», lo que es considerado como el nacimiento de la misma (Ertel, 2017).

La historia de este campo es la de una montaña rusa, en los que períodos de intenso interés, investigación y expectativa (conocidos como «Veranos de IA») son seguidos de épocas de crítica y decepción como consecuencia de no haber cumplido las expectativas impuestas tras años de espera (los «Inviernos de IA») (Chollet, 2018; Torres, 2020). Estos últimos inundarán de decepción el último cuarto del siglo XX.

Los períodos mencionados, unidos al exponencial desarrollo de las ciencias de la computación vivido en el último siglo, han dado lugar a definiciones y enfoques muy distintos en este campo: redes bayesianas, máquinas de soporte vectorial y árboles de decisión son algunos ejemplos actuales de la alta ramificación de la IA. A la hora de diseñar un sistema «inteligente», se cuenta con dos posibles planteamientos totalmente contrapuestos: el primero se centra en el estudio del pensamiento y hábitos del ser humano, para poder enseñar

a las máquinas a comportarse como nosotros; el segundo explora el problema planteado y busca la solución más óptima a través de un enfoque matemático (Ertel, 2017).

La historia de la IA, como se ha visto, pasa de estar en un segundo plano en sus inicios, con las aportaciones de Alan Turing, a estar en un primerísimo primer plano hoy en día. Los avances en los últimos años han sido increíbles y el Deep Learning, que se nutre del Machine Learning y de las redes neuronales, ha sido el gran responsable de esto. Actualmente, son numerosas las tareas en las que las máquinas se están equiparando a las personas: reconocimiento de voz, procesado del lenguaje natural o visión por computador son solo algunos ejemplos. En palabras de François Chollet, creador de Keras, "se está viviendo el tercer verano de IA de la historia, protagonizado por el Deep Learning, que está siendo el más prolongado y satisfactorio de todos" (Chollet, 2018, p. 315).

2.2. Machine Learning

En muchos casos a lo largo de la historia de la humanidad, distintos investigadores han llegado a conclusiones similares, o incluso a la misma, sin saberlo. Este es el caso de la *back propagation*, que fue por tercera vez descubierta por David E. Rumelhart, y que se basa en reutilizar el error, medido como la diferencia entre la predicción que debería haber hecho el modelo y lo que realmente ha predicho, de manera que se minimice a través de un bucle. Se usa este error para mejorar el modelo progresivamente, hasta optimizarlo.

El redescubrimiento de la *back propagation* sirvió al Machine Learning (ML) para revolucionar la idea de programación: antes se proporcionaban reglas y datos para obtener respuestas, y ahora se obtenían reglas a partir de datos y respuestas; se pasó de un sistema de reglas condicionales a otro basado en el conocimiento y en la forma del ser humano de reconocer su propio entorno (ver Figura 2.1). Este «boom» del ML fue tal que aún llega hasta nuestros días, y la llegada de un nuevo agente, el Deep Learning, otorgará mayor relevancia a esta rama (Chollet, 2018).

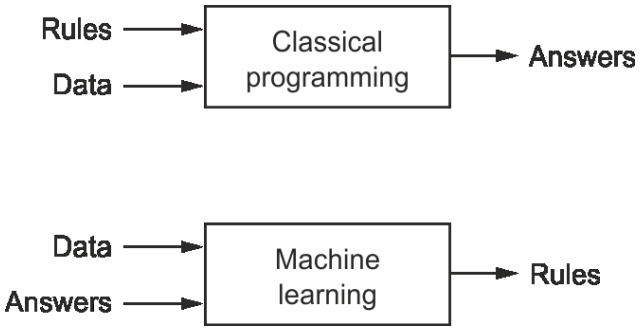


Figura 2.1: Diferencias entre la programación clásica y el Machine Learning. Fuente: Chollet, 2018, p. 5

En este sentido, esta rama de la IA va un poco más allá que su superiora, ya que lo que

trata de hacer es un poco más complejo: dar a las máquinas las herramientas necesarias para que ellas mismas puedan resolver un problema concreto de manera indirecta. Esto es, que las máquinas adquieran conocimientos, aprendan, de manera similar a la que los humanos lo hacemos, con el objetivo de resolver un problema concreto. Este proceso es más de generalización que de especificación; el aprendizaje funciona de tal manera que es necesario aprender las características particulares de un tipo de objeto para después asociarlas a todos los objetos de ese tipo.

Este campo contiene un gran número de algoritmos, que se pueden agrupar en cuatro categorías, aunque en algunos casos concretos las fronteras entre ellas no son del todo sólidas (Chollet, 2018). A continuación, se presentan brevemente.

Aprendizaje supervisado

Este enfoque comprende modelos en los que se usan las etiquetas (*labels*, la solución deseada) para entrenar el modelo. Lo que se busca en este caso es aprender una función que mapee los datos de entrada a la salida deseada, proporcionándole una serie de ejemplos. Este es el caso más usual en Deep Learning.

Los ejemplos más claros de esta rama pueden ser el reconocimiento óptico de características, reconocimiento del lenguaje, clasificación de imágenes o traducción de idiomas, aunque también hay unas variantes un poco más específicas, como la generación de texto a partir de imágenes, la detección de objetos o la segmentación de imágenes, punto en el que se centrará en este trabajo.

Aprendizaje autosupervisado

Es muy similar al caso anterior, con la diferencia de que las etiquetas no son anotadas por los humanos, sino que lo son por sistemas automáticos a partir de otros datos o heurísticos. Un muy buen ejemplo de este caso es el de autocodificadores, que se basan en enlazar las estructuras de codificador y decodificador. Se suelen usar junto con otras redes, y son muy útiles a la hora de proporcionar a la red datos libres de ruido, o de reducirlos dimensionalmente (Dertat, 2017).

Aprendizaje no supervisado

Son aquellos modelos en los que, a diferencia del aprendizaje supervisado, no se incluye en el entrenamiento las etiquetas, obligando al modelo a clasificar la información por sí mismo. En este caso, los ejemplos son mucho menos aplicados, y suelen estar relacionados con los datos: división de los datos en grupos y la asociación de los resultados con los datos.

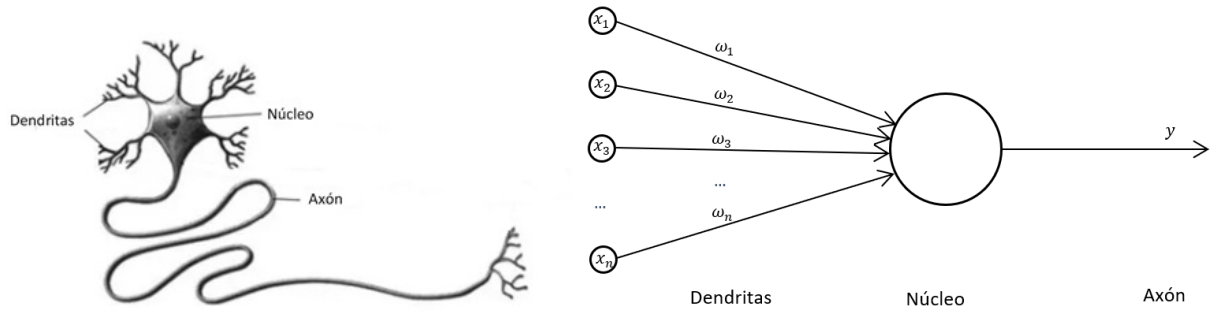
Aprendizaje por refuerzo

El aprendizaje por refuerzo se centra en dejar que el modelo explore libremente las soluciones, y un agente externo (el programador) recompensará o penalizará las decisiones del modelo. Ejemplos representativos de esta categoría son aquellos modelos centrados en aprender a jugar a juegos, como «Snake», en el que se controla una serpiente que tiene que comer fruta y evitar chocarse. Hay numerosos casos en Internet en los que se intenta entrenar modelos para jugar a este juego y uno de ellos es el proporcionado por el usuario de GitHub llamado greerviau, que incluso contiene un vídeo en el que se muestra el proceso de aprendizaje hasta llegar a un muy buen resultado.

Otro caso más meritorio es el de Alphastar, una IA de la empresa DeepMind, que pertenece al gigante Google. Este modelo se centra en Starcraft II, uno de los juegos de estrategia en tiempo real más complicados de jugar, y que consigue a superar al 99.8 % de los jugadores (Statt, 2019). Las decisiones tomadas por el modelo solamente se podían recompensar o penalizar, consiguiendo que el algoritmo se adapte casi a la perfección al juego.

2.3. Redes neuronales

Como se ha expuesto en las líneas anteriores, la evolución de la IA y la de sus distintos campos no se puede entender sin el desarrollo de la Lógica, la Psicología y la Ciencia Cognitiva. Fueron estos campos los que aportaron luz al desarrollo de las redes neuronales artificiales, que se basan en el comportamiento biológico del cerebro humano y en sus unidades estructurales, las neuronas.



(a) Neurona biológica. Fuente: Anónimo, s.f.-e

(b) Neurona artificial. Fuente: creación propia

Figura 2.2: Comparativa entre neurona biológica y artificial

En las figuras 2.2a y 2.2b se puede apreciar algo bastante obvio: el modelo matemático de la neurona biológica es muy pobre. Las dendritas, que facilitan la conexión entre distintas neuronas, se ven representadas en el modelo a través de las entradas x_i donde $i = 1, 2, ..., n$. Además, los pesos ω_i servirán para destacar más unas entradas sobre otras. El núcleo se encarga de procesar de alguna manera la información que le llega a través de las dendritas,

y de proporcionar el resultado *y* a través del axón. Esta es la primera representación de una neurona, que daría lugar a las redes neuronales complejas que se pueden encontrar hoy en día, y que se abordan en los siguientes subapartados.

2.3.1. El perceptrón

El perceptrón, la red neuronal más simple posible, fue diseñado en 1957 por Frank Rosenblatt. Es el primer ejemplo de red neuronal (Torres, 2020), que modifica muy levemente el modelo neuronal creado por McCulloch y Pitts. Consta de una única capa de entrada y un nodo de salida, que realiza dos operaciones sobre las entradas: una primera, que está representada con un sumatorio, es lineal, y realiza la suma ponderada de sus entradas a través de los pesos. La segunda etapa, conocida como función de activación, se encarga de realizar una operación no lineal, y está representada por la función escalón.

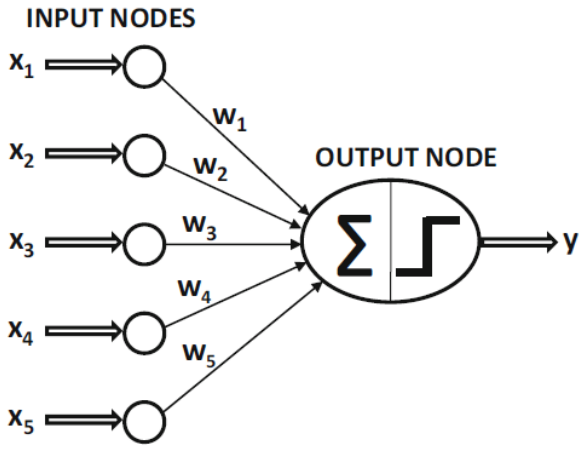
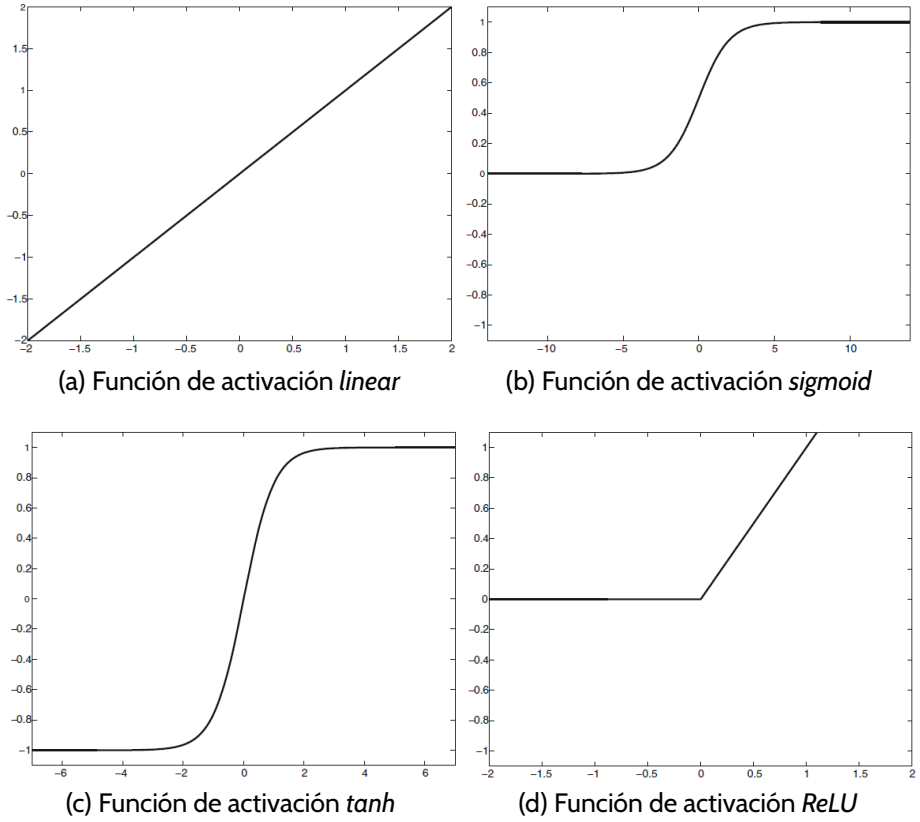


Figura 2.3: Esquema de un perceptrón. Fuente: Aggarwal, 2018, p. 5

La función de activación juega un papel muy importante. No todos los conjuntos de datos son linealmente separables, por lo que añadir una no linealidad permitirá realizar una mejor clasificación de estos. Algunas de las funciones de activación más importantes pueden encontrarse representadas gráficamente en la Figura 2.4.

Aun con estas modificaciones sobre el primer modelo de neurona artificial, el perceptrón tiene sus limitaciones, como el problema XOR (Skansi, 2018), que demuestra que el perceptrón es incapaz de implementar la función XOR, es decir, dados los mismos datos de entrada, no es capaz de proporcionar las mismas salidas que esta función.



(a) Función de activación *linear* (b) Función de activación *sigmoid*

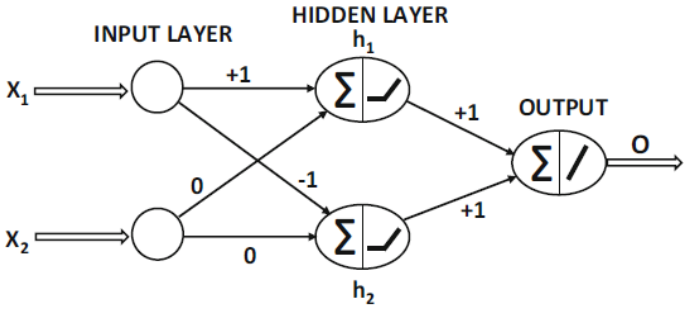
(c) Función de activación *tanh* (d) Función de activación *ReLU*

Figura 2.4: Distintas funciones de activación. Fuente: Aggarwal, 2018, p. 13

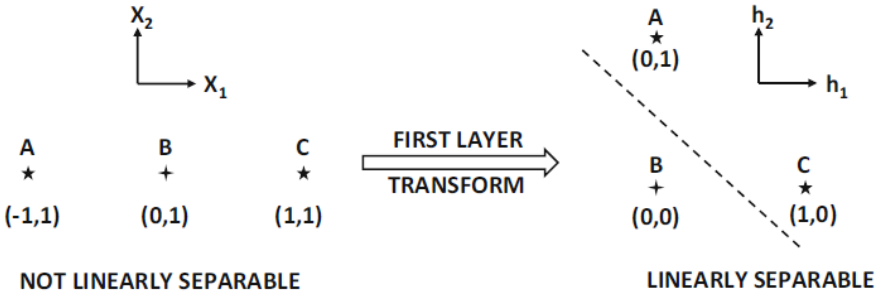
2.3.2. El perceptrón multicapa

Para resolver esto, la solución adoptada fue la más simple: conectar varias capas (conformadas por varios perceptrones en paralelo) en cascada, de manera que las salidas de una capa conecten con la entrada de la siguiente. Se puede ver un esquema básico en la parte inferior de la Figura 2.5a.

De esta manera, lo que se consigue al concatenar varios perceptrones, es que los datos que no podían ser clasificados por la primera capa pudieran ser organizados de una manera más fácilmente clasificable para la siguiente capa, gracias a la activación no lineal. La Figura 2.5b muestra un ejemplo bastante gráfico. Los datos que se introducen en la primera capa no son separables linealmente, por lo que se les pasa por una capa intermedia con dos perceptrones, cuya función de activación es ReLU, que transforma los valores negativos en 0, y no aplica ninguna transformación a los valores positivos. Esta función deja en bandeja la separación lineal al siguiente perceptrón.



(a) Perceptrón multicapa



(b) Transformación de los datos de una capa a otra en el perceptrón multicapa

Figura 2.5: Perceptrón multicapa y tratamiento de los datos. Fuente: Aggarwal, 2018

En esta nueva estructura es posible definir una capa extra a las de entrada y salida[1]: la capa oculta, que es la capa (o capas) situadas entre la entrada y la salida, cuyas operaciones no son visibles para el usuario. Las capas ocultas pueden ser tantas como se requiera, lo que da lugar a grafos muy enrevesados y computacionalmente pesados.

2.3.3. Redes neuronales convolucionales

En este caso, se vuelve a encontrar un ejemplo de inspiración biológica: las ideas de David H. Hubel y Torsten Weisel presentadas en 1968, y basadas en el córtex visual de los gatos, llevaron a Yann LeCun y otros a desarrollar las redes neuronales convolucionales en 1998 (Skansi, 2018).

La idea de este tipo de redes es, precisamente, introducir la operación de la convolución, y sus características deseables asociadas en las redes neuronales. Para ello, dado una capa q de la red, se utiliza una serie de filtros (o kernels) usualmente cuadrados. Estos tienen una determinada altura y anchura ($H_q = B_q = F_q$), y una profundidad (d_q), que dependerá de la profundidad de la matriz de datos a convolucionar: 1 si es una imagen en blanco y negro, 3 si es una imagen RGB, o mayor si son mapas de características. En este caso, los pesos no funcionan igual que en las redes neuronales convencionales; habrá un peso por píxel del filtro. Si se toma el filtro 2D de la Figura 2.6, con $F_q = 3$ píxeles, se obtendría $F_q^2 = 9$ pesos en

[1]En el caso del ejemplo, sería la capa intermedia.

el filtro, uno por píxel.

Figura 2.6: Filtro 3x3 usado como ejemplo. Adaptado de: Chollet, 2018, p. 127

Estos filtros se encargan de recorrer los datos de entrada, en este caso imágenes, de izquierda a derecha y de arriba hacia abajo, multiplicando cada uno de los píxeles sobre los que se sitúa por sus pesos. En la Figura 2.7 se presenta un ejemplo del barrido que realizaría el filtro anterior sobre un parche de dimensiones $H_q = W_q = 5$. El tamaño de la matriz de datos tras haber realizado la convolución en esta capa q sería, por tanto, de una altura igual a $H_{q+1} = H_q - F_q + 1$ y una anchura de $W_{q+1} = W_q - F_q + 1$, que en el presente ejemplo sería de $H_{q+1} = W_{q+1} = 3$.

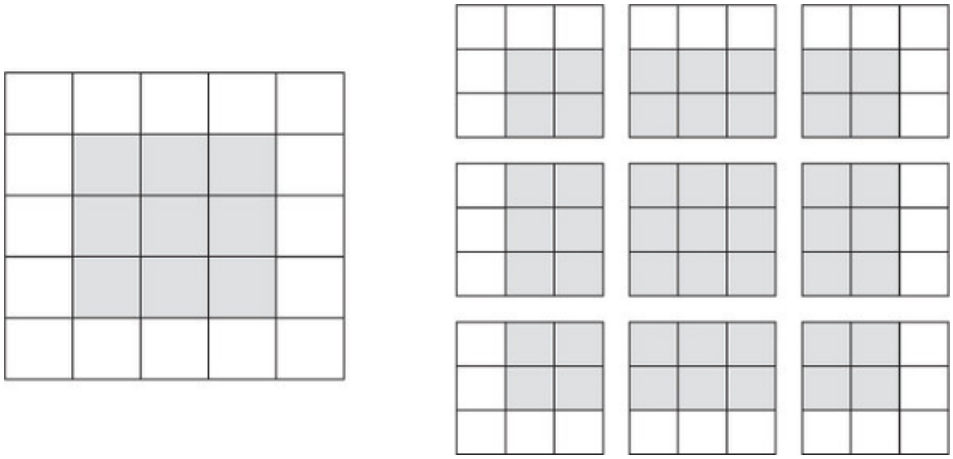


Figura 2.7: Posiciones válidas del filtro 3x3 en una matriz 5x5. Fuente: Chollet, 2018, p. 126

Se está ahora, por tanto, en disposición de definir formalmente la operación de la convolución entre las capas q y $q + 1$:

$$h_{ijp}^{(q+1)} = \sum_{r=1}^{F_q} \sum_{s=1}^{F_q} \sum_{s=1}^{d_q} \omega_{rsk}^{(p,q)} h_{i+r-1,j+s-1,k}^{(q)} \qquad \forall i \in 1, ..., L_q - F_q + 1 \tag{1}$$

Donde los subíndices i, j y p recorren cada una de las posiciones del mapa de características a convolucionar, y los subíndices r, s y k recorren las posiciones del filtro. Los superíndices p y q se refieren, respectivamente, al filtro y a la capa de la red en la que se sitúa. Por último, ω y h son los pesos del filtro y los píxeles del mapa de características, respectivamente.

Otro aspecto importante de estas redes es que tienen dos propiedades muy útiles (Chollet, 2018):

- Los patrones aprendidos son invariantes a la traslación: una vez aprendido un patrón, el filtro es capaz de localizarlo en cualquier otra parte de la imagen.
- Los patrones se aprenden de manera jerárquica: las primeras capas aprenden trazos más simples, que en la siguiente capa se combinan para formar otros trazos un poco más complejos, y así sucesivamente hasta formar objetos completos.

Esta transferencia de información de una capa a otra se hace de la siguiente manera: una vez realizada la convolución con la imagen, se obtendrá una serie de mapas de características, que contienen información de las imágenes de entrada, y que serán los que se transfieran a la capa siguiente. Antes de transferirlos, se suele realizar una operación de *pooling*, que se basa en coger distintas ventanas de los mapas de características (que no tienen por qué ser iguales a los filtros) y tomarlas como un único píxel, realizando un *downsample* de la información. Hay principalmente dos formas de hacerlo: el *max-pooling*, que se basa en tomar el valor máximo de esa ventana, y el *average-pooling*, que se basa en tomar el valor medio. En la Figura 2.8 se puede ver un ejemplo de *max-pooling* realizado con una ventana de 2x2. Esta técnica ayuda a forzar el aprendizaje jerárquico de patrones, a la vez que se ahorra un cierto espacio de memoria (Chollet, 2018).

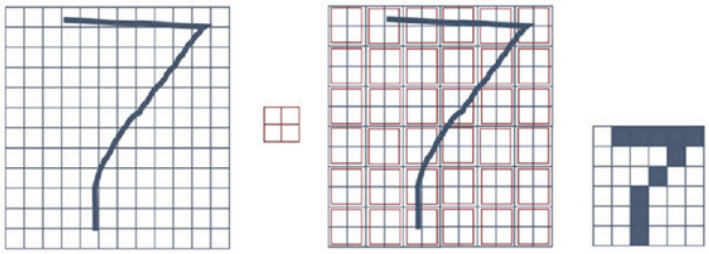


Figura 2.8: Ejemplo de max-pooling con una ventana 2x2. Fuente: Torres, 2020, p. 164

Además del tamaño de los filtros, hay dos parámetros de interés en las capas convolucionales: el *padding* y el *stride*. El *padding* surge como solución a la pérdida de información asociada a los bordes al realizar la convolución, y consiste en rodear la imagen (o la matriz de valores) por números. Este *padding* suele hacerse utilizando ceros, lo que es conocido como *zero-padding*, aunque puede realizarse de varias maneras. Para saber el número de valores que se debe poner alrededor del mapa de características para no disminuir su tamaño, se tiene que añadir añadir: $(F_q - 1)/2$ ceros. Por tanto, si se aplica un *padding* de uno a la Figura 2.9, dará como resultado lo que muestra la Figura 2.10, en la que se puede ver que, si se aplicara el *pooling*, se obtendrían las mismas dimensiones que la matriz inicial; una matriz de 5x5 (Torres, 2020).

Por otro lado, el *stride* permite controlar la longitud del paso con la que el filtro recorre

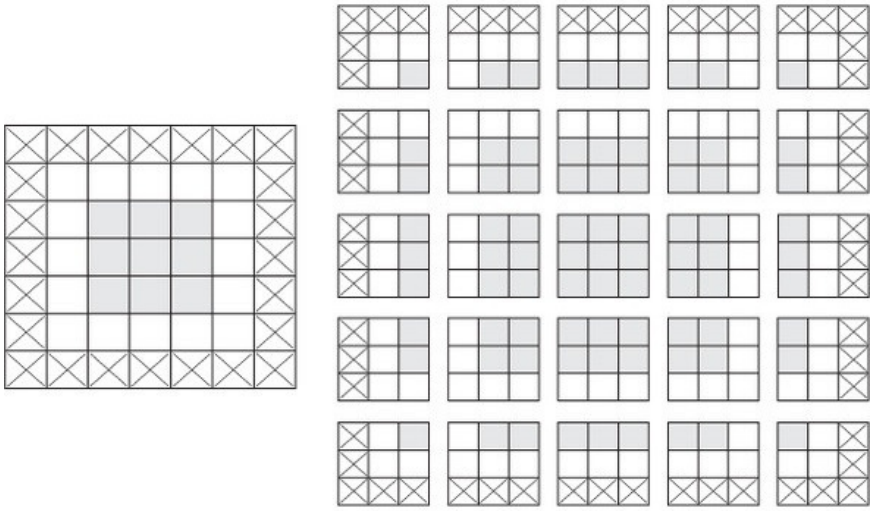


Figura 2.9: Ejemplo de filtro 3x3 con padding igual a uno. Adaptado de: Torres, 2020, p. 127

la matriz de datos. Usualmente tiene valor igual a uno (que se corresponde con la convolución genérica), pero en la Figura 2.10 se presenta un ejemplo de un *stride* de dos. En este caso, se obtiene una matriz de datos de 2x2 a partir de una de 5x5. Este resultado sigue las expresiones $(L_q - F_q)/S_q + 1$ $(B_q - F_q)/S_q + 1$. Esta técnica permite reducir espacialmente las dimensiones del mapa de características (Torres, 2020).

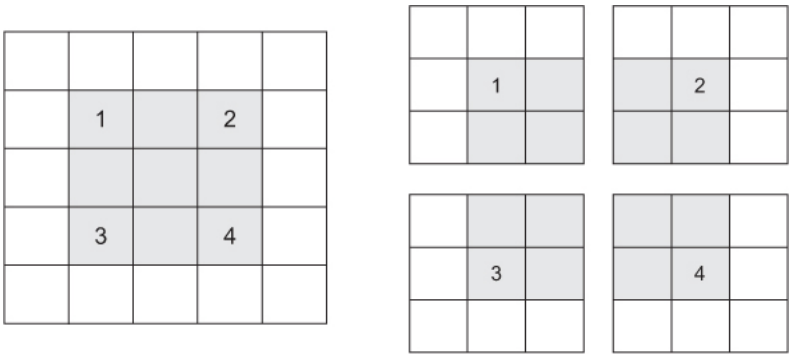


Figura 2.10: Ejemplo de filtro 3x3 con stride igual a dos. Fuente: Chollet, 2018, p. 126

Este tipo de redes no solamente son mejores en tareas de visión computacional, debido al aprendizaje de patrones, sino que tienen un número de parámetros mucho menor, lo que las hace también más eficientes en este caso.

2.4. Deep Learning

El Deep Learning (DL) es un subcampo del ML. Nace tanto de las contribuciones arquitectónicas de las redes neuronales como de los distintos algoritmos desarrollados por el ML. La unión de estos dos mundos dio como resultado, a mediados de 2010, lo que hoy en día es la forma más precisa de programar una máquina con IA. En la Figura 2.11 se puede ver

una comparativa entre el DL y el ML convencional en términos de datos y precisión. En este sentido, el DL es muy superior a cualquier otro enfoque, consiguiendo resultados mucho mejores.

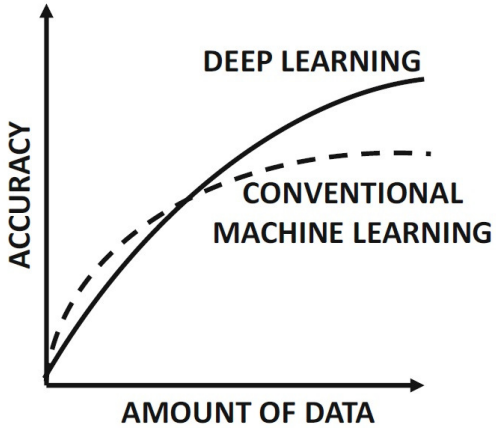


Figura 2.11: Comparativa de la precisión del ML y el DL. Fuente: Aggarwal, 2018, p. 3

Para considerar un modelo como «Deep», es necesario tener una gran cantidad de capas ocultas, lo que es una de las causas directas de que el DL no se haya desarrollado antes.

Un rápido desarrollo

Las bases del DL fueron asentadas a lo largo de la segunda mitad del siglo XX, pero no ha sido hasta los últimos años que se ha podido desarrollar y perfeccionar. Esto es debido, principalmente, a tres factores (Chollet, 2018):

- El desarrollo de hardware especializado: no solamente hay que tener en cuenta los avances en las CPUs (Central Processing Unit), sino también la influencia del mercado del videojuego, que ha contribuido a desarrollar GPUs (Graphics Processing Unit) mucho más rápidas y potentes, especialmente útiles para acelerar cálculos en coma flotante. Además, también es destacable el desarrollo de nuevas herramientas, como las TPUs (Tensor Processing Unit), que son circuitos integrados usados para acelerar las operaciones de cálculo tensorial, correspondientes al entrenamiento de la red (Anónimo, s.f.-a).
- Los conjuntos de datos: el desarrollo de Internet ha sido fundamental en este sentido. Hoy en día hay infinidad de imágenes y texto listo para ser usado por la red. Además, sitios web como IBM o ImageNet son muy útiles a la hora de proporcionar una base de datos con la que entrenar una red.
- Nuevos algoritmos: en los últimos años ha habido un gran número de avances y mejoras en los algoritmos usados para optimizar una red neuronal. Además, las competi-

ciones masivas de ML, especialmente las organizadas en Kaggle han causado un afán aún mayor por intentar mejorar los modelos y desarrollar nuevas técnicas.

Por último, movimientos como la denominada «democratización del DL» (Chollet, 2018) han provocado una mayor afluencia de nuevos usuarios al DL. En esta, numerosos investigadores han desarrollado herramientas de libre acceso como Keras o TensorFlow, así como la publicación abierta de la mayoría de los avances en el campo a través de sitios web como arXiv. Además, el entorno gratuito Google Colaboratory, nombrado anteriormente, ha facilitado el acceso a nuevos interesados en el campo, haciendo que hoy en día cualquier persona con acceso a internet y un mínimo de interés en el campo pueda formarse.

Problemas típicos

En el apartado correspondiente al ML se ha indicado que el trabajo se centraría en el aprendizaje supervisado. Dentro de este subcampo, hay principalmente tres tipos de problemas:

- Clasificación: es el tipo más habitual, y el que se abordará más adelante. En él, se le proporcionarán las herramientas necesarias a una red para que divida los datos en varias clases. Dependiendo del número de clases que se quiera clasificar, se tienen dos tipos de problema:
  - Clasificación binaria: hay solamente dos clases. Un ejemplo sería el caso en el que se tenga una red que clasifique imágenes de animales entre perros y gatos.
  - Clasificación múltiple: se tienen más de dos clases, como clasificar entre distintas razas de perros.
- Regresión: en este caso, en lugar de predecir resultados discretos (la pertenencia a una clase u otra), se predicen resultados continuos, como puede ser el precio de una vivienda o la temperatura de un día concreto (Chollet, 2018).

El procesado de los datos

En el caso que compete, clasificación, se va a tener una serie de imágenes en un único conjunto de datos. Lo que se tendrá que hacer es dividirlo en tres subconjuntos: entrenamiento, validación y test. El conjunto de entrenamiento será el empleado para entrenar a la red, por lo que debe ser el más grande. El conjunto de validación será utilizado para dar una primera idea de que la red está generalizando correctamente, y no se está produciendo *overfitting*, concepto que se explorará más adelante. Por último, el conjunto de test, que la red no ha visto en ningún momento, será el que confirme la bondad del modelo, y el que se debe observar a la hora de decidir el valor de los hiperparámetros.

Estas imágenes estarán representadas en forma de *tensor*, que son la generalización

de escalares (tensor 0D), arrays (tensor 1D) y matrices (tensor 2D) a cualquier dimensión (Kan, 2018). En caso de trabajar con imágenes, se estará hablando de tensores 4D, y cuyas dimensiones serán:

1. Número de muestras.
2. Canales (1 si la imagen es en blanco y negro o 3 si es RGB).
3. Altura.
4. Anchura.

No obstante, en algunos casos, puede ser que se encuentren los canales como última dimensión. De cualquier manera, es bastante intuitivo saber que todas las imágenes deben tener los mismos valores en todas las dimensiones.

A la hora de proporcionarle los datos a la red es necesario codificarlos, para facilitarle el trabajo con estos. Principalmente, hay dos técnicas cuando se trabaja con clasificación: *label encoding* y *one-hot encoding*. El primero se basa en asignar números a cada clase: por ejemplo, si se trabaja con un modelo que pretende clasificar entre perros, gatos y patos, se le asignaría un número a cada uno (1, 2 o 3). El problema de este enfoque es que se ordenan previamente las clases, lo que puede proporcionar resultados inesperados.

El segundo enfoque, que se corresponde con la codificación *one-hot*, evita este problema. Ello lo hace clasificando los datos de manera binaria creando una columna en los datos por cada clase existente, situando un 1 en las filas correspondiente a la clase. A continuación, se puede un ver ejemplo de cómo quedarían clasificados los datos en la Tabla 2.1 (DelSole, 2018).

Label encoding			*One-hot encoding*			
Animal	Clasificador	Datos	Perro	Gato	Pato	Datos
Perro	1	...	1	0	0	...
Gato	2	...	0	1	0	...
Pato	3	...	0	0	1	...

Tabla 2.1: Comparativa entre *label encoding* y *one-hot encoding*. Fuente: DelSole, 2018

Aprendizaje de una red neuronal

Hasta ahora, solamente se ha visto que la red es capaz de aprender características de una imagen concreta y realizar predicciones gracias a los pesos de cada una de sus capas. Sin embargo, no se ha explicado cómo se actualizan los pesos de manera que estas predicciones sean cada vez más acertadas. En la Figura 2.12 se esquematiza el funcionamiento global del entrenamiento de una red.

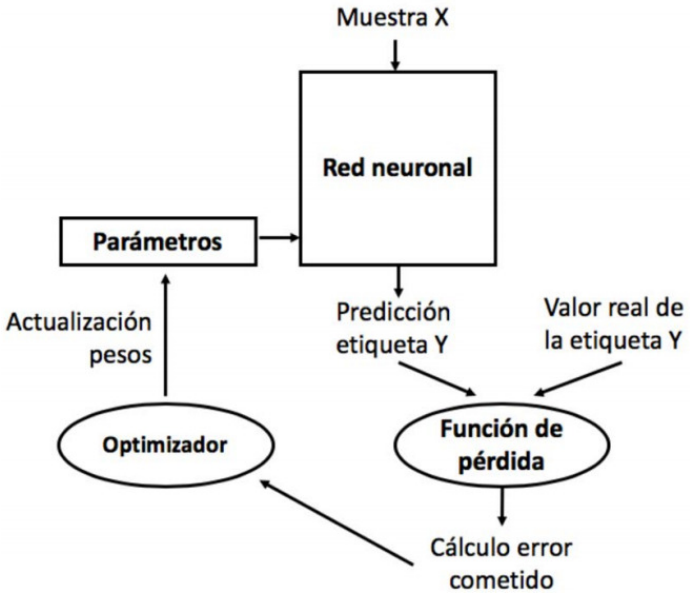


Figura 2.12: Esquema global del entrenamiento de una red neuronal. Fuente: Torres, 2020, p. 127

En primer lugar, se pasan los datos *X* a lo largo de la red, y se realiza las predicciones con los pesos inicializados a un valor concreto, ya sea aleatorio o predefinido. Con la salida *Y* de la red y la etiqueta correspondiente, que no es más que lo que se debería haber predicho, se mide el error con la función de pérdidas. Este error sirve al optimizador para actualizar los pesos del filtro e intentar minimizar el error. Al finalizar este proceso, comenzará una nueva iteración, realizando una predicción con los nuevos pesos.

Se puede, por tanto, entender el entrenamiento de una red como la unión de dos procesos. En el primero, denominado *forward propagation* (o propagación hacia delante), la red realiza las predicciones y mide el error. En el segundo proceso, conocido como *back propagation* (o propagación hacia atrás), el error medido se utiliza para modificar los pesos de manera que se minimice la diferencia entre la predicción y la etiqueta. Estos procesos ya se introdujeron en el apartado correspondiente al ML.

El agente principal en el proceso de *back propagation* es el optimizador, que básicamente implementa un algoritmo que calcula el gradiente del error cometido con respecto a cada peso, para que así se pueda conocer qué pesos contribuyen más a ese error. La modificación que aplica a cada peso será controlada gracias a un «paso» (o *learning rate*), es proporcional al gradiente calculado, y se debe hacer en dirección contraria a este. El optimizador suele aplicar el SGD (Stochastic Gradient Descent), o alguno de los algoritmos derivados de este, como el RMSProp (*Root Mean Square Propagation*) o al Adam (*Adaptative Moment Stimation*).

Una manera intuitiva de comprender lo que hace el optimizador es observar la Figura 2.13. La gráfica representa la evolución del error total con respecto al valor de los pesos. Gra-

cias a los mecanismos explicados anteriormente, el optimizador busca el valor que consigue minimizar el error yendo en contra del gradiente. Esto lo consigue dando un paso hacia el mínimo del error en cada iteración.

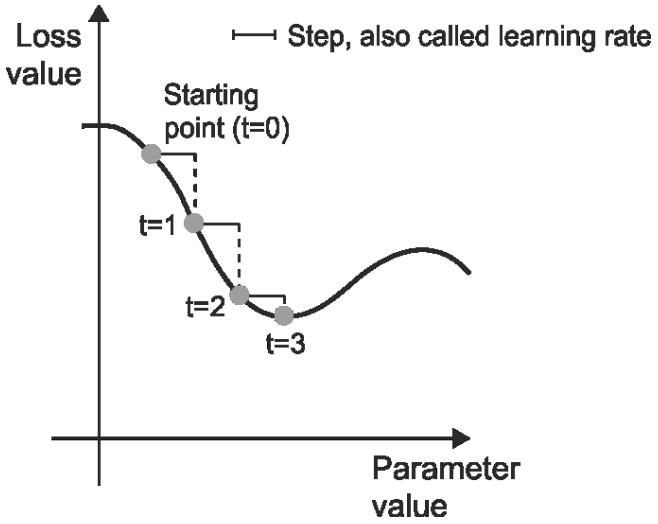


Figura 2.13: Descenso Estocástico del Gradiente. Fuente: Chollet, 2018, p. 49

Función de pérdidas y métricas

Los otros dos agentes que han de tenerse en cuenta a la hora de diseñar una red son la función de pérdidas y la métrica (o métricas) escogida. La primera servirá para hacerse una idea de cuán lejos está el punto del que se parte de lo que se quiere obtener, y debe escogerse dependiendo del tipo de problema ante el que se encuentre (Torres, 2020). En este sentido, también será muy importante la función de activación de la última capa de la red neuronal. La combinación más común de estas dos funciones, en función del tipo de problema, es la siguiente,:

- Clasificación: hay que elegir funciones de pérdidas que computen la probabilidad de pertenencia a una clase u otra.
    - Clasificación binaria: *binary_crossentropy* como función de pérdida y *sigmoid* como función de activación para la última capa.
    - Clasificación múltiple: *sparse_categorical_crossentropy* o *categorical_crossentropy* (si se está usando codificación *one-hot*) como función de pérdida y *softmax* como función de activación para la última capa.
- Regresión: se ha de tener en cuenta funciones de pérdida que calculen el error, del tipo *MSE* (Mean Square Error). No es necesario tener función de activación en la última capa.

En cuanto a las métricas, servirán para observar la evolución del aprendizaje de la red, y para detectar el *overfitting*. Tienen un objetivo que es bastante similar al de la función de

pérdidas, con la diferencia de que no se tienen en cuenta a la hora de entrenar la red; es simplemente orientativa. Ejemplos de métricas son ACC (*accuracy*) o AUC (*Area Under the Curve*).

Hiperparámetros de la red

Si se suele decir que el ML es más un arte que una ciencia es debido a los hiperparámetros. Hacer que un modelo funcione mejor o peor depende del valor que se le dé a estos, ya que necesitan ser determinados de manera externa por el programador. Un hiperparámetro, toda aquella variable que determina la estructura o el entrenamiento de la red, no puede ser aprendido por la red. La búsqueda del mejor valor suele estar influenciada o bien por la experiencia o bien por la prueba y error. Es importante no confundir este concepto con los parámetros de la red, que sí se aprenden automáticamente. Dentro de los hiperparámetros, se distinguen dos tipos (Torres, 2020):

- Hiperparámetros estructurales: número de capas de la red, número de neuronas por capa, funciones de activación, inicialización de los pesos... Se volverá a ellos en apartados posteriores, cuando se trabaje con Deeplabv3+.
- Hiperparámetros de entrenamiento (Skansi, 2018):
  - Tamaño de lote (*batch size*): dado unos datos de entrenamiento, el tamaño de lote es el número de divisiones que se hace sobre el conjunto de datos inicial, para después pasarlos por la red durante la etapa de entrenamiento.
  - Número de épocas (*epochs*): una época es el proceso de pasar los datos hacia delante (*forward propagation*) y hacia atrás (*back propagation*) de la red. Por tanto, al definir el número de épocas de entrenamiento, se está limitando el número de pasos de los datos a lo largo de la red.

    Es importante no confundir el concepto de época con el de iteración, introducido en el apartado correspondiente al entrenamiento. Una iteración es un paso de un lote hacia delante y hacia atrás de la red: una época tendrá tantas iteraciones como el tamaño de *batch size* escogido.
  - Tasa de aprendizaje (*learning rate*): este hiperparámetro sirve para controlar la longitud del paso con la que el algoritmo va a modificar los valores de los pesos, como está ilustrado en la Figura 2.13. El valor que tome debe estar compensado: si es demasiado grande, el aprendizaje divergirá; si es grande, el aprendizaje será corto, pero puede que no alcance el mínimo; y si es demasiado pequeño puede tardar demasiado en alcanzarlo. En la Figura 2.14, se muestra lo que se pretende describir aquí.

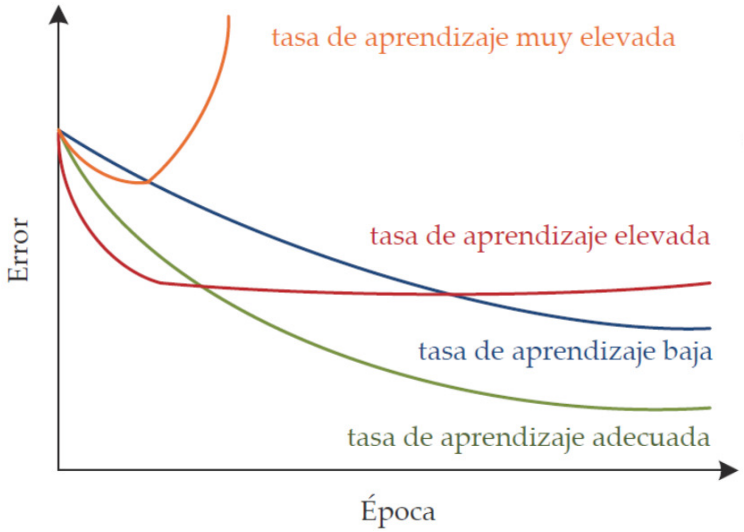


Figura 2.14: Evolución del error en función del Learning Rate seleccionado. Fuente: Berzal, s.f., p. 30

- Decaimiento de la tasa de aprendizaje (*learning rate decay*): es utilizado para solventar el problema que anteriormente se nombraba. A medida que avanzan las épocas, la tasa de aprendizaje disminuirá por un determinado factor (proporcional al *learning rate decay*), haciéndose más pequeño cuando se acerque al mínimo. De esta manera, se puede llegar a conseguir una tasa de aprendizaje óptima.
- *Momentum*: la motivación de usar este hiperparámetro radica en que existe la posibilidad de que el optimizador quede en un mínimo local en lugar de un mínimo global, que se puede ver en la Figura 2.15. El algoritmo que implementa se basa en la interpretación física de la inercia: cuando una bola baja una montaña, no se para justo al llegar a un pequeño valle sino que, debido a la velocidad que lleva, continúa un poco más. Para ello, es necesario tener en cuenta el paso realizado en la iteración anterior para realizar la actualización de los pesos.

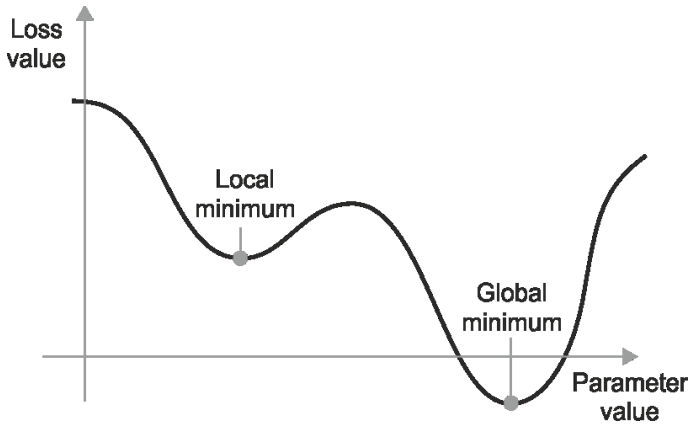
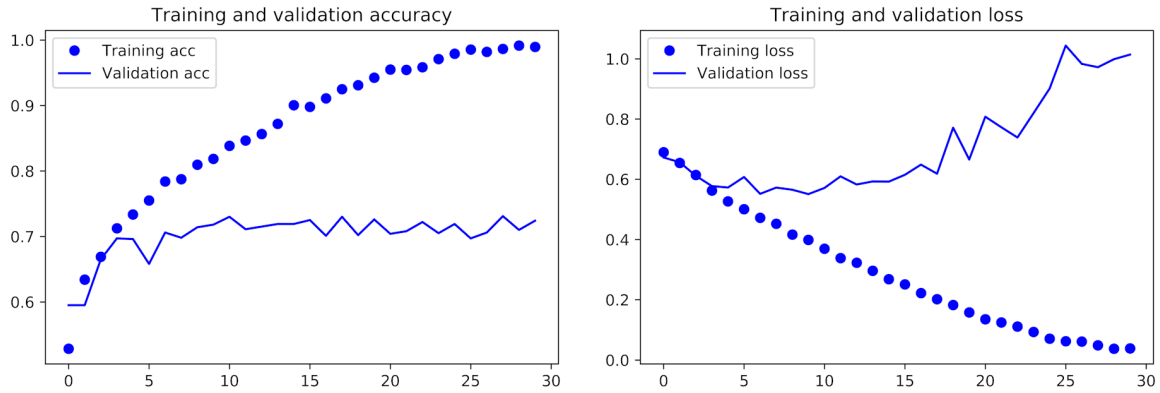


Figura 2.15: Momentum. Fuente: Chollet, 2018, p. 51

Overfitting

En este apartado se explorará el que es uno de los mayores enemigos del programador a la hora de entrenar una red: el sobreentrenamiento (*overfitting*). Este problema se refiere al hecho de que el modelo dé buenos resultados al trabajar con los datos de entrenamiento, pero no al trabajar con los de test. La red se ha adaptado demasiado a los datos de entrenamiento, y ha perdido la capacidad de generalizar.

Se puede ver un ejemplo de *overfitting* en la Figura 2.16. Al principio, el modelo tiene un comportamiento similar en los conjuntos de test y validación pero, a partir de la cuarta época, el modelo comienza a sobreentrenarse. Este problema es crítico en casos en los que los datos de los que se parte son muy pequeños y la red es compleja.



(a) Precisión (b) Pérdidas

Figura 2.16: Visualización del *overfitting*. Fuente: Creación propia

Si una de las principales tareas del programador es buscar el valor óptimo para los hiperparámetros, también lo es el hecho de usar técnicas para luchar contra el *overfitting*. Aquí se presentan algunas de las más utilizadas (Chollet, 2018):

- Disminuir el tamaño de la red: la manera más simple (y menos recomendable) para combatir el *overfitting* es disminuir la capacidad de aprendizaje del modelo eliminando capas o neuronas del modelo. De esta manera, se limitarán las características que la red puede aprender del conjunto de datos, haciendo que se adapte menos al conjunto destinado al entrenamiento. Esto solamente tiene sentido para problemas de clasificación relativamente fáciles, en los que la red tiene pocas características que aprender.
- Regularización: el otro enfoque a la hora de simplificar la red se basa en disminuir la entropía de la red, por medio de forzar a los pesos de la red a tomar valores bajos. Para lograr esto, se añade una función de coste a la función de pérdida, que perjudica los valores altos. Esta regularización puede ser fundamentalmente de dos tipos:

- Regularización L1: el coste añadido es proporcional al valor absoluto de los pesos.
- Regularización L2: el coste añadido es proporcional al cuadrado de los pesos.

  El uso de esta técnica tiene una desventaja: empeora el problema de *vanishing gradients* que, afortunadamente, no tienen cabida en las redes neuronales convolucionales, por lo que no será explicado (Aggarwal, 2018).

- *Dropout*: esta técnica, mucho más efectiva que las anteriores, consiste en «apagar» las neuronas de una capa durante una iteración con una determinada probabilidad. Este apagado solamente será efectivo durante la etapa de entrenamiento, mientras que en la de test no habrá ninguna neurona apagada. La idea es forzar que la red aprenda únicamente las características necesarias para generalizar correctamente.
- *Early-stopping*: la idea detrás de este método es detectar el momento en el que comienza el *overfitting* (cuando la tendencia de error del conjunto de validación deja de ser similar a la de entrenamiento), para parar el entrenamiento. Este método suele tener un parámetro de paciencia, que sirve para, una vez detectado el sobreentrenamiento, esperar un cierto número de épocas antes de dar por finalizado el entrenamiento. Cuando esto ocurra, se mantendrán los pesos que mejor resultado hayan dado sobre el conjunto de validación.
- Aumento de datos: otra forma bastante simple de mejorar la capacidad generalizadora de una red es usar más datos. Ante la imposibilidad de no poder acceder a más, nace el aumento de datos, que se basa en generar nuevos datos por medio de aplicar transformaciones aleatorias a los ya existentes: rotaciones, desplazamientos o zooms.

Otras técnicas

Batch normalization: esta técnica bastante usada es muy útil para frenar el fenómeno conocido como *internal covariate shift*, que se basa en que el cambio de los parámetros a lo largo del entrenamiento provoca un cambio en las variables intermedias ocultas, que ocasiona una convergencia más lenta durante el entrenamiento (Aggarwal, 2018). Al aplicar este método, que se basa simplemente en normalizar los datos antes de pasarlos por las capas, se consigue una convergencia más rápida y precisa.

Ajuste fino (*fine-tuning*): lo que se trata de hacer es, una vez entrenado el modelo, dejar «descongeladas» únicamente las últimas capas de la red, y entrenar el modelo con nuevos datos, haciendo que solo se modifiquen los pesos de las capas descongeladas (Chollet, 2018). Habitualmente suele hacerse con modelos preentrenados, a los que se les quiere añadir alguna capa más para intentar clasificar unos datos concretos dentro de los utilizados para preentrenar el modelo.

3 Desarrollo del proyecto

3.1. Materiales utilizados

Para el desarrollo práctico de este Trabajo de Fin de Grado se ha optado por utilizar, de entre todas las posibles opciones, las herramientas que se exponen a continuación.

Python

La elección de un lenguaje de programación como Python en lugar de R, (uno de los lenguajes más utilizados para ML), Java, MATLAB, Torch o JavaScript, se basa en dos pilares:

- La curva de aprendizaje: Python tiene una sintaxis más limpia y fácil de aprender que otros lenguajes, que lo hace más adecuado para nuevos usuarios de ML.
- La facilidad a la hora de trabajar con gran cantidad de datos: Python resulta mucho más simple que otros lenguajes, en parte por la gran cantidad de librerías utilizadas para DL, que facilitan enormemente la programación de redes neuronales profundas. Scikit-learn, NumPy, Keras y TensorFlow son solamente algunos ejemplos de las librerías usadas con este propósito (Srivastava, 2020).

Keras, TensorFlow y Scikit-learn

TensorFlow, desarrollado por Google, se ha convertido en los últimos años en uno de las principales entornos para el desarrollo de proyectos de DL. (Torres, 2020) Se trata de una herramienta muy versátil, que es capaz de funcionar simultáneamente en CPU y GPU, usando librerías de bajo nivel como CUDA y Eigen (Chollet, 2018).

Por otro lado, Keras, que es una API (*Application Programming Interface*) de IA utilizada para programar a alto nivel, es capaz de usar como soporte TensorFlow, CNTK (ahora conocido como Microsoft Cognitive Toolkit) o Theano para construir redes neuronales (Pradhan, 2016). De esta manera, simplemente mediante la definición de capas, Keras consigue simplificar enormemente las programación de un modelo de ML .

Scikit-learn es una biblioteca de código abierto que soporta tanto aprendizaje supervisado como no supervisado, proporcionando herramientas para el ajuste del modelo o el procesado de datos, entre otros.

Google Colaboratory

Por último, y como solución a la escasa potencia de computación disponible para poder entrenar adecuadamente una red neuronal, es destacable el uso de Google Colaboratory, que permite el acceso gratuito a capacidad de computación (tanto GPUs como TPUs), dando lugar a una mayor democratización del campo.

En DiBattista (2020) se hace una comparativa entre las distintas GPU proporcionadas por Colab, y otra adquirida por el propio autor del post. En este artículo se pone de manifiesto que no sólo se proporciona capacidad de computación gratuita, sino también de calidad. Colab pone en disposición de cualquier usuario de Google distintas gráficas de NVIDIA, empresa que en los últimos años ha decidido apostar de manera muy fuerte por el DL, convirtiéndose en líder absoluto en el mercado de GPUs destinadas a investigación.

3.1.1. Bases de datos

Resulta fundamental en un trabajo de este calibre nombrar las bases de datos con las que se procederá a entrenar el modelo con el objetivo de segmentar retinografías. De esta manera, se dispone de dos bases de datos, una primera con la que buscará optimizar la red (RIMONEr3), y otra que dispone de un mayor número de imágenes (RIGA), con la que se pretende llevar a la red hasta su mejor resultado.

RIMONEr3 es una base de datos abierta, que contiene 159 segmentaciones de retinografías, tanto de ojos sanos como de distintos grados de enfermedad por glaucoma, y han sido realizadas manualmente por dos expertos (MIAG, 2015). Por su parte, RIGA une tres bases de datos distintas, proporcionando un total de 750 imágenes, que han sido segmentadas manualmente por seis expertos (Almazroa et al., 2018). El desarrollo del proyecto tratado parte del uso de estas bases de datos, sin las cuales no podría haberse desarrollado.

3.2. Metodología

Previo al desarrollo de este trabajo, hubo una etapa de documentación sobre el campo de la Inteligencia Artificial en el que está enmarcado el mismo: la lectura de "Deep Learning with Python", de François Chollet. Este estadío inicial sería seguido por la búsqueda de información sobre las particularidades del modelo que trata de solventar el problema expuesto en la introducción, Deeplabv3+.

El siguiente paso fue el de puesta en marcha de la red. Tras buscar distintas implementaciones en Keras, se escogió una de ellas, la del usuario MLearning, y se comprobó que no hubiera errores en la red, comparando con los artículos disponibles en ArXiv.

Tras esta sección, se escogieron las métricas y el optimizador a usar (Adam), y se comenzó a realizar las sucesivas pruebas sobre la primera base de datos, que se corresponde con el disco óptico. Estas pruebas buscaban encontrar los mejores hiperparámetros de la red, y poder obtener las mejores prestaciones sobre esta base de datos. Finalmente, se obtuvieron los resultados correspondientes a la excavación y el disco óptico utilizando las dos bases de datos disponibles.

3.3. *State of the Art*

3.3.1. Deeplabv3+

Deeplab es un modelo diseñado para la tarea de segmentación semántica de imágenes, que nació bajo la idea de solucionar los dos principales problemas de las Deep Convolutional Neural Networks (DCNNs) a la hora de segmentar imágenes: (1) la reducida resolución de características causada por las operaciones consecutivas de pooling y convoluciones con stride, y (2) la existencia de objetos a múltiples escalas en una misma imagen, que no pueden ser tratados de la misma manera. “La primera característica es la que le permite a las redes neuronales aprender de forma jerárquica. Sin embargo, esta invarianza a las transformaciones locales puede impedir tareas de predicción profunda, donde es necesario tener información detallada" (Chen, Zhu et al., 2018, pp. 1-2).

Para diseñar un modelo que evite estos dos problemas, en primera instancia se optó por usar una combinación de DCNNs y Campos Aleatorios Condicionales (CRFs), que es una herramienta muy usada en tareas de supresión de ruido en mapas de características. Sin embargo, en futuras versiones se decidió eliminar el postprocesado con CRFs para abordar el problema, y se decidió usar la ASPP y la «atrous convolution» (o convolución dilatada) que, junto con la estructura de codificador-decodificador, son los principales ingredientes de la última versión del modelo: Deeplabv3+.

Atrous convolution

La convolución dilatada es un algoritmo que permite realizar la operación de la convolución sin reducir el tamaño de los mapas de características, además de hacerlo de manera más "densa". Para ello, computa la siguiente operación (en 1D):

$$y[i] = \sum_{k=1}^{K} x[i + rk]\omega[k] \tag{1}$$

Donde el parámetro *r* se corresponde con el stride con el que se muestrea la señal de entrada, y es el que se debe cambiar si se desea modificar el campo de visión del algoritmo (FOV).

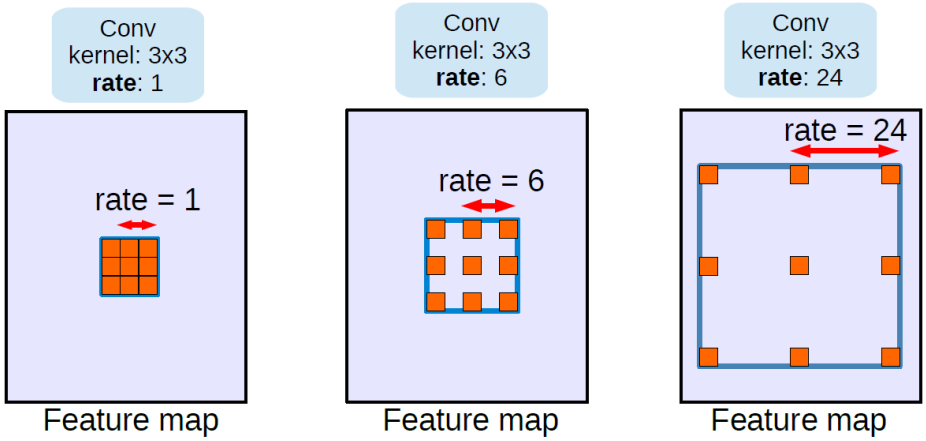


Figura 3.1: Ejemplo de convolución dilatada. Fuente: Chen, Papandreou et al., 2018, p. 1

"Esta operación es equivalente a convolucionar la entrada *x* con filtros interpolados, que se han obtenido al introducir $r - 1$ ceros entre dos valores consecutivos del filtro en cada dimensión espacial"(Chen, Papandreou et al., 2018, p. 3).

En primera instancia, este algoritmo se usó exclusivamente para controlar el FOV del algoritmo, pero pronto se adoptó una estrategia que permitiría tener distintos campos de visión superpuestos: el Atrous Spatial Piramid Pooling (ASPP)

ASPP

Esta técnica se basa en aplicar varias convoluciones dilatadas con distinto valor de *r* al último mapa de características de la red convolucional, de manera que se obtenga resultados con información a distinta escala, más o menos detallada. Los resultados de las sucesivas convoluciones serán después concatenados junto con un pooling de la imagen y una convolución 1x1, y nuevamente pasados por una convolución 1x1. Se puede ver un esquema que resume este funcionamiento en la figura 3.2.

Codificador-Decodificador

Por último, cabe destacar que la ASPP descrita se encuentra integrada como codificadora en una estructura de codificador-decodificador. Para el decodificador, la estructura es considerablemente simple: se realiza una interpolación bilineal de las características extraídas por el codificador, y se concatenan con las características extraídas a bajo nivel (que no es más que el mapa de características saliente de la red pasado por una convolución 1x1), para después ser convolucionados con un filtro de tamaño 3x3. Por último, se vuelve a ha-

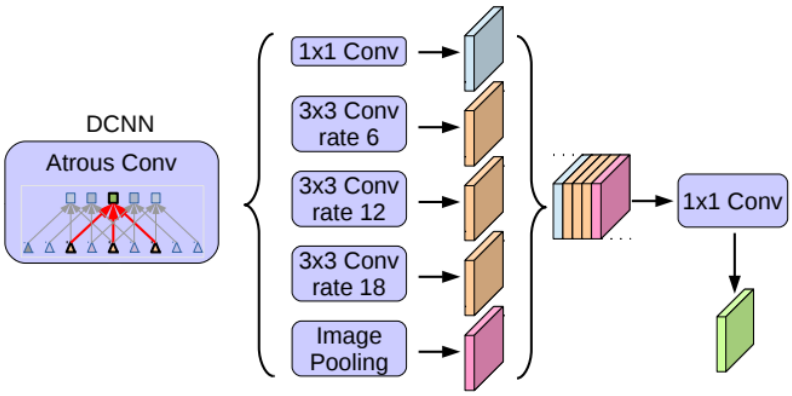


Figura 3.2: Funcionamiento de la ASPP. Fuente: Chen, Zhu et al., 2018, p. 4

cer una interpolación bilineal del resultado, y se proporciona la predicción. En la figura 3.3 se puede ver un esquema del funcionamiento del codificador-decodificador.

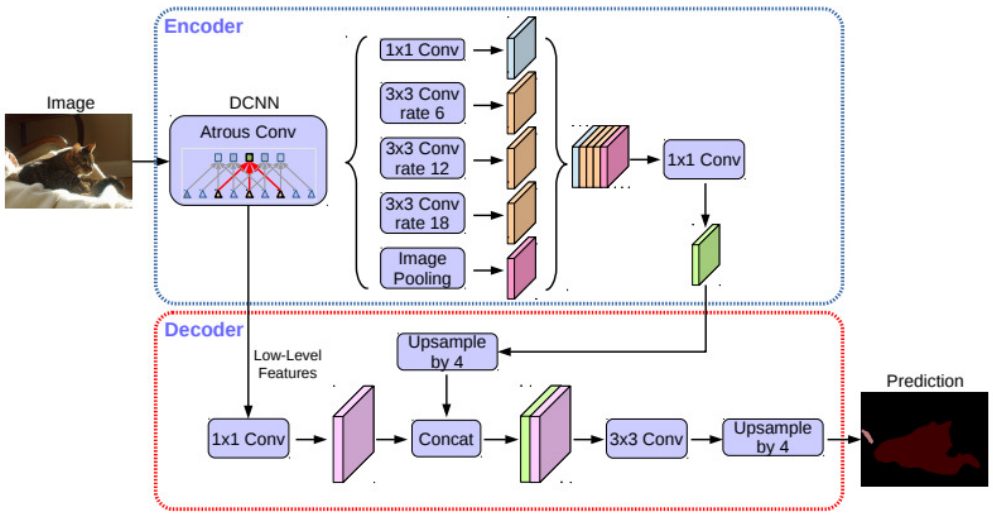


Figura 3.3: Esquema del codificador-decodificador. Fuente: Chen, Zhu et al., 2018, p. 4

Estas características son las que hacen de Deeplabv3+ un modelo tremendamente eficaz para segmentación de imágenes, que "ha conseguido establecer una nueva meta en los últimos avances en segmentación semántica" (Chen, Zhu et al., 2018, p. 14).

3.3.2. Xception

Xception es una red desarrollada por Google, que se basa en el uso la *Deepwise Separable Convolution* para disminuir el número de parámetros y coste computacional de una red para la clasificación de imágenes. Este tipo de convolución se basa en factorizar la operación de la convolución en otras dos variantes de la convolución (Tsang, 2018):

- La *Dephwise Convolution* es una convolución realizada en cada canal de manera independiente. Es decir, cada filtro existente recorre las posiciones válidas de un canal en

concreto

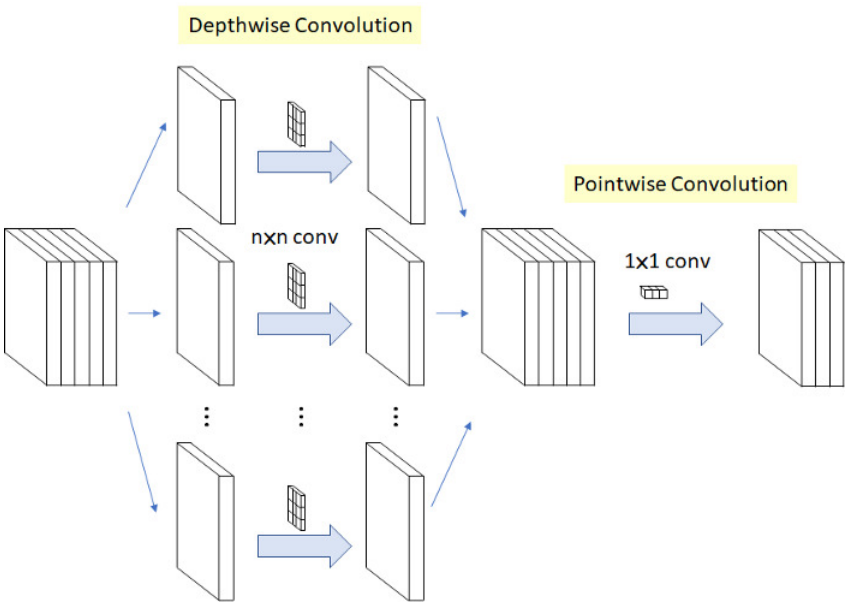
- La *Pointwise Convolution* es una convolución 1x1 realizada habitualmente para modificar el número de canales del mapa de características de la entrada.

Gracias a la unión de estos dos tipos de convoluciones, se consigue realizar la misma operación con un número menor de parámetros y con unas mejores prestaciones. Xception no fue la primera red en aplicar esta idea de separar la convolución, sin embargo, sí fue la primera red en factorizar la convolución de manera que ser realice primero la *Pointwise Convolution* y después la *Dephwise Convolution* (Tsang, 2018).

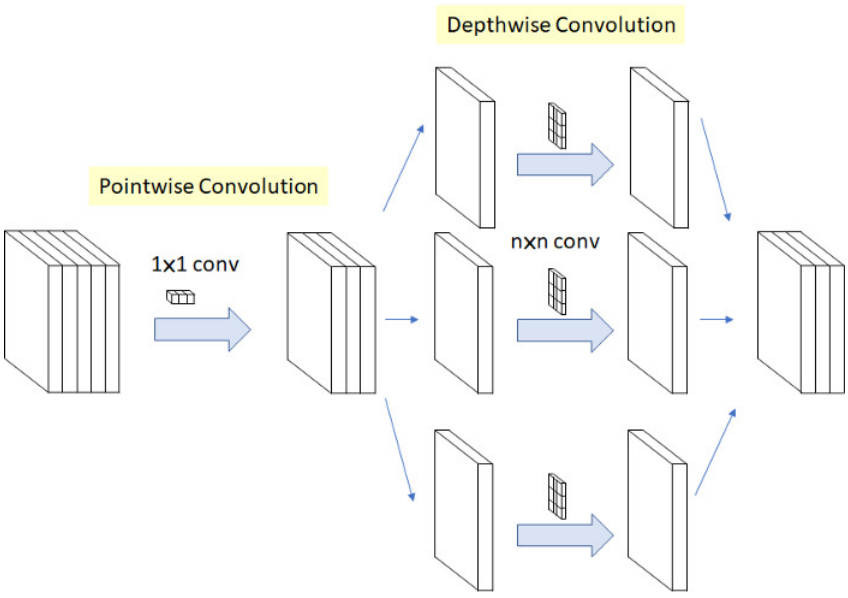
La red utilizada por Chen, Zhu et al. (2018) se basa en la unión de tres estructuras, que se esquematizan para favorecer su comprensión en la figura 3.5. Esta versión de la red realiza modificaciones sobre la red Aligned Xception, que es una reinterpretación de la red Xception original, y la reconvierte para la tarea de segmentación semántica. Todas estas modificaciones se centran en los siguientes factores.

- Construir un Xception más profundo, muy similar al de la Aligned Xception, aunque sin cambiar la estructura del flujo de entrada, para una mayor eficiencia computacional y de memoria.
- Reemplazar todas las operaciones de *max-pooling* por *Deepwise Separable Convolutions*, que permitan aplicar la *atrous separable convolution* para extraer mapas de características más densos, mejorando aún más las prestaciones de la red.
- Uso de más *batch normalization* en las capas intermedias, y de ReLU tras cada *Deepwise Convolution*.

Globalmente, en la red que muestra la figura 3.5, se diferencian tres grandes bloques: el flujo de entrada, el medio y el de salida. La concatenación de estos bloques consigue una red tremendamente eficaz en el campo de clasificación de imágenes, y que muy fácilmente puede reutilizarse para la segmentación semántica, añadiendo el modelo Deeplabv3+ tras la última capa de la red.



(a) Dephwise Separable Convolution en Inception (red predecesora de Xception)



(b) Dephwise Separable Convolution en Xception

Figura 3.4: Cambios en la Dephwise Separable Convolution efectuados por Xception. Fuente: Tsang, 2018

Optimizador y activación de la última capa

Para este último apartado, se debe tener en cuenta ante qué tipo de problema se encuentra. La segmentación semántica no es más que la clasificación de distintos píxeles de la imagen en clases, por lo que hay que distinguir únicamente entre el tipo de clase a clasificar. En el caso del presente trabajo se tiene más de una clase, y se usará la función de activación softmax.

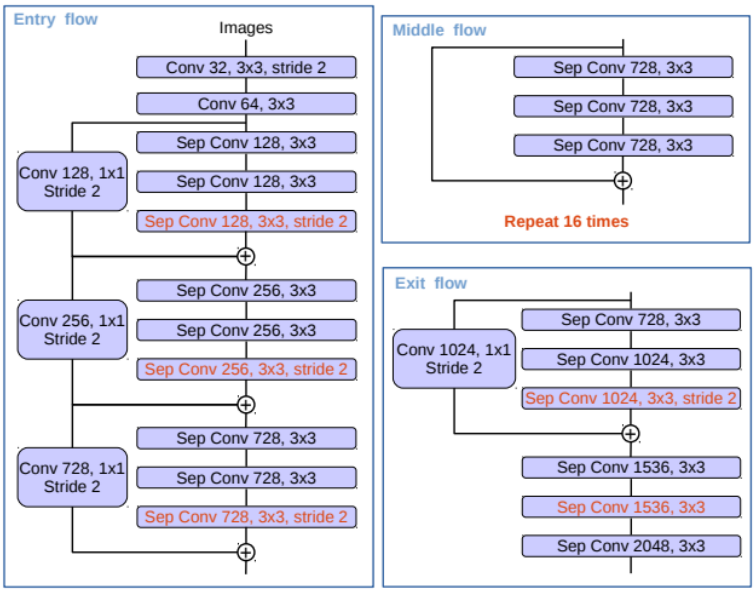


Figura 3.5: Estructura de la red Xception. Fuente: Chen, Zhu et al., 2018

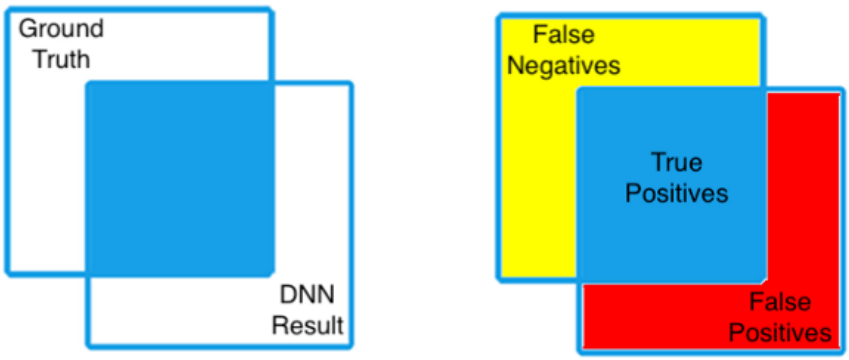
Por otro lado, el optimizador utilizado será Adam (*Adaptative Moment Stimation*), que combina las ventajas de dos optimizadores muy populares: AdaGrad (*Adaptative Gradient Algorithm*) y RMSProp. Adam implementa *momentum*, y decaimientos tanto para las tasas de aprendizaje como para el propio *momentum*. Es un optimizador que en los últimos años ha sido tremendamente usado y, que en palabras de los autores, "es un algoritmo simple y eficiente para optimización de funciones objetivo estocásticas" (Kingma y Ba, 2017, p. 1).

3.3.3. Métricas escogidas

Una vez explicada la red con la que se va a trabajar, se han de comentar las distintas métricas que se utilizarán para comprobar la bondad del modelo. Para facilitar su comprensión, se precisan algunas definiciones previas:

- Verdaderos positivos (*tp*): son aquellos píxeles clasificados correctamente por el modelo. Son aquellos píxeles de la imagen predicha que se corresponden con la etiqueta.
- Falsos positivos (*fp*): son los píxeles que no deberían haber sido clasificados por el modelo, ya que no son «nada» para esa clase, pero el modelo ha clasificado erróneamente como pertenecientes a la clase.
- Falsos negativos (*fn*): son los píxeles que deberían haber sido clasificados por el modelo, pero no ha conseguido segmentar.

A continuación, se definen las distintas métricas que se usará para comprobar la bondad del modelo en materia de segmentación semántica de imágenes. Es necesario destacar que todas ellas son binarias.



(a) Superposición de predicción y etiqueta

(b) Falsos negativos, falsos positivos y verdaderos positivos

Figura 3.6: Definiciones útiles en segmentación de imágenes. Adaptado de: Parsad, 2018

Precision y Recall

Estas métricas sirven para conocer si el clasificador está dividiendo de una manera correcta las muestras positivas y las negativas. Se definen desde cada clase:

- Precision: es la habilidad del clasificador de no marcar como positivo una muestra negativa. Se mide mediante la siguiente fórmula: $tp/(tp + fp)$ (Anónimo, s.f.-f).
- Recall: es la habilidad de encontrar todas las muestras positivas. Se mide mediante la siguiente fórmula: $tp/(tp + fn)$ (Anónimo, s.f.-g).

Son métricas que tomarán valores entre 0 y 1, y se considerarán mejores si están más próximas a 1. Interesa que estos valores estén lo más compensados posible, es decir, que tomen valores similares.

Curva ROC

La curva ROC (*Receiver Operating characteristic Curve*) es una gráfica que ilustra la habilidad de un clasificador binario cuando se varía el umbral de discriminación. Se crea a partir de la proporción de positivos reales (tp) respecto a la proporción de positivos falsos (fp), variando el umbral de discriminación (Anónimo, s.f.-h).

Lo que interesa a la hora de comprobar la bondad de la prueba es el Área Bajo la Curva ROC (AUC), que tomará valores situados entre 0.5 y 1. Cuanto mayor sea el valor tomado, mejor será el clasificador. En la gráfica 3.7 se muestra la tendencia de la curva en función del AUC. En caso de que el valor esté cercano a 0.5, se considerará que el clasificador está clasificando de manera aleatoria, por lo que no estará funcionando de manera correcta (Anónimo, s.f.-i).

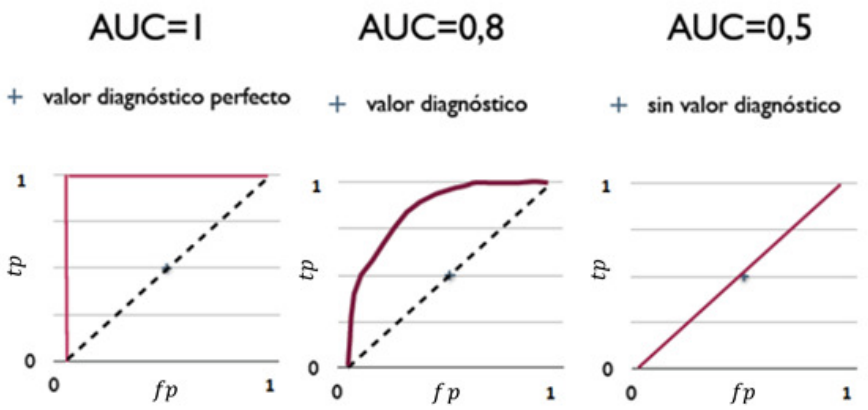


Figura 3.7: Conclusiones extraíbles de los valores de AUC. Fuente: Anónimo, s.f.-b

Índices de Jaccard y Dice

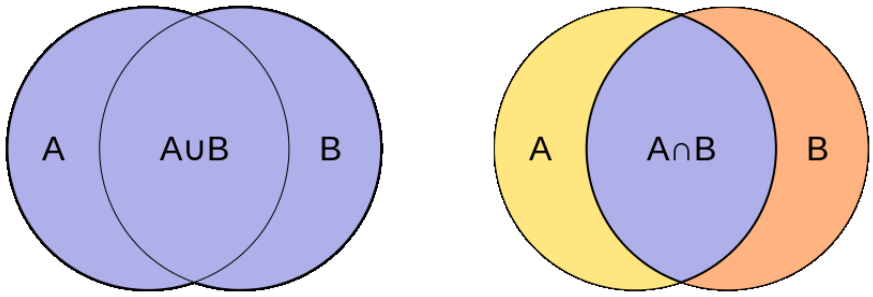
En este caso, estos dos estadísticos que se centran en medir la similitud entre de dos muestras, a través de la unión e intersección de conjuntos, que están representados gráficamente en al Figura 3.8. En primer lugar, el índice de Jaccard (o IOU), es el cociente entre la intersección y la unión de dos conjuntos de datos:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{tp}{tp + fn + fp} \tag{2}$$

Por otro lado, el índice de Dice, que se puede obtener a partir del índice de Jaccard, intenta realizar esta medida de una manera distinta (Parsad, 2018):

$$D(A,B) = \frac{2|A \cap B|}{|A| + |B|} = \frac{2tp}{(tp + fp) + (tp + fn)} = \frac{2J(A,B)}{J(A,B) + 1} \tag{3}$$

Estos dos índices tomarán valores entre 0 y 1, y serán mejores cuanto más próximos sean a 1, ya que tendrán más superficie en común las dos segmentaciones.



(a) Unión de conjuntos (b) Intersección de conjuntos

Figura 3.8: Intersección y unión de conjuntos. Fuente: Anónimo, s.f.-d

Distancia Hausdorff

La distancia Hausdorff mide cuán lejos se encuentran dos subconjuntos de un espacio métrico. Dos conjuntos están próximos si todo punto del primer conjunto está próximo a algún punto del segundo conjunto, y viceversa. La definición formal es la siguiente:

$$d_H(A,B) = max\left\{\sup_{a\in A}\inf_{b\in B} d(a,b), \sup_{b\in B}\inf_{a\in A} d(a,b)\right\} \tag{4}$$

En la figura 3.9 se muestra un ejemplo gráfico del cálculo de la distancia Hausdorff entre dos figuras. Evidentemente, a la hora de interpretar los resultados numéricos, interesará que esta medida tome los valores más bajos posibles, ya que se considerarán mejores predicciones. (Anónimo, s.f.-c)

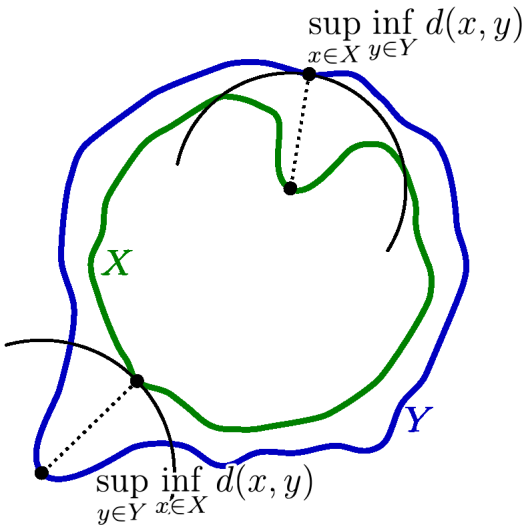


Figura 3.9: Ejemplo de medida de distancia Hausdorff. Fuente: Anónimo, s.f.-c

En el caso de este trabajo, se normalizará la distancia Hausdorff por las dimensiones de la imagen, y se trabajará con el valor en porcentaje, ya que se tratará de métricas de un valor muy pequeño. Por tanto, la fórmula será la siguiente:

$$d_H, norm(A,B) = \frac{d_H(A,B)}{\sqrt{XY}} \tag{5}$$

Donde X es la altura e Y es la anchura de la imagen.

4 Resultados

En este apartado se presenta el núcleo sobre el que se centra el proyecto, es decir, el entrenamiento del modelo con el propósito de segmentación semántica de retinografías. En este proceso, será necesario trabajar con las dos bases de datos disponibles, anteriormente mencionadas (RIMONEr3 y RIGA). Con este propósito, se buscará el valor más apropiado para cada hiperparámetro, y se obtendrán segmentaciones tanto del disco óptico como de la excavación, y de ambas segmentaciones en una misma imagen. Los hiperparámetros a modificar serán las épocas, el *batch size*, el *Output Stride* (OS) y el tamaño de la imagen de entrada, que modifica el tamaño de la primera capa, por lo que lo se considera un hiperparámetro.

En primer lugar, se partirá de un entrenamiento del disco óptico con la base de datos RIMONEr3, que servirá para tener una primera referencia de las prestaciones del modelo. Posteriormente, se realizarán sucesivos entrenamientos para encontrar los hiperparámetros que consigan obtener unos valores óptimos, mejorando tanto las métricas como los resultados visuales de la primera prueba. Una vez encontrados, se realizarán entrenamientos con las etiquetas correspondientes a la excavación, y con la excavación y el disco óptico en una misma imagen. Para finalizar, se hará esta última prueba con la base de datos RIGA.

4.1. Disco óptico

En primer lugar, se presentarán los resultados correspondientes al disco óptico, debido a que son los más fáciles de diferenciar por el ojo humano en una retinografía. Por ello, la hipótesis que se plantea es que será más simple para la red separar patrones que se pueden ver a simple vista. Para ello, la red será entrenada con las imágenes de la base de datos RIMONEr3, donde las etiquetas contendrán exclusivamente las segmentaciones del disco óptico.

En la primera prueba, los hiperparámetros tomarán unos valores predeterminados, y se realizarán sucesivas pruebas, intentando buscar un valor óptimo para cada uno de ellos. Finalmente, se volverá a entrenar la red con los hiperparámetros obtenidos, para comprobar que se obtiene un mejor resultado.

4.1.1. Primera prueba

En este primer caso, los hiperparámetros tomarán los siguientes valores:

- Número de épocas: 30
- *Batch size*: 4
- *Output Stride* (OS): 16
- Tamaño imagen: 100x100

Además, el conjunto de datos disponibles se ha dividido en un 10 % para test, un 15 % para validación y un 75 % para entrenamiento.

En la Tabla 4.1, se encuentran las métricas correspondientes al conjunto de test, que muestran un resultado prometedor. La red ha sido capaz de generar resultados buenos en algunos casos, como las imágenes 3, 6, 9 y 14, que poseen valores de Índice de Jaccard y Dice por encima de 0.940 y una distancia Hausdorff que del 2 %. Por otro lado, también se aprecian resultados bastante peores, como las imágenes 2, 8 y 16, siendo la peor de todas la última, debido a su alta distancia Hausdorff. En estos casos, cada una de las distintas métricas muestran los peores resultados de la Tabla.

Visualizando las medias, parece factible que la red pueda segmentar correctamente este tipo de imágenes, ya que se obtienen Índices de Jaccard y Dice considerablemente elevados, a la par que una distancia Hausdorff bastante buena. Cabe destacar que los valores de varianza obtenidos también son elevados, y deberían disminuir para considerar que el modelo es apropiado para este objetivo. Los resultados obtenidos se pueden contrastar uno a uno más adelante, en las Figuras 4.1 y 4.2, donde se encuentran las imágenes correspondientes a la Tabla 4.1, organizados de izquierda a derecha y de arriba abajo. Por tanto, para observar la imagen 5, habrá que dirigirse a la primera imagen de la segunda fila comenzando por la izquierda.

La Tabla 4.2 contiene los valores de *precision* y *recall*, que están compensados en caso del fondo, y bastante descompensados en el caso del disco óptico. Este será uno de los parámetros que más interese aumentar en futuros entrenamientos de la red.

Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
1	0.920	0.839	0.912	2.236
2	0.649	0.248	0.397	17.692
3	**0.981**	0.903	0.949	**2.000**
4	0.971	0.767	0.868	5.385
5	0.970	0.910	0.953	**2.000**
6	0.971	**0.920**	**0.958**	2.236
7	0.915	0.824	0.903	3.162
8	0.794	0.494	0.662	14.866
9	0.950	0.892	0.943	**2.000**
10	0.924	0.848	0.918	3.000
11	0.943	0.881	0.937	3.000
12	0.879	0.759	0.863	3.606
13	0.952	0.829	0.907	3.000
14	0.973	0.905	0.950	**2.000**
15	0.922	0.819	0.901	3.162
16	0.702	0.403	0.574	22.361
Media	0.899 $\pm$0,099	0.765 $\pm$0,195	0.850 $\pm$0,157	5.732 $\pm$6,242

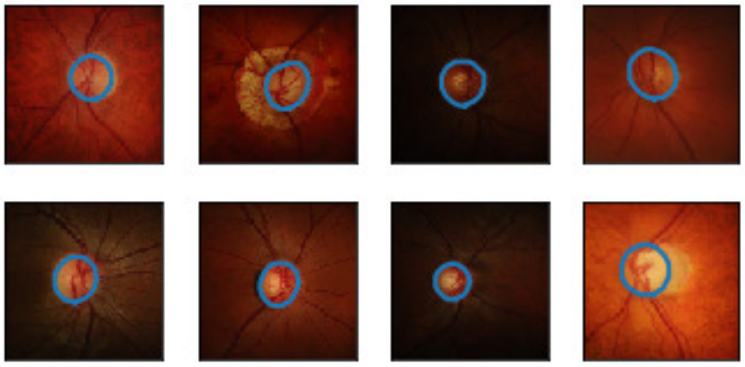
Tabla 4.1: Métricas de la primera prueba

Zona segmentada	*Precision*	*Recall*
Fondo	0.978	0.922
Disco óptico	0.995	0.725

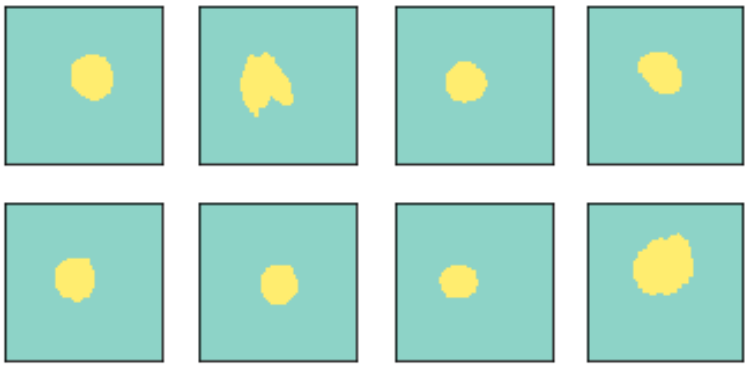
Tabla 4.2: *Precision* y *recall* de la primera prueba

Por otro lado, en la Figuras 4.1 y 4.2 se encuentran tanto las segmentaciones realizadas por el experto (Figuras 4.1a y 4.2a) como las realizadas por la red (Figuras 4.1b y 4.2b) y las predicciones de la red superpuestas a las segmentaciones del experto (Figuras 4.1c y 4.2c). Este último conjunto de imágenes resulta especialmente útil a la hora de asegurarse de que la red está segmentando correctamente. Las imágenes se organizarán de izquierda a derecha y de arriba abajo, numerándolas en función de su posición.

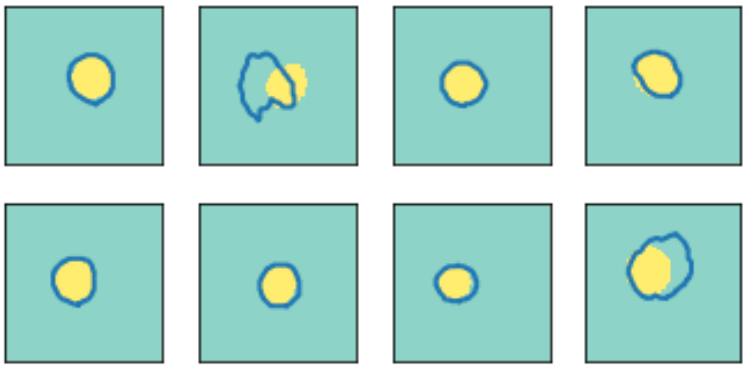
También es especialmente importante visualizar las gráficas correspondientes al entrenamiento, donde se puede ver el comportamiento de la red sobre los conjuntos de test y validación en función de las épocas. Estas gráficas se encuentran en la Figura 4.3, que muestra la evolución de la precisión (*accuracy*) en la Figura 4.3a, la del área bajo la curva (*Area Under the Curve*) en la Figura 4.3b y la de las pérdidas (*loss*) en la Figura 4.3c. En ellas se observa un claro ejemplo de *overffiting* a partir de la época 12, aproximadamente. Este problema debe solventarse, así como tratar de mejorar los resultados correspondientes a las imágenes.



(a) Segmentaciones del experto 1 (etiquetas)

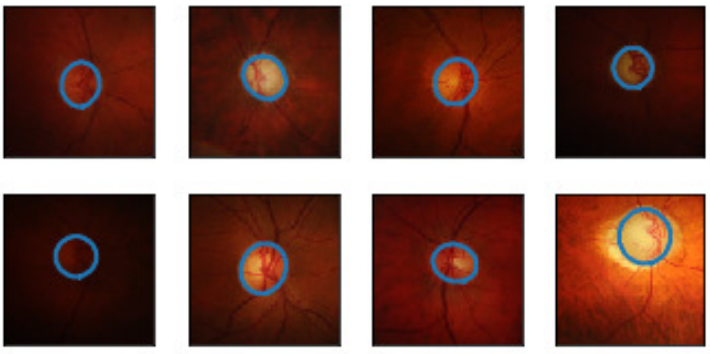


(b) Predicciones de la red

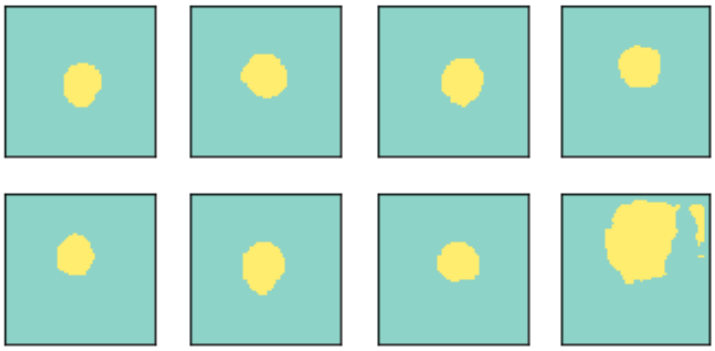


(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

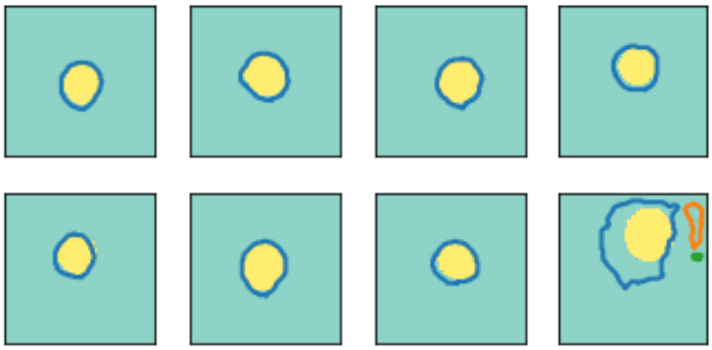
Figura 4.1: Resultados gráficos de la primera prueba (1)



(a) Segmentaciones del experto 1 (etiquetas)



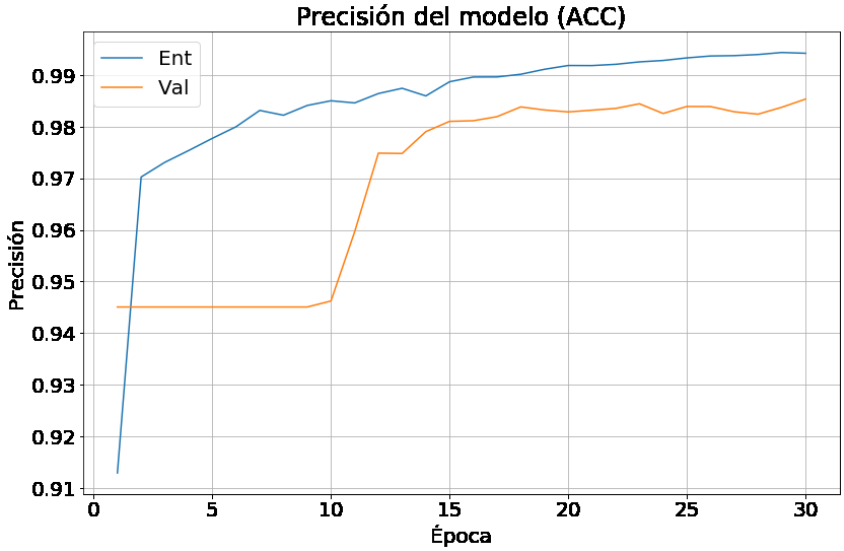
(b) Predicciones de la red



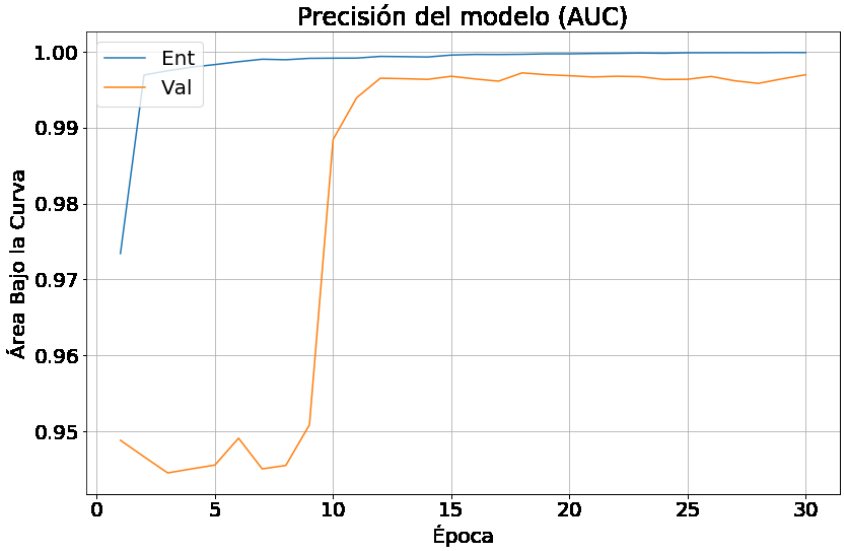
(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

Figura 4.2: Resultados gráficos de la primera prueba (1)

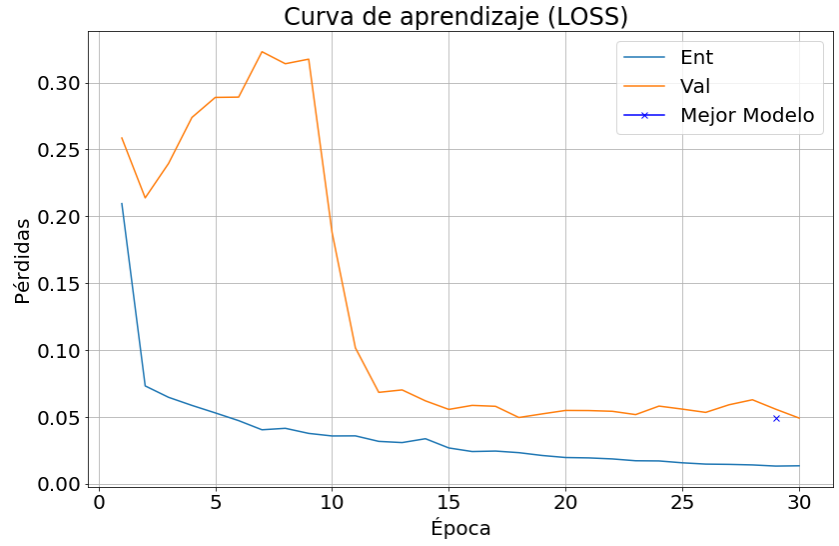
Por último, se ha marcado el mejor modelo en la gráfica de las pérdidas, es decir, la época en la que mejores resultados ha habido sobre el conjunto de validación. Es importante comprender que estas gráficas permitirán comprobar qué métrica es más sensible al *overfitting*, ya que tienen variaciones distintas. Este enfoque será explorado más detalladamente en el siguiente apartado.



(a) Precisión en función de las épocas



(b) Área bajo la curva en función de las épocas



(c) Pérdidas en función de las épocas

Figura 4.3: Gráficas de la primera prueba

4.1.2. Modificación de los hiperparámetros

Como se ha explicado previamente, es necesario mitigar el *overfitting*, y mejorar los resultados del conjunto de test tras el entrenamiento. Para ello, se cambiarán ciertos valores de los hiperparámetros, de manera que se consigan unas mejores prestaciones sobre el conjunto de test.

División de los conjuntos

En primer lugar, se explorarán dos configuraciones distintas de la división de los conjuntos de datos que se utilizarán para entrenar, validar y evaluar la red. Para ello, se cambiarán las proporciones de la división, manteniendo los hiperparámetros del primer entrenamiento. En este caso, los datos se presentarán únicamente en media y varianza, para tener en cuenta el resultado global del entrenamiento, y no los casos individuales.

En la Tabla 4.3, aparecen mejores resultados en el caso de tener un mayor conjunto de entrenamiento y un menor conjunto de test. Aunque el hecho de tener un conjunto de test mayor (y, por tanto, un conjunto de entrenamiento menor) empeore los resultados, se seleccionará esta opción. Esto es debido a que el conjunto de test será más apropiado para evaluar la bondad del modelo, sacrificando parte del conjunto de entrenamiento para esta causa.

Además, si se visualiza la Tabla 4.4, se aprecia una mayor compensación entre los parámetros de *precision* y *recall*, en caso de usar una separación de conjuntos 75-15-10. En los siguientes apartados, se procederá a la modificación de los distintoss hiperparámetros.

Conjuntos			AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
E	V	T				
75	15	10	0.914 $\pm$0,091	0.764 $\pm$0,200	0.848 $\pm$0,160	5.459 $\pm$5,757
70	15	15	0.894 $\pm$0,108	0.742 $\pm$0,220	0.826 $\pm$0,207	6.239 $\pm$7,536

Tabla 4.3: Métricas de la comparativa entre evaluaciones con distinta división de conjuntos

Conjuntos			*Precision*	*Recall*	Zona segmentada
E	V	T			
75	15	10	0.991	0.987	Fondo
			0.805	0.854	Disco óptico
70	15	15	0.991	0.987	Fondo
			0.799	0.885	Disco óptico

Tabla 4.4: *Precision* y *recall* de pruebas con distintas divisiones de conjuntos

Early-stopping

Ya se ha explicado en el apartado 2.4 que una herramienta muy útil a la hora de evitar el *overfitting* es el *early-stopping*. Esta técnica permite usar los pesos del modelo que proporcionan un mejor resultado desde el punto de vista del conjunto de validación. En este caso, se comprobará sobre qué métrica (ACC, AUC o LOSS) se debe monitorizar la parada del entrenamiento.

Para realizar esto, se entrenará la red durante un número de épocas elevado, tanto con *early-stopping* como sin él. Es necesario tener en cuenta que siempre se trabajará con el mismo conjunto de datos, para poder evaluar qué método está funcionando mejor bajo condiciones similares. En este caso, para intentar minimizar los efectos «aleatorios» del algoritmo de optimización, se han realizado tres simulaciones, que posteriormente se han promediado, obteniendo los siguientes resultados:

Early-Stopping	Épocas	AUC	índice Jaccard	Índice Dice	Distancia Hausdorff (%)
No	35	0.926 $\pm$0,121	0.786 $\pm$0,225	0.852 $\pm$0,223	5.496 $\pm$6,418
ACC	29	0.915 $\pm$0,095	0.751 $\pm$0,182	0.842 $\pm$0,145	5.887 $\pm$5,676
AUC	24	0.899 $\pm$0,107	0.709 $\pm$0,211	0.823 $\pm$0,182	6.450 $\pm$5,398
LOSS	25	0.918 $\pm$0,086	0.756 $\pm$0,162	0.850 $\pm$0,123	5.654 $\pm$4,811

Tabla 4.5: Comparativa de las métricas al usar *Early-Stopping*

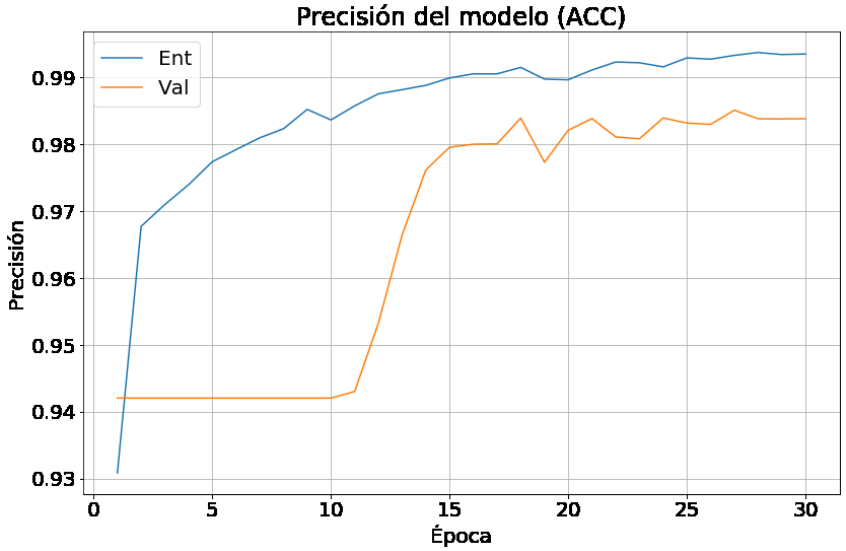
Early-Stopping	Épocas	*Precision*	*Recall*	Zona segmentada
No	35	0.990	0.992	Fondo
		0.866	0.835	Disco óptico
ACC	29	0.990	0.988	Fondo
		0.811	0.845	Disco óptico
AUC	24	0.989	0.986	Fondo
		0.785	0.819	Disco óptico
LOSS	25	0.990	0.989	Fondo
		0.831	0.846	Disco óptico

Tabla 4.6: Comparativa de *precision* y *recall* al utilizar *Early-Stopping*

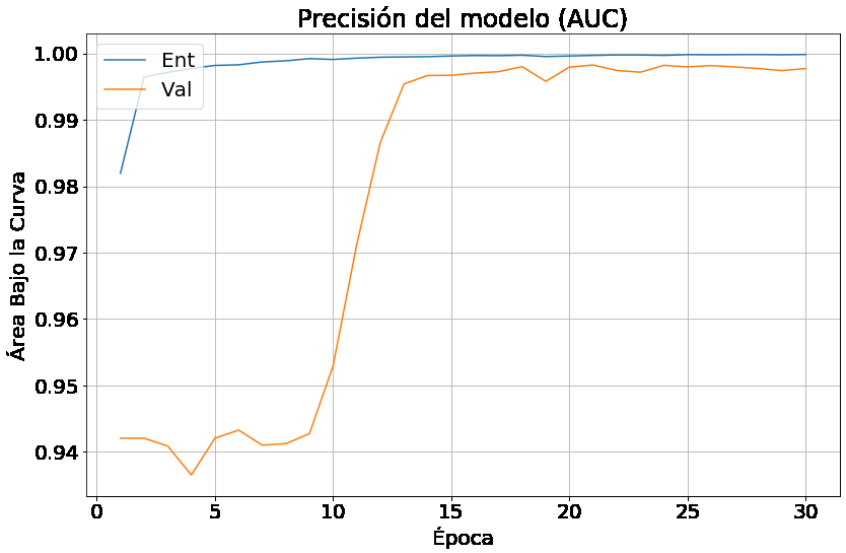
En las Tablas 4.5 y 4.6 se aprecia que el *early-stopping* funciona mejor en el caso de monitorizar las pérdidas (LOSS), ya que proporciona la pareja de valores *precision-recall* más compensada, aparte de proporcionar las mejores métricas (AUC e Índices de Jaccard y Dice más elevados, y distancia de Hausdorff más bajas) en media y varianza, visibles en las Tablas 4.5 y 4.6, respectivamente. Por otro lado, se ha ilustrado en la Figura 4.4 la evolución de las

distintas métricas, para comprender por qué está funcionando mejor una u otra para realizar la parada del entrenamiento.

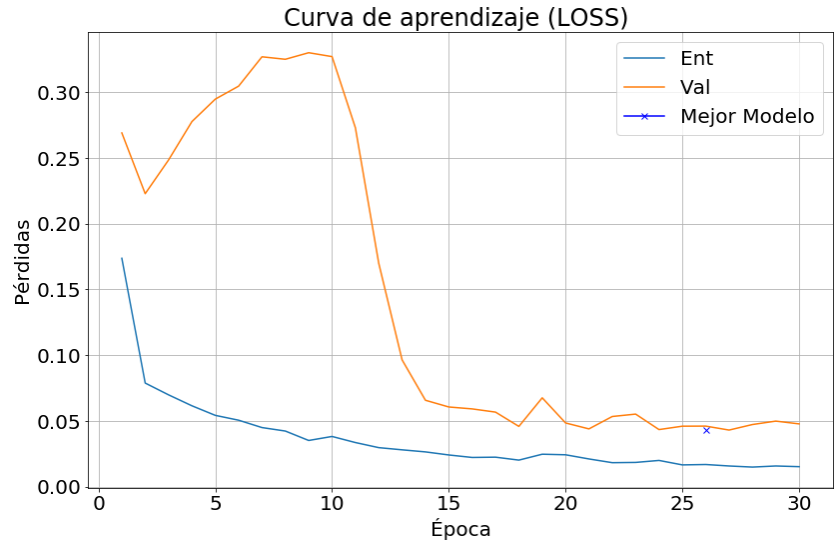
Se puede comprobar que las pérdidas (LOSS) se sitúan entre la regularidad de la AUC y la irregularidad de la precisión (ACC). Por tanto, parece razonable monitorizar esta primera métrica ya que, una vez alcanzado el *overfitting*, resultará más fácil quedarse con el mejor modelo al tener unas variaciones más suaves.



(a) Precisión en función de las épocas



(b) Área bajo la curva en función de las épocas



(c) Pérdidas en función de las épocas

Figura 4.4: Comparativa de la evolución de las distintas métricas

Batch size

El *batch size* es un hiperparámetro difícil de optimizar debido a que el conjunto de datos con el que se entrena el modelo es muy pequeño. Esto provocará que el modelo se ajuste muy rápidamente a este y, como consecuencia de las variaciones propias del algoritmo de optimización, se deben realizar varias ejecuciones y promediarlas, para poder tomar una decisión correcta. En este caso, se realizarán cinco simulaciones, consideradas suficientes como para poder observar una tendencia clara en los datos.

Las Tablas 4.7 y 4.8 permiten comprobar que el impacto del *batch size* en las métricas finales no es muy importante, ya que no hay mucha diferencia entre tomar un valor u otro. Los resultados muestran una mejoría en función del aumento del valor de *batch size*, por lo que se podría pensar que un valor elevado en este hiperparámetro provocaría un resultado mejor. Esto no es así, ya que se entrenó la red con un valor de *batch size* igual a 5, que desencadenó un *overfitting* de memoria. Es por esto que el modelo trabajará con un valor de *batch size* igual a 4, que fue el máximo que se pudo obtener bajo las condiciones de simulación.

Batch size	Épocas	AUC	índice Jaccard	Índice Dice	Distancia Hausdorff (%)
2	29	0.906 $\pm$0,116	0.740 $\pm$0,228	0.830 $\pm$0,219	6.944 $\pm$9,163
3	28	0.910 $\pm$0,116	0.759 $\pm$0,233	0.834 $\pm$0,221	6.274 $\pm$9,446
4	29	0.914 $\pm$0,092	0.753 $\pm$0,228	0.850 $\pm$0,219	6.132 $\pm$8,629

Tabla 4.7: Evaluación de las métricas con distinto *batch size*

Batch size	Épocas	*Recall*	*Precision*	Zona segmentada
2	24	0.991	0.988	Fondo
		0.814	0.845	Disco óptico
3	30	0.990	0.989	Fondo
		0.821	0.841	Disco óptico
4	30	0.989	0.990	Fondo
		0.836	0.819	Disco óptico

Tabla 4.8: Comparativa de *precision* y *recall* de los resultados usando distinto *batch size*

Output Stride (OS)

Como se definió anteriormente, el *Output Stride* es la relación entre la imagen de entrada y la de salida de la red. En el artículo correspondiente al desarrollo de Deeplabv3+ (Chen, Zhu et al., 2018), los autores recomiendan utilizar los valores de OS igual 8 o 16 a la hora de entrenar la red con el objetivo de segmentar imágenes. Por tanto, serán estos valores los usados para comprobar cuál es el mejor valor para entrenar la red, siempre desde el punto de vista del conjunto de test.

En el caso de este trabajo, se ha experimentado una mejora muy leve en caso de utilizar un OS menor. Esto queda reflejado en las Tablas 4.9 y 4.10, en las que no se aprecia un gran cambio entre estas dos posibilidades. A pesar de que conlleva un coste computacional levemente superior, al primar la bondad de los resultados frente al coste, se seleccionará un valor de OS igual a 8.

OS	Épocas	AUC	índice Jaccard	Índice Dice	Distancia Hausdorff (%)
16	30	0.908 ±0,097	0.757 ±0,235	0.832 ±0,227	6.243 ±9,449
8	29	0.906 ±0,094	0.757 ±0,229	0.833 ±0,222	6.200 ±8,899

Tabla 4.9: Comparativa de las métricas al usar distinto Output Stride

OS	Épocas	*Precision*	*Recall*	Zona segmentada
16	30	0.990	0.989	Fondo
		0.828	0.840	Disco óptico
8	29	0.990	0.990	Fondo
		0.836	0.836	Disco óptico

Tabla 4.10: Evaluación de *precision* y *recall* en función del Output Stride

Tamaño imagen entrada

En este trabajo se considera el tamaño de la imagen de entrada un hiperparámetro debido a que modifica de la primera capa de la red. Por ello, para comprobar cuál es el mejor valor para este hiperparámetro, se utilizarán tamaños mayores de imagen que el inicialmente usado (100x100) en todas las pruebas. Esto se debe a que valores menores perjudicarían el resultado final del modelo por la menor resolución de imagen y, por ende, habría una cierta pérdida de información, que puede resultar valiosa para el aprendizaje del modelo. Se realizarán numerosas pruebas, de manera que se pueda apreciar un cierto cambio en función del tamaño de imagen seleccionado.

Como se intuía previamente que los resultados mejorarán conforme aumente el tamaño de la imagen. Además, esta mejoría se da de una manera aproximadamente lineal en el caso de todas las métricas, como se puede observar en la Tabla 4.11. Igual que antes, se tendrá en cuenta el tiempo de procesado de las imágenes, buscando una compensación coste computacional-rendimiento lo más óptima posible.

Tamaño	Épocas	AUC	índice Jaccard	Índice Dice	Distancia Hausdorff (%)
100x100	30	0.940 ±0,074	0.790 ±0,152	0.871 ±0,128	5.040 ±6,705
125x125	29	0.966 ±0,047	0.813 ±0,131	0.889 ±0,097	4.791 ±6,069
150x150	29	0.961 ±0,027	0.832 ±0,107	0.904 ±0,074	3.766 ±4,506
180x180	27	0.958 ±0,039	0.845 ±0,096	0.912 ±0,067	3.694 ±4,952
200x200	28	0.945 ±0,059	0.832 ±0,155	0.897 ±0,126	3.207 ±3,292
220x220	30	0.963 ±0,035	0.850 ±0,092	0.916 ±0,059	3.072 ±2,332

Tabla 4.11: Métricas en función del tamaño de imagen de entrada

Por otro lado, la Tabla 4.12 resulta relevante destacar que las parejas de valores de *precision* y *recall* se mantienen siempre alrededor de los mismos valores, mejorando y empeorando en los distintos entrenamientos. De esta manera, de entre todas las posibilidades, se ha seleccionado el valor de 200x200 para la imagen debido a que, en este caso, el aumento de coste computacional era mayor que en el caso del *Output Stride*.

Tamaño	Épocas	*Precision*	*Recall*	Zona segmentada
100x100	30	0.993	0.995	Fondo
		0.908	0.882	Disco óptico
125x125	29	0.991	0.997	Fondo
		0.937	0.847	Disco óptico
150x150	29	0.993	0.996	Fondo
		0.930	0.884	Disco óptico
180x180	27	0.995	0.996	Fondo
		0.928	0.907	Disco óptico
200x200	28	0.994	0.994	Fondo
		0.894	0.895	Disco óptico
220x220	30	0.994	0.996	Fondo
		0.929	0.897	Disco óptico

Tabla 4.12: *Precision* y *recall* en función del tamaño de imagen de entrada

4.1.3. Prueba final

Ahora que se han obtenido los valores óptimos para cada hiperparámetro, se puede realizar una simulación con todos ellos, y así comprobar que la mejora del modelo es efectiva.

Los resultados referidos a las imágenes, que se encuentran en las Tablas 4.13 y 4.14, parecen ser mejores que en el primer caso, mejorando tanto en media como en varianza en la mayoría de métricas. En concreto, en la Tabla 4.13 destaca la imagen 18, que posee un valor de distancia Hausdorff igual a 1.118 %. De igual manera, tiene valores elevados para el resto de métricas, destacando un valor de 0.972 en el índice de Dice para esta misma imagen. Respecto al resto, encontramos tres que tienen un índice de Dice por encima de 0.96, que son las imágenes 7, 12 y 20, con distancias Hausdorff que rondan el 2 %.

Por otro lado, es posible encontrar los peores valores de esta distancia para las imágenes 5, 8, 14 y 19, cuyos valores superan el 13 %. Si se visualizara el resto de resultados, se podría pensar que las segmentaciones de las retinografías son bastante buenas. Este hecho queda confirmado si se observan las medias y varianzas, y también puede ser comprobado de manera gráfica en las Figuras 4.5 y 4.6. En estas, se podrá encontrar la explicación a no haber obtenido una menor distancia Hausdorff. La imagen 8 es una segmentación bastante mala, que contiene zonas no unidas entre sí.

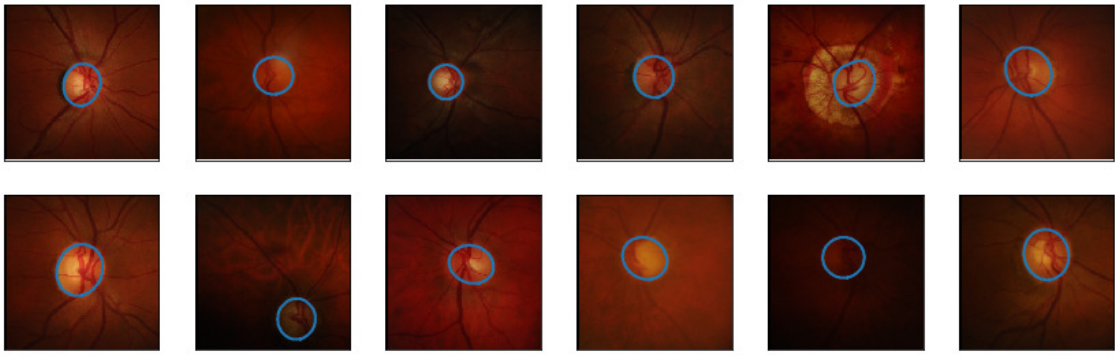
Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
1	0.970	0.904	0.950	1.803
2	0.939	0.835	0.910	3.041
3	0.955	0.848	0.918	1.803
4	0.964	0.900	0.948	1.581
5	0.913	0.645	0.784	19.416
6	**0.994**	0.832	0.908	2.915
7	**0.994**	0.927	0.962	1.803
8	0.475	0.000	0.000	37.047
9	0.954	0.851	0.919	3.202
10	0.954	0.840	0.913	3.041
11	0.988	0.565	0.722	5.852
12	0.972	0.928	0.963	2.121
13	0.988	0.851	0.920	2.500
14	0.818	0.543	0.704	17.500
15	0.985	0.922	0.959	1.803
16	0.941	0.861	0.925	3.536
17	0.963	0.912	0.954	1.500
18	0.989	**0.946**	**0.972**	**1.118**
19	0.777	0.489	0.657	13.946
20	0.968	0.925	0.961	1.500
21	0.975	0.889	0.941	1.803
22	0.936	0.869	0.930	2.693
23	0.917	0.833	0.909	1.693
24	0.943	0.876	0.934	1.803
Media	0.928 $\pm$0,107	0.791 $\pm$0,207	0.861 $\pm$0,199	5.668 $\pm$8,207

Tabla 4.13: Métricas asociadas al disco óptico con los hiperparámetros óptimos

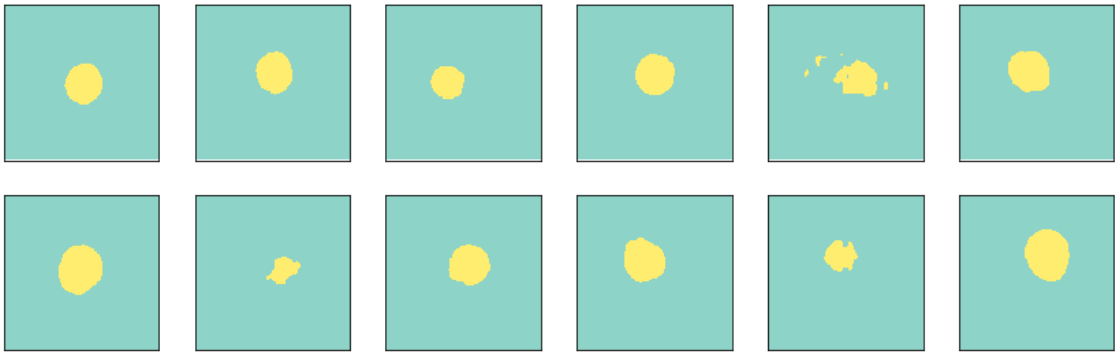
Zona segmentada	*Precision*	*Recall*
Fondo	0.992	0.992
Disco óptico	0.866	0.868

Tabla 4.14: *Precision* y *recall* del disco óptico con los hiperparámetros óptimos

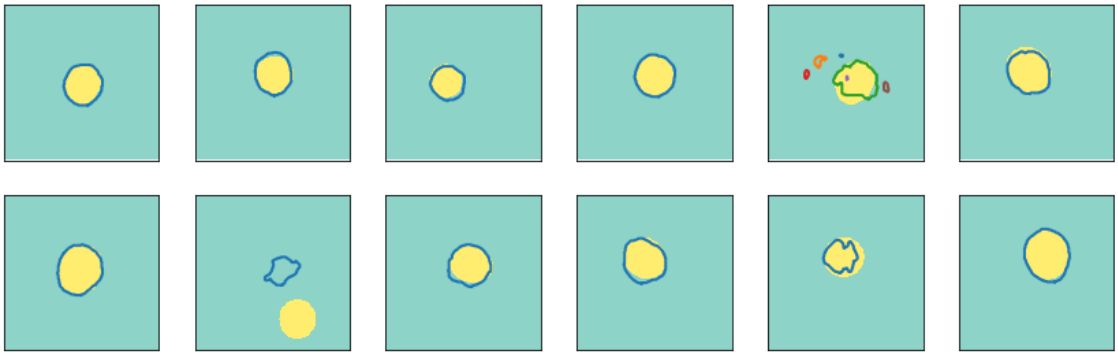
Como en casos anteriores, los datos se han organizado de la siguiente manera: en primer caso, las imágenes de entrada de la red con la segmentación del experto superpuesta (Figura 4.6a), seguidas de las predicciones realizadas por la red (Figura 4.6b) y por las segmentaciones del experto superpuestas a las predicciones de la red (Figura 4.6c).



(a) Segmentaciones del experto 1 (etiquetas)

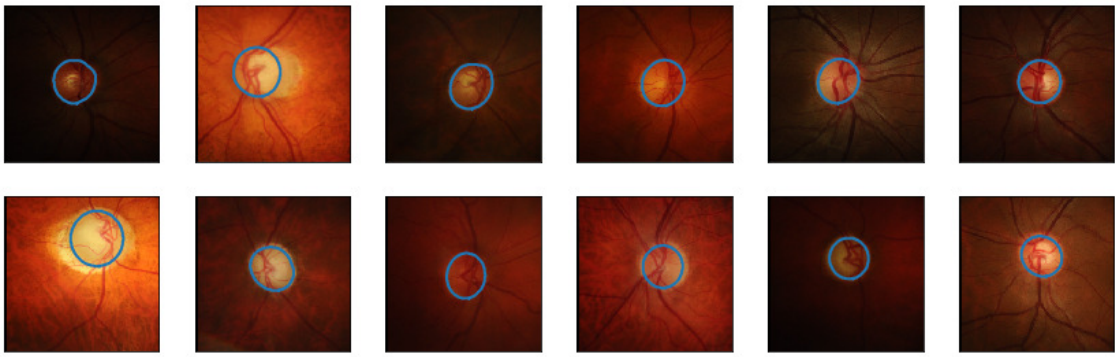


(b) Predicciones de la red

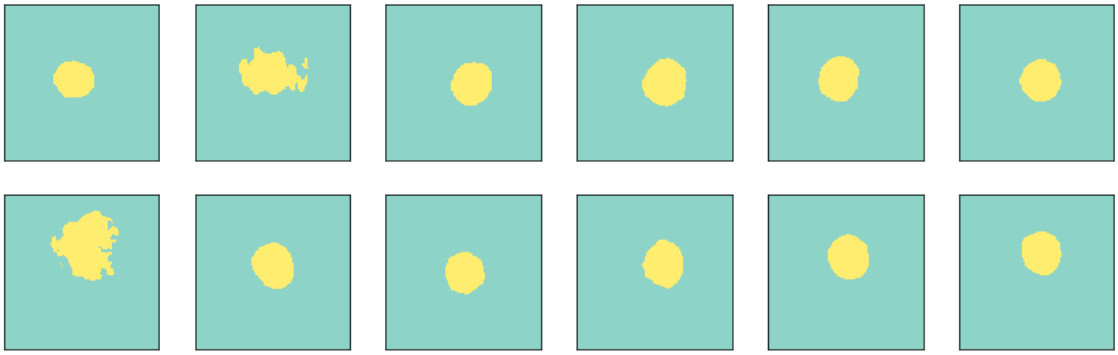


(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

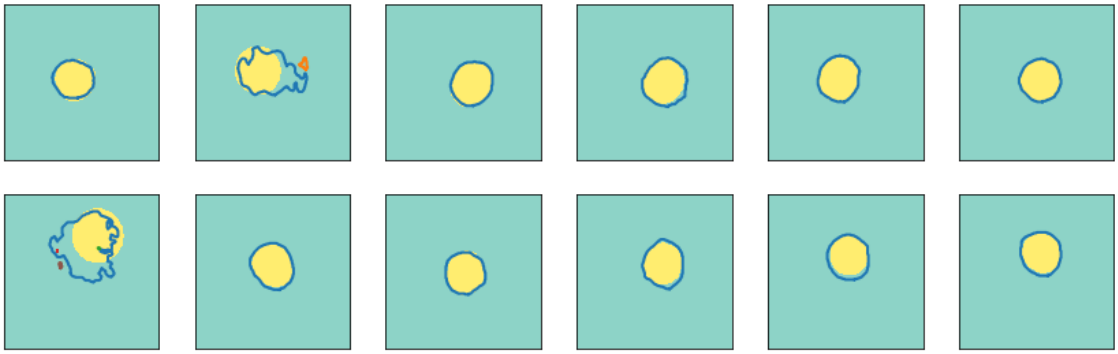
Figura 4.5: Resultados del disco óptico con los hiperparámetros óptimos (1)



(a) Segmentaciones del experto 1 (etiquetas)



(b) Predicciones de la red



(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

Figura 4.6: Resultados del disco óptico con los hiperparámetros óptimos (2)

4.1.4. Comparativa del cambio entre la primera y la última simulación

En el apartado anterior se pudo concluir que se producía una mejora notable en los resultados. Sin embargo, debido a que la red se entrenaba, validaba y evaluaba con distintos conjuntos de datos, no fue posible medir esta mejoría. En este apartado, se pretende comprobar en qué medida se ha mejorado el desempeño de la red sobre el conjunto de test, utilizando distintos hiperparámetros, y unos mismos conjuntos de entrenamiento, validación y test.

Es por ello que se presentan, en primer lugar, los resultados correspondientes a los hiperparámetros que se usaron en la primera simulación (*Output Stride* = 16, Tamaño de imagen = 100 x 100 y sin mecanismo de parada) y, posteriormente, los correspondientes a los que se utilizaron en última instancia (*Output Stride* = 8, Tamaño de imagen = 200 x 200 y las pérdidas como mecanismo de parada). El *batch size* será un parámetro fijo, con un valor igual a 4.

En caso de comparar los resultados mostrados en las Tablas 4.15 y 4.17, se puede hacer tanto imagen a imagen como en media. Los resultados mejoran en todas y cada una de las imágenes, excepto la número 5, que empeora en distancia Hausdorff (aunque mejora levemente en el resto). Siguiendo esta línea, en la imagen 23 se aprecia un empeoramiento de los índices de Dice y Jaccard, aunque se mejora la distancia Hausdorff. Por último, cabe destacar que la media del AUC empeora levemente, aunque esto seguramente sea debido al mayor número de píxeles en este segundo caso.

Por otro lado, en cuanto a la *precision* y el *recall*, que se pueden ver en las Tablas 4.16 y 4.18, mejoran notablemente en el segundo caso, ya que hay una pareja de valores mucho más compensada en ambos casos (fondo y disco óptico).

- Entrenamiento con los hiperparámetros iniciales:

Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
1	0.987	0.709	0.830	5.000
2	0.984	0.725	0.840	3.606
3	0.984	0.756	0.861	4.000
4	0.969	0.822	0.902	4.123
5	0.986	0.626	0.770	5.000
6	0.980	0.423	0.594	10.440
7	0.987	0.667	0.800	5.000
8	0.476	0.000	0.000	39.217
9	0.991	0.652	0.789	5.385
10	0.968	0.611	0.759	9.434
11	0.767	0.533	0.695	9.434
12	**0.992**	0.798	0.888	5.000
13	0.942	0.798	0.888	5.000
14	0.772	0.198	0.330	15.524
15	0.986	0.769	0.870	**3.000**
16	0.984	0.642	0.782	5.000
17	0.942	0.663	0.797	5.831
18	0.991	0.685	0.813	5.000
19	0.594	0.087	0.159	23.087
20	**0.992**	0.796	0.887	3.606
21	0.991	0.696	0.821	5.000
22	0.989	0.799	0.888	4.000
23	0.938	**0.865**	**0.928**	4.243
24	**0.992**	0.695	0.820	6.708
Media	0.922 $\pm$0,132	0.626 $\pm$0,224	0.738 $\pm$0,233	7.985 $\pm$7,847

Tabla 4.15: Métricas obtenidas utilizando los hiperparámetros iniciales

Zona segmentada	*Precision*	*Recall*
Fondo	0.979	0.994
Disco óptico	0.868	0.644

Tabla 4.16: *Precision* y *recall* obtenidas con los hiperparámetros iniciales

- Entrenamiento con los hiperparámetros finales

Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
1	0.963	0.902	0.949	2.236
2	0.929	0.840	0.913	2.916
3	0.919	0.825	0.904	3.000
4	0.974	0.941	0.970	1.414
5	0.976	0.638	0.779	7.500
6	0.968	0.799	0.888	4.123
7	**0.988**	**0.962**	**0.981**	**1.118**
8	0.475	0.000	0.000	36.912
9	0.949	0.865	0.928	3.000
10	0.937	0.827	0.905	2.500
11	0.937	0.858	0.924	2.500
12	0.978	0.939	0.969	1.500
13	0.973	0.920	0.959	1.414
14	0.807	0.520	0.684	11.000
15	0.974	0.930	0.964	1.414
16	0.909	0.819	0.900	4.743
17	0.951	0.901	0.948	2.500
18	0.965	0.918	0.957	1.581
19	0.639	0.185	0.312	20.056
20	0.941	0.882	0.937	1.414
21	0.962	0.884	0.939	2.616
22	0.887	0.774	0.873	2.693
23	0.918	0.834	0.909	2.500
24	0.929	0.854	0.921	2.500
Media	0.910 $\pm$0,115	0.784 $\pm$0,231	0.850 $\pm$0,223	5.108 $\pm$7,755

Tabla 4.17: Métricas obtenidas utilizando los hiperparámetros finales

Zona segmentada	*Precision*	*Recall*
Fondo	0.992	0.989
Disco óptico	0.831	0.867

Tabla 4.18: *Precision* y *recall* obtenidos con los hiperparámetros finales

4.2. Excavación

En este caso, se entrenará la red con las imágenes correspondientes a la excavación, manteniendo los hiperparámetros del apartado anterior. Esto se debe a que se trata de un problema de segmentación muy similar al anterior. En él se trabajará con las mismas imágenes, a excepción de la etiqueta, que se corresponde con la excavación. Además, este proceso de búsqueda de hiperparámetros implicaría emplear unos recursos temporales que se prefieren emplear de otra manera.

La zona correspondiente a la excavación no será tan fácil de distinguir, como sí lo era el disco óptico. Esto queda remarcado en los resultados numéricos de la Tabla 4.19, que son bastante peores (en media) que los correspondientes al disco óptico, aunque no son del todo malos. En esta misma Tabla se pueden ver dos imágenes en las que la red no ha podido realizar ninguna predicción. Estas son la 2 y la 11, y han sido marcadas con una línea horizontal debido a que no han generado ningún valor.

Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
1	0.956	0.781	0.877	1.500
2	—	—	—	—
3	0.887	0.651	0.788	1.803
4	0.917	0.732	0.845	1.500
5	0.780	0.286	0.445	20.353
6	0.828	0.645	0.784	2.915
7	0.949	0.805	0.892	3.500
8	0.983	0.025	0.049	14.431
9	0.971	0.785	0.880	1.118
10	0.955	0.858	0.924	2.062
11	—	—	—	—
12	0.890	0.775	0.873	3.000
13	0.837	0.672	0.804	2.500
14	0.692	0.364	0.533	17.000
15	0.802	0.575	0.730	5.500
16	0.707	0.414	0.586	6.185
17	0.732	0.443	0.614	3.202
18	0.968	0.736	0.848	3.000
19	0.713	0.411	0.582	27.987
20	0.666	0.319	0.484	8.485
21	0.854	0.128	0.227	10.404
22	0.660	0.320	0.484	4.610
23	0.982	0.654	0.791	3.536
24	0.989	0.859	0.924	1.500
Media	0.812 $\pm$0,167	0.510 $\pm$0,274	0.624 $\pm$0,290	6.087 $\pm$6,948

Tabla 4.19: Métricas de la excavación

También es posible comprobar que las mejores imágenes poseen una distancia Hausdorff por debajo del 2 %, como las imágenes 1, 4, 9 y 24. Además, estas tienen valores de

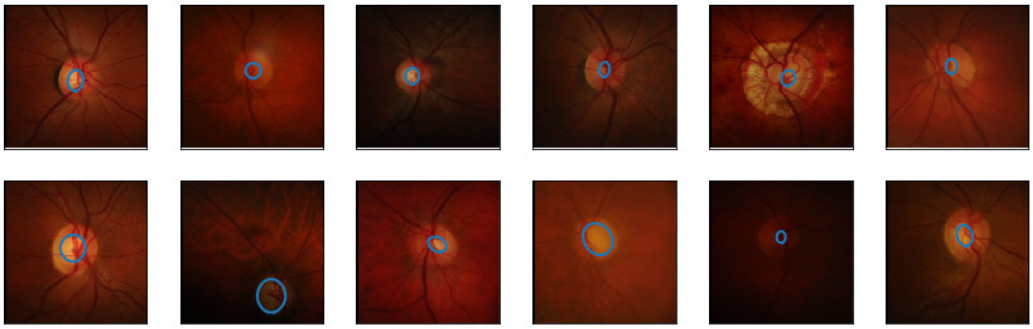
índice de Dice situados por encima de 0.840, y de Jaccard alrededor de 0.730. Por otro lado, las imágenes 5, 8, 14, 19 y 21 tienen valores de distancia Haussdorf por encima del 10 %. Respecto a los índices de Dice y Jaccard, estas imágenes también presentan valores muy bajos, que se sitúan por debajo de 0.590 y, en el caso de la imagen 8, es prácticamente cero.

Por otro lado, la Tabla 4.20 muestra unos resultados bastante descompensados a la hora de identificar la excavación, que no llega a ser del todo bajo en el caso del *recall*. En este caso no se puede alterar ningún hiperparámetro para mejorar la identificación de la excavación, por lo que se tomará un enfoque distinto en el siguiente apartado.

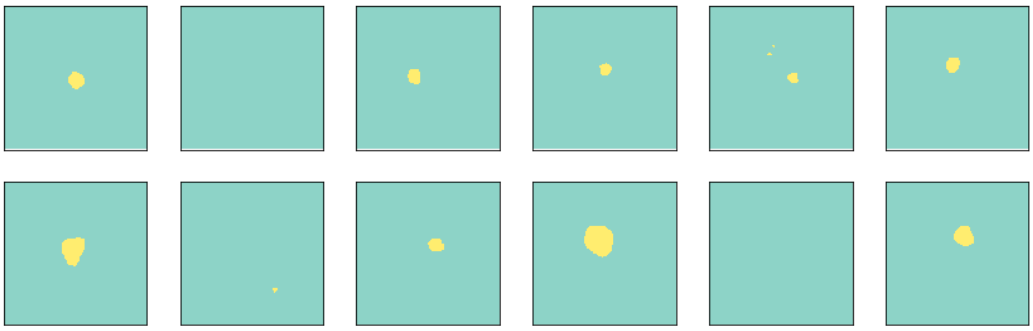
Zona segmentada	*Precision*	*Recall*
Fondo	0.992	0.996
Excavación	0.597	0.736

Tabla 4.20: *Precision* y *recall* de la excavación

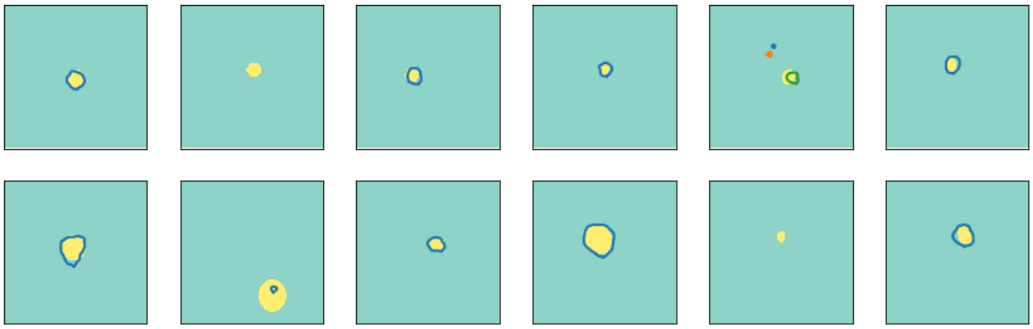
Por último, en las Figuras 4.7 y 4.8, se muestran las imágenes correspondientes a las Tablas 4.19 y 4.20. En las Figuras 4.7c y 4.8c, se aprecia que, en la mayoría de casos, la red sí es capaz de intuir en qué zona se encuentra la excavación, siendo capaz de hacer algunas segmentaciones prácticamente idénticas. Sin embargo, en los casos ya nombrados (las imágenes 2 y 11), no es capaz de predecir nada en la imagen.



(a) Segmentaciones del experto 1 (etiquetas)

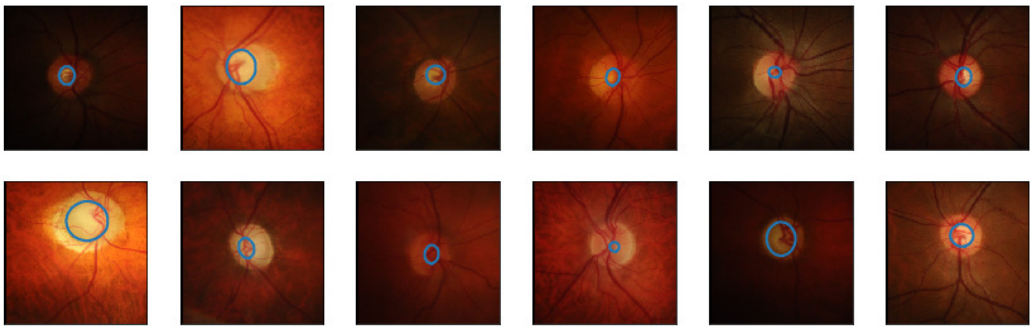


(b) Predicciones de la red

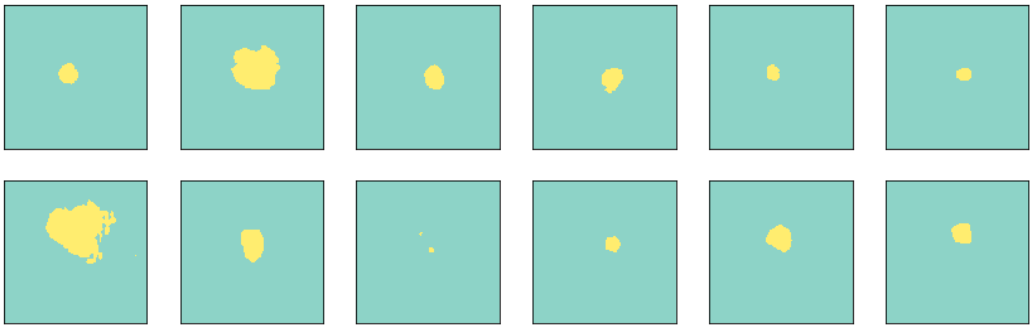


(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

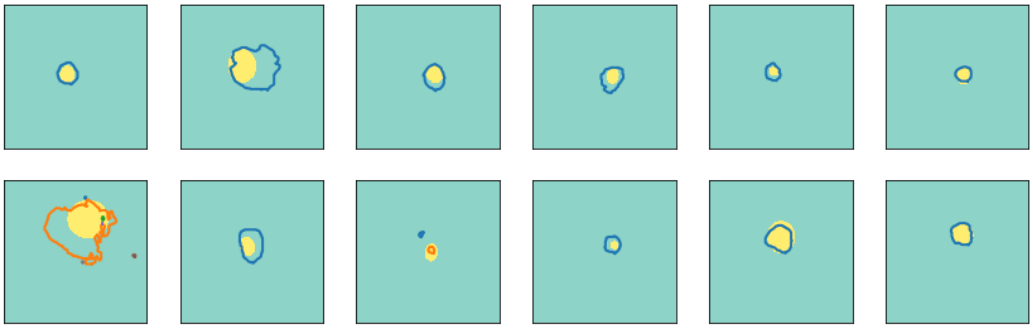
Figura 4.7: Comparativa entre las segmentaciones de la excavación (1)



(a) Segmentaciones del experto 1 (etiquetas)



(b) Predicciones de la red



(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

Figura 4.8: Comparativa entre las segmentaciones de la excavación (2)

De cualquier manera, la red es capaz de superar cualquier identificación visual de segmentaciones en retinografías realizadas por parte de personal no experto. Este hecho resulta bastante prometedor de cara al siguiente apartado.

4.3. Disco óptico y excavación con RIMONEr3

En este apartado se explora un enfoque distinto para intentar mejorar los resultados de la excavación, centrado en utilizar las etiquetas de la base de datos que contienen tanto el disco óptico como la excavación para el entrenamiento de la red. El hecho de tener dos segmentaciones, una dentro de otra, puede ayudar a la red a predecir la excavación, diferenciándola del disco óptico, y a relativizar los tamaños de estas partes de la retinografía.

En este caso, la red debe separar la imagen en tres clases: fondo, disco óptico y excavación. Es por ello que se ha de cambiar el enfoque utilizado para interpretar los datos: será necesaria una tabla para interpretar las métricas del disco óptico y otra para interpretar las correspondientes a la excavación, debido a que las distintas métricas utilizadas (AUC, Índice de Jaccard y de Dice y distancia Hausdorff) son binarias.

Respecto a la *precision* y el *recall*, se trata de métricas definidas en función de cada clase, por lo que no hace falta replantear el enfoque. En la Tabla 4.21, se pueden encontrar las etiquetas correspondientes a cada una de las clases segmentadas.

Zona segmentada	*Precision*	*Recall*
Fondo	0.995	0.996
Disco óptico	0.762	0.815
Excavación	0.829	0.676

Tabla 4.21: *Precision* y *recall* del disco óptico y la excavación

Existe una mejoría en la clasificación de los píxeles correspondientes a la excavación, mientras que los correspondientes al disco óptico han empeorado levemente. A continuación, se muestran las imágenes detalladas de cada una de estas dos segmentaciones.

- Disco óptico:

En los resultados de la Tabla 4.22 destacan las imágenes 2, 20 y 22 como las peores. En estos casos, se obtiene una distancia Hausdorff muy elevada debido a la poca similitud de la forma segmentada, unos índices de Jaccard y Dice muy bajos, debido a la la poca zona común entre la etiqueta y el resultado de la red, y un AUC bajo debido al poco porcentaje de píxeles correctamente clasificados.

Por otro lado, se encuentran unos resultados muy buenos en las imágenes 11, 21 y 23, en las que la distancia Hausdorff ronda el 1.5 %, los índices de Jaccard y Dice son muy próximos a uno y el AUC supera el 0.970. Resulta destacable el caso de las imágenes 6 y 24, en las que se obtienen unos valores de índice de Dice y Jaccard elevados, pero tienen una Distancia Hausdorff muy baja. Esto es debido a que tienen una gran zona en común, pero no se han segmentado los bordes de manera adecuada.

Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
1	0.988	0.889	0.941	2.693
2	0.803	0.599	0.749	4.272
3	0.936	0.859	0.924	2.828
4	0.934	0.865	0.927	2.500
5	0.982	0.905	0.950	2.500
6	0.995	0.886	0.939	9.500
7	0.969	0.895	0.945	2.000
8	0.987	0.821	0.902	2.550
9	0.955	0.805	0.892	4.243
10	0.975	0.886	0.940	2.236
11	0.991	0.931	0.964	**1.500**
12	0.993	0.912	0.954	**1.500**
13	0.958	0.910	0.953	**1.500**
14	0.967	0.869	0.930	2.000
15	0.966	0.890	0.942	2.000
16	0.956	0.885	0.939	2.000
17	0.949	0.842	0.914	2.121
18	0.989	0.771	0.871	3.354
19	0.989	0.858	0.924	2.550
20	0.811	0.581	0.735	3.905
21	0.975	**0.935**	**0.966**	1.581
22	0.954	0.693	0.819	7.018
23	**0.997**	0.927	0.962	1.581
24	0.993	0.800	0.889	4.272
Media	0.963 $\pm$0,060	0.842 $\pm$0,094	0.911 $\pm$0,061	3.008 $\pm$1,840

Tabla 4.22: Métricas del disco óptico

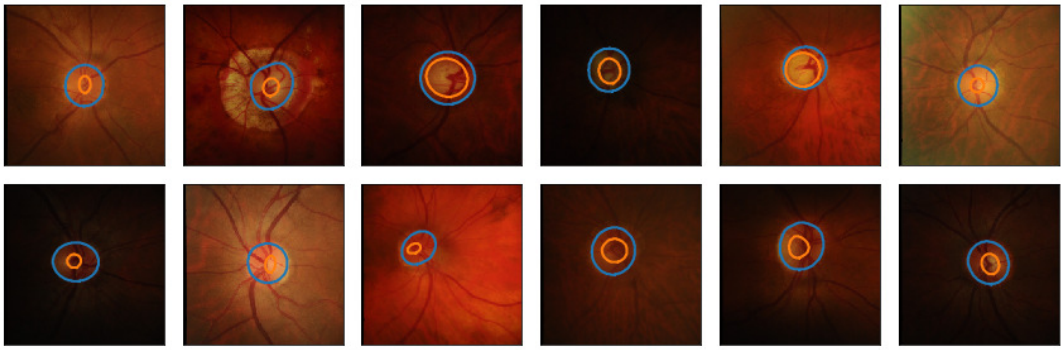
- Excavación:

Los resultados correspondientes a la excavación se encuentran en la Tabla 4.23. En ellos se observan valores en media algo bajos, destacando las imágenes 6, 9 y 22, que muestran unas métricas bajas, y las imágenes 2 y 20, que también mostraban malos resultados en la Tabla 4.22. Sin embargo, se aprecia una mejoría en las que se pueden considerar como mejores imágenes: índices de Jaccard y Dice superiores a 0.81 y distancias Hausdorff situadas entre el 3 % y el 3.5 % en las imágenes 3, 5 y 17. Estos resultados mejoran levemente los de la Tabla 4.19, correspondientes al entrenamiento de la red exclusivamente con las etiquetas de la excavación.

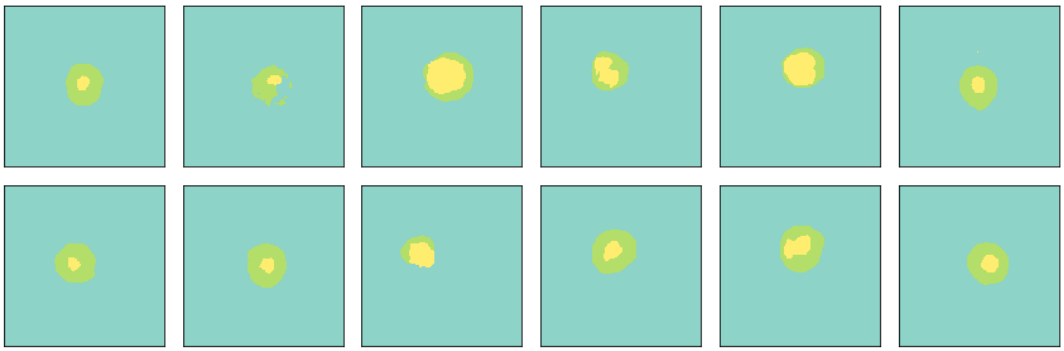
Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
1	0.863	0.533	0.696	2.236
2	0.604	0.152	0.264	7.616
3	0.940	**0.835**	**0.910**	3.500
4	0.897	0.566	0.723	4.610
5	0.940	0.817	0.899	3.350
6	0.974	0.645	0.785	17.500
7	0.915	0.714	0.833	**1.803**
8	0.919	0.509	0.674	4.500
9	0.935	0.144	0.252	12.021
10	0.795	0.585	0.738	4.031
11	0.916	0.519	0.683	4.610
12	0.917	0.733	0.846	2.000
13	0.862	0.715	0.834	2.236
14	0.940	0.725	0.840	2.828
15	0.835	0.614	0.761	2.828
16	0.896	0.775	0.873	2.500
17	0.911	0.814	0.898	3.041
18	0.966	0.447	0.618	4.610
19	0.892	0.641	0.782	2.000
20	0.691	0.259	0.411	4.528
21	0.993	0.368	0.538	5.408
22	0.975	0.358	0.527	9.220
23	0.926	0.664	0.798	3.354
24	**0.996**	0.628	0.771	3.808
Media	0.896 $\pm$0,089	0.573 $\pm$0,195	0.706 $\pm$0,183	4.756 $\pm$3,553

Tabla 4.23: Métricas de la excavación

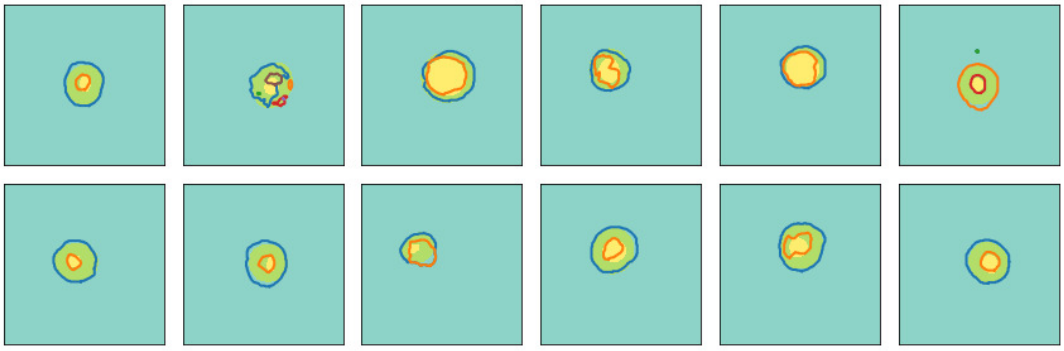
Por último, en las Figuras 4.9 y 4.10, se encuentran las imágenes resultantes del entrenamiento, y se puede comprobar las distintas segmentaciones de la red. Resulta curioso que haya imágenes en las que se ha obtenido métricas bajas para el disco óptico y la excavación, como la imagen 2, que se encuentra en la Figura 4.9c, no tiene prácticamente área en común con la segmentación del experto, y la 20, en la Figura 4.10c. Así mismo, la imagen 19 se trata de una segmentación bastante buena, que es muy similar a la realizada por el experto.



(a) Segmentaciones del experto 1 (etiquetas)

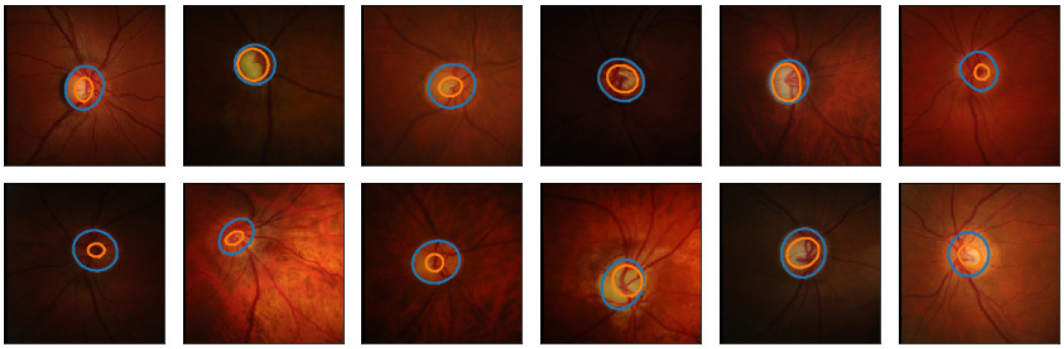


(b) Predicciones de la red

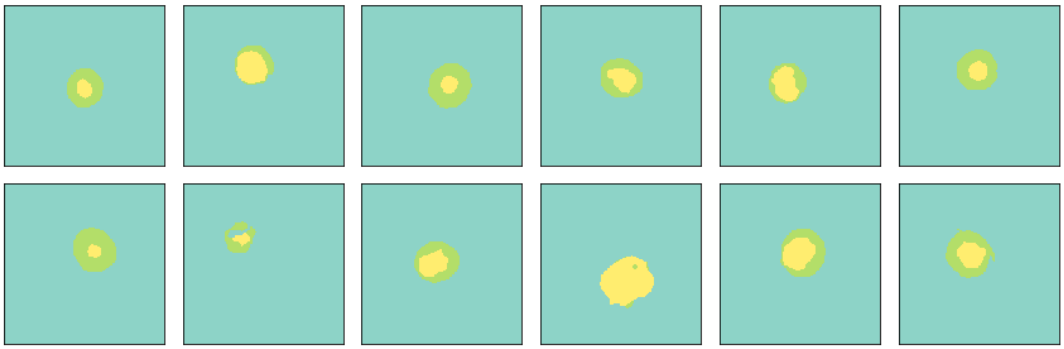


(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

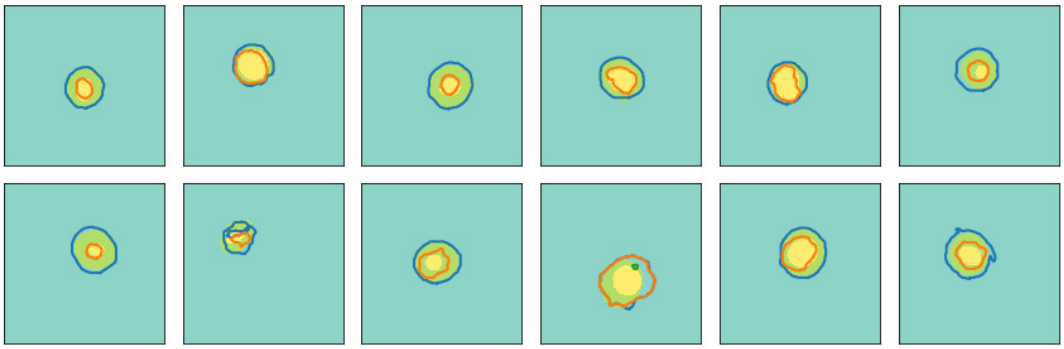
Figura 4.9: Comparativa de las segmentaciones del disco óptico y la excavación (1)



(a) Segmentaciones del experto 1 (etiquetas)



(b) Predicciones de la red



(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

Figura 4.10: Comparativa de las segmentaciones del disco óptico y la excavación (2)

Como se ha podido comprobar, en el caso de entrenar la red con las etiquetas del disco óptico y la excavación, la segmentación del disco óptico empeora levemente. Sin embargo, la de la excavación mejora, compensándose de alguna manera la una con la otra. El resultado global es considerablemente bueno y esperanzador para el siguiente apartado: el entrenamiento con una base de datos mayor.

4.4. Disco óptico y excavación con RIGA

Este último subapartado se centrará en el entrenamiento de la red con un mayor número de imágenes lo que, a priori, parece que proporcionará unos mejores resultados a la red frente a datos que no ha procesado anteriormente. Para ello, se seguirá el enfoque visto en el apartado anterior, segmentar la excavación y el disco óptico en la misma imagen. La diferencia con el caso anterior es que se dispone de más datos para los conjuntos de entrenamiento, validación y test, siendo este último superior a las cien imágenes. Por ello, en las tablas se reflejarán únicamente aquellas imágenes que tengan relevancia, las que sean mejores o peores dentro del conjunto de test.

Si se deseara observar los resultados asociados al resto de imágenes, se puede recurrir al Apéndice A1.1, en el que se han presentado los datos de la misma manera que en el apartado anterior: en primer lugar, las tablas que contienen las métricas correspondientes al disco óptico, seguidos de los de la excavación y, posteriormente, las imágenes correspondientes, con las segmentaciones del experto, las predicciones de la red y la superposición de ambas.

En este apartado se encuentran los resultados dispuestos de esta misma manera: se pueden encontrar las métricas correspondientes al disco óptico en la Tabla 4.25 y los correspondientes a la excavación en la Tabla 4.26. Por último, los resultados gráficos de las imágenes se encontrarán en las Figuras 4.11 y 4.12. Para facilitar el acceso a cada imagen, se ha creado una columna a la derecha de la tabla, que contiene la posición que ocupa la imagen en las Figuras 4.11 y 4.12. Para ver la imagen asociada a una fila, hay que tomar su posición dentro de la figura, que se encuentra en la última columna de la tabla. Es necesario recordar que las imágenes se encuentran organizadas de izquierda a derecha y de arriba abajo, por lo que habrá que contar las posiciones siguiendo este orden hasta llegar a la imagen en cuestión.

Estos datos vendrán precedidos por la Tabla 4.24, que se corresponde con la *precision* y el *recall*, en la que se encuentran valores elevados y muy compensados para todas las clases, lo que presagia unos buenos resultados para el resto de métricas.

Zona segmentada	*Precision*	*Recall*
Fondo	0.997	0.997
Disco óptico	0.904	0.906
Excavación	0.870	0.881

Tabla 4.24: *Precision* y *recall* de las segmentaciones del disco óptico y la excavación

- Disco óptico:

En la Tabla 4.25 se encuentran casos muy buenos, en los que la distancia Hausdorff ronda la unidad, o incluso es menor que ella, como son las imágenes 5, 15, 31, 33 y 36. Además, estos resultados superan el valor de 0.980 en el Índice de Dice y el de 0.960 en el Índice de Jaccard, con un AUC superior a 0.980. Por otro lado, se observa que los peores casos son las imágenes 7, 64, 98 y 110. En estos casos, los resultados tienen una distancia Hausdorff situada entre el 2.5 % y 6 %, y unos Índices Jaccard y Dice cuyo mínimo valor está sobre el 0.780. Aun así, estas métricas son levemente inferiores a la media del caso anterior, en los que la red se entrenó con la base de datos RIMONEr3, lo que es otra demostración de la mejoría del modelo.

Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)	Posición
5	0.996	**0.970**	**0.985**	**0.707**	1
7	0.931	0.845	0.916	3.606	2
15	0.989	0.961	0.980	**0.707**	3
26	0.947	0.893	0.943	2.236	6
29	0.962	0.867	0.929	2.500	8
31	0.988	0.966	0.983	1.000	9
33	0.987	0.961	0.980	1.118	10
36	0.991	0.965	0.982	**0.707**	11
42	0.976	0.876	0.934	2.500	12
48	0.948	0.886	0.940	2.500	14
54	0.920	0.839	0.913	2.121	15
60	0.956	0.897	0.946	2.500	17
64	0.935	0.862	0.926	5.385	18
65	0.931	0.860	0.924	2.550	19
71	0.939	0.878	0.935	2.500	21
74	0.983	0.959	0.979	**0.707**	24
93	0.984	0.877	0.934	2.121	29
98	0.916	0.832	0.908	2.500	32
107	0.978	0.944	0.971	1.118	33
108	**0.997**	0.885	0.939	2.000	34
110	0.995	0.786	0.880	2.693	35
Media	0.974 $\pm$0,017	0.924 $\pm$0,031	0.960 $\pm$0,017	1.613 $\pm$0,636	

Tabla 4.25: Métricas del disco óptico de las imágenes más relevantes

De cualquier manera, los valores altos en media y bajos en varianza de este entrenamiento se traducen en unos resultados muy buenos, que son los mejores para disco óptico vistos hasta ahora.

- Excavación:

Respecto a las métricas correspondientes a la excavación, visibles en la Tabla 4.26, destaca que los datos se concentran en un rango de valores menor, que se refleja tanto en la media como en la varianza de las métricas. Los peores valores de distancia Hausdorff, correspondientes a las imágenes 43, 60, 73 y 89, están situados entre el 3 % y el 4 %. Resulta destacable que no necesariamente coinciden los peores valores de Índice de Jaccard y Dice con los de distancia Hausdorff, ya que en este caso las peores imágenes son la 72, 73 y 89, que tienen valores de índice de Jaccard y Dice menores a 0.570 y 0.730, respectivamente. Nuevamente, esto mejora al caso anterior ya que, al entrenar la red con RIGA, se obtienen valores del orden de la media de las métricas de RIMONEr3, que suponen los mejores resultados del modelo.

Por otro lado, cabe destacar que en ningún caso se consigue una distancia Hausdorff inferior a la unidad, y que el Índice de Dice se sitúa por encima de 0.970 únicamente en un caso. Además, si se visualizan las imágenes en las Figuras 4.11 y 4.12, es aún más notable esta mejoría. No existe gran diferencia entre las segmentaciones del experto y las realizadas por la red.

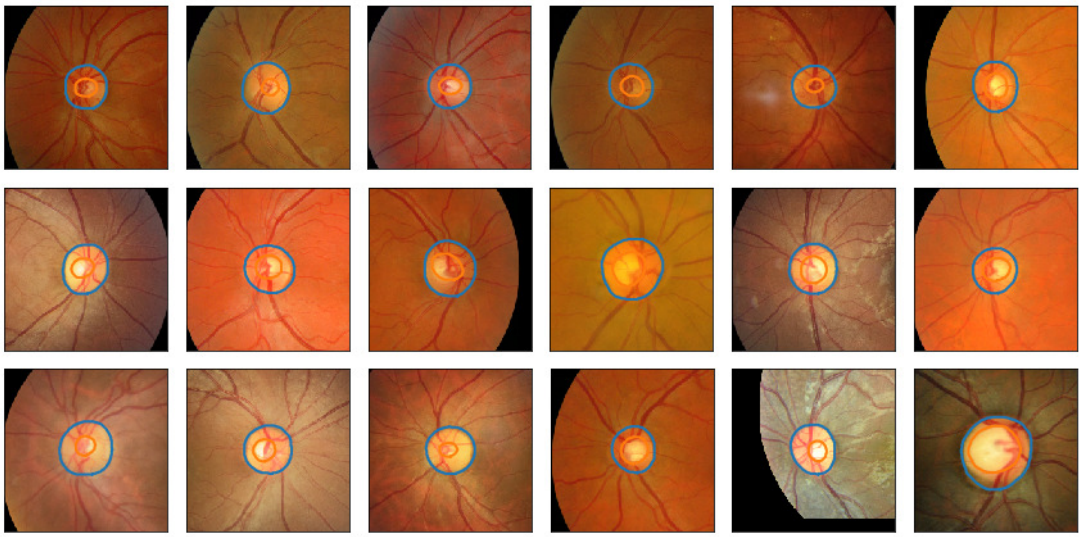
Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)	Posición
7	0.983	0.638	0.779	2.500	2
17	0.809	0.619	0.764	2.062	4
24	**0.998**	0.598	0.749	2.236	5
26	0.977	**0.940**	**0.969**	**1.000**	6
27	0.870	0.621	0.766	3.162	7
42	0.962	0.873	0.932	1.118	12
43	0.995	0.589	0.742	3.606	13
57	0.965	0.905	0.950	**1.000**	16
60	0.996	0.584	0.738	4.610	17
68	0.990	0.879	0.956	1.803	20
72	0.961	0.558	0.717	3.041	22
73	0.903	0.567	0.723	3.606	23
75	0.987	0.593	0.745	3.808	25
80	0.970	0.913	0.954	**1.000**	26
84	0.938	0.813	0.987	1.803	27
89	0.777	0.514	0.679	3.202	28
94	0.967	0.900	0.947	**1.000**	30
97	0.961	0.609	0.757	2.062	32
107	0.984	0.897	0.946	**1.000**	33
Media	0.943 $\pm$0,040	0.775 $\pm$0,088	0.871 $\pm$0,058	2.101 $\pm$0,722	

Tabla 4.26: Métricas de la excavación de las imágenes más relevantes

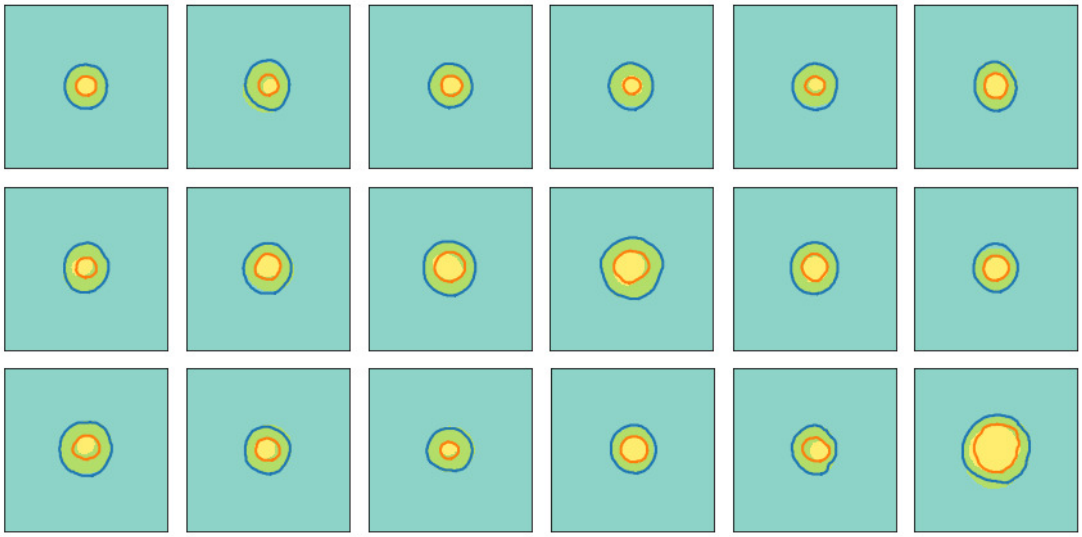
Las imágenes mejor segmentadas, teniendo en cuenta tanto el disco óptico como la excavación, son la 26, 42 y 107, de entre las que se puede destacar la 107 como la mejor. Se puede encontrar esta imagen en la Figura 4.12, ocupando la última posición. Por otro lado,

las peores imágenes son la 7 y la 60, siendo esta última la peor. Esta imagen en concreto tiene asociada la posición 17, y se encuentra en el penúltimo lugar de la tercera fila de la Figura 4.11, comenzando por la izquierda. A pesar de estos últimos resultados, se trata de imágenes muy bien segmentadas, con una diferencia leve con la segmentación del experto y sin casos erróneos con segmentaciones para una misma clase en distintas zonas de la imagen.

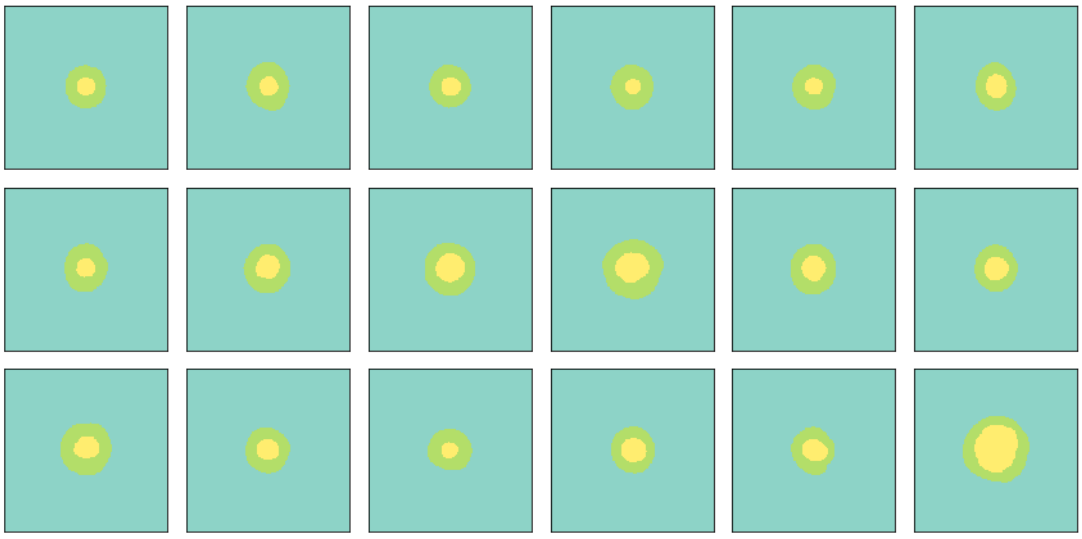
En último lugar, se encuentran las gráficas que representan la evolución de la ACC, el AUC y la LOSS en función de las épocas, que se encuentran en la Figura 4.13. El modelo prácticamente se sobreentrena en la época 4, demostrando que la red tiene una muy rápida adaptación al conjunto de entrenamiento. En comparación con las gráficas obtenidas con anterioriad, es destacable la estabilidad de las curvas una vez alcanzado el mínimo (o máximo).



(a) Segmentaciones del experto 1 (etiquetas)

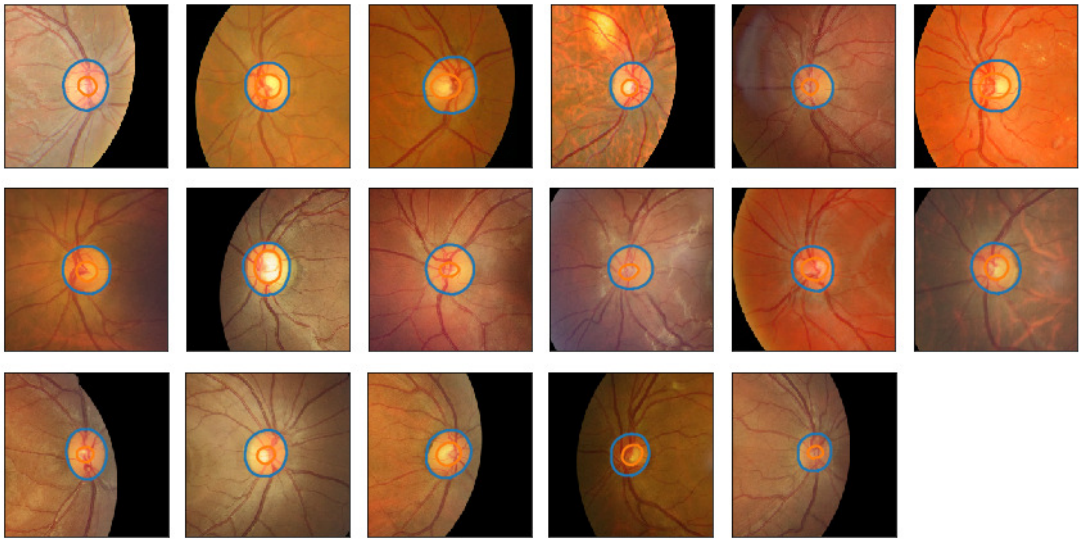


(b) Predicciones de la red

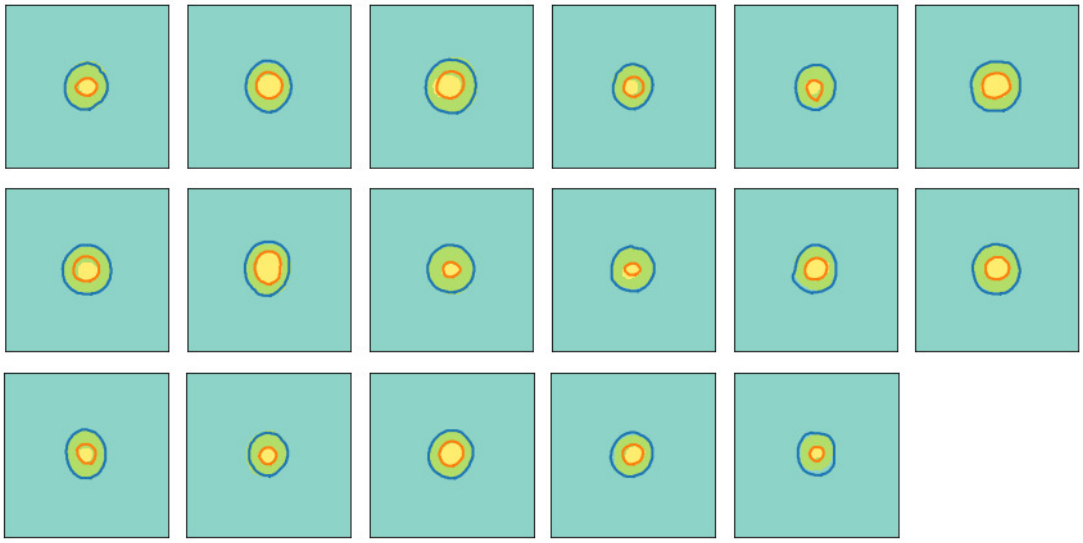


(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

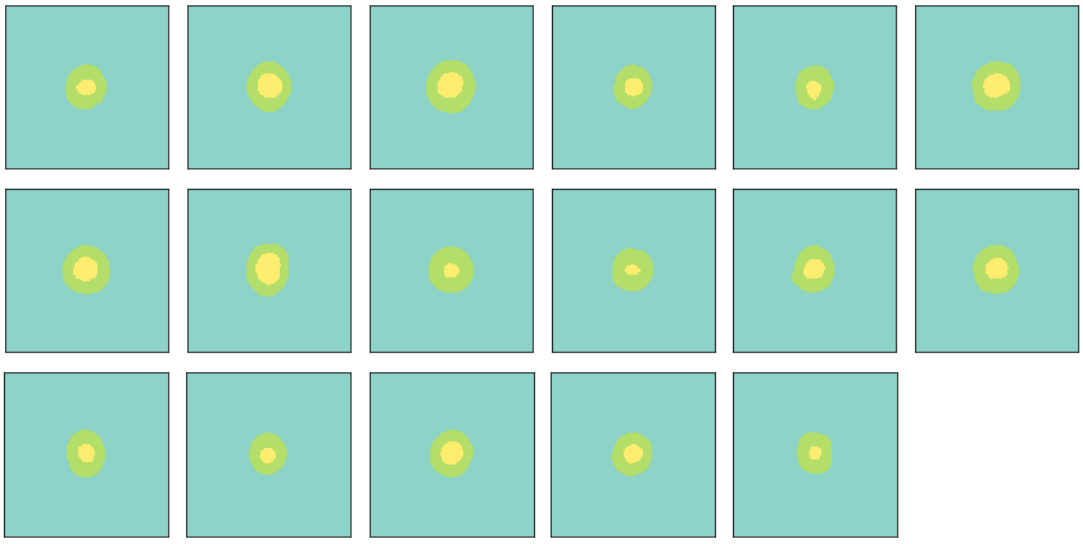
Figura 4.11: Comparativa de las segmentaciones de la imágenes más destacables (1)



(a) Segmentaciones del experto 1 (etiquetas)

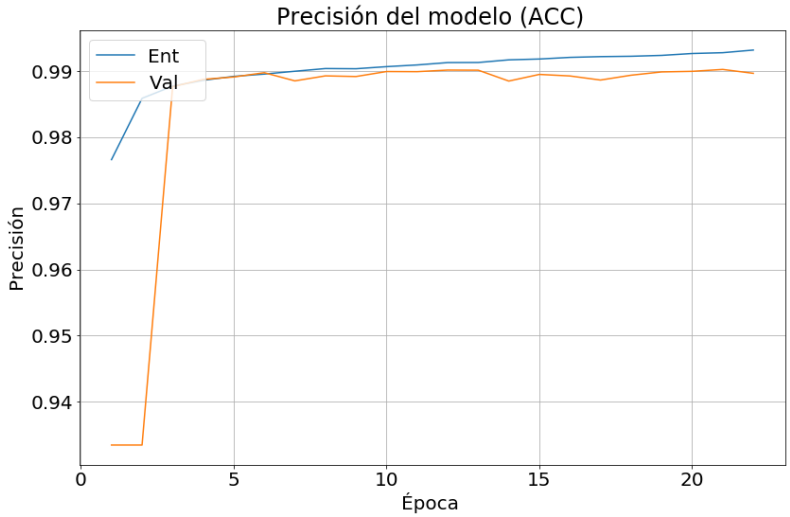


(b) Predicciones de la red

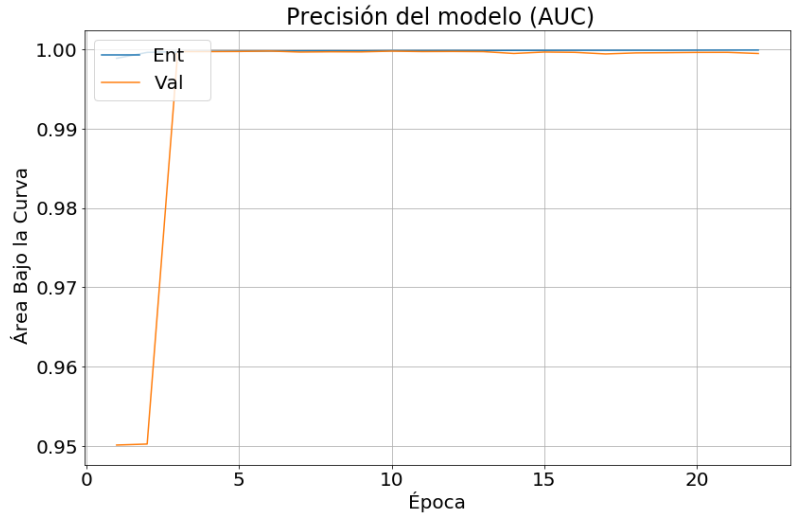


(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

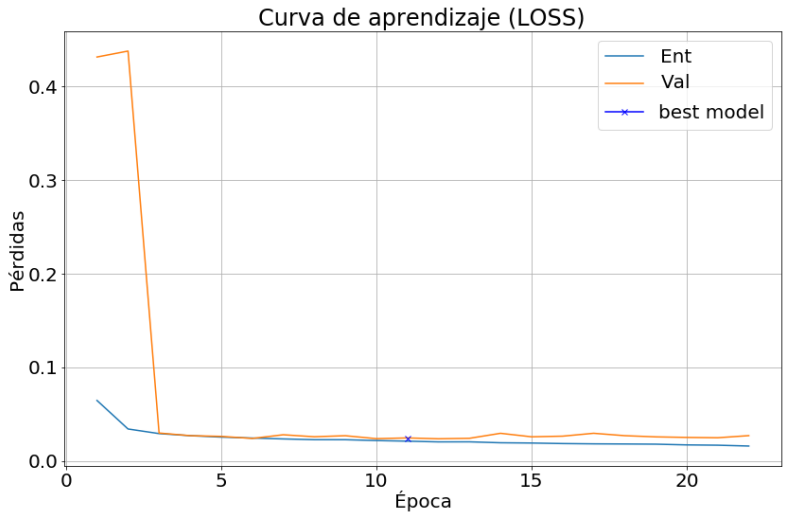
Figura 4.12: Comparativa de las segmentaciones de la imágenes más destacables (2)



(a) Precisión en función de las épocas



(b) AUC en función de las épocas



(c) Pérdidas en función de las épocas

Figura 4.13: Gráficas asociadas al proceso de entrenamiento de la red con RIGA

5 Discusión

En el presente proyecto, se ha buscado una solución a la difícil segmentación de retinografías con el objetivo de diagnosticar el glaucoma, apoyándose en la corriente de investigaciones que pretenden solucionar problemas médicos de esta índole. Con este propósito, se ha obtenido un modelo clínicamente significativo, basado en técnicas de Deep Learning, que es capaz de segmentar imágenes de una manera bastante precisa. El último entrenamiento de la red, realizado con la base de datos RIGA, ha obtenido unas medias de índice de Dice de 0.960 y una distancia Hausdorff de 1.649 % para el disco óptico. En cuanto a la excavación, se ha logrado una media de índice de Dice de 0.863, y de distancia Hausdorff de 2.213 %. Esto, unido a la baja varianza de estas métricas (inferiores a 0.06 en el caso del índice de Dice y menores que el 1 % en la distancia Hausdorff), supone unos resultados muy esperanzadores para la segmentación de retinografías.

Como posible línea futura, podría obtenerse el CDR de cada imagen, que serviría al profesional como parámetro indicativo a la hora de diagnosticar el glaucoma. Esto se podría lograr de varias maneras, pero podría hacerse rápidamente midiendo directamente las áreas del disco óptico y la excavación usando el propio lenguaje de programación con el que se esté trabajando.

Otra manera distinta de abordar el problema sería entrenar el modelo de Deeplabv3+ con el propósito de clasificación de imágenes, utilizando un *Output Size* de 32 (Chen, Zhu et al., 2018); o añadiendo unas capas finales, que podrían destinarse a realizar esta misma tarea, y aplicando un ajuste fino sobre estas. Esta opción podría ser interesante, debido a que el modelo tendría mayor información para clasificar las imágenes como «sospechosas» o «sanas».

Por otro lado, podrían alcanzarse unas métricas ligeramente mejores en caso de utilizar una base de datos aún mayor, que permitiría al modelo extraer al máximo las características de los patrones de la retinografía. Esta afirmación se apoya en el hecho de que los creadores de Deeplabv3+ entrenan el modelo con PASCAL VOC 2012 en su último lanzamiento, una base de datos de unas 11500 imágenes (Everingham, van Gool, Williams, Winn y Zisserman, 2012), obteniendo como resultado un modelo con un índice de Jaccard medio de 0.89 (Chen, Zhu et al., 2018), que es superior al logrado en este proyecto, situado en 0.85,

aproximadamente. Si bien es cierto que sería necesario un gran número de imágenes para conseguir unos resultados notablemente mejores, todavía hay margen de mejora para la excavación. Con este objetivo, podría utilizarse un *Output Stride* de 16 que, si bien produciría unas prestaciones algo inferiores, reduciría considerablemente los tiempos de cómputo. Es necesario recordar que en este trabajo se optó por un *Output Stride* de 8 debido a que se priorizó la bondad del modelo frente a la pesadez computacional.

Como se ha indicado en la introducción, la presente pandemia ha puesto de manifiesto la necesidad de aportar una cierta ayuda al personal médico, y somos los ingenieros quienes podríamos proporcionársela. Posiblemente, el hecho de programar código como si fuera una «caja negra», en el sentido de que el profesional es capaz de obtener una segmentación únicamente a partir de la imagen original en segundos, será de mucha utilidad en los próximos años. De cualquier manera, el hecho de proporcionar una segmentación junto con el CDR puede generar un menor escepticismo en el médico, que tendría algo más que un simple valor numérico para comprobar que la segmentación se ha realizado adecuadamente. Esto sería especialmente útil, ya que no siempre se obtiene un algoritmo infalible al usar un modelo de Deep Learning, y se puede encontrar algún resultado inesperado, como se ha expuesto en el apartado 4.4.

Además, la posibilidad de procesar cualquier tipo de imágenes en la nube de manera gratuita (como se ha hecho a lo largo de este trabajo mediante Google Colaboratory), o mediante un pequeño pago, habiendo desarrollado una aplicación en la nube, supone que no se tengan que realizar grandes inversiones en equipos potentes en estos centros médicos. Este es el principal inconveniente de los algoritmos de Deep Learning, que sería evitado teniendo que pagar únicamente por los recursos consumidos. De esta manera, este enfoque permitiría a centros médicos de todo el país acceder a estas herramientas, independientemente de su localización geográfica. En cualquier caso, el desarrollo de aplicaciones en la nube es un campo que, al igual que la Inteligencia Artificial, está alcanzando niveles que nadie esperaba (Torres, 2020). Hoy en día, es mucho más fácil y rentable recurrir a este tipo de plataformas, que permiten el procesado de imágenes a distancia, y con un tiempo de procesado insignificante.

El proceso que engloba el desarrollo de este proyecto no solamente supone un aporte significativo a la segmentación de retinografías, en el que hay varios enfoques abiertos para su resolución, sino un despertar sobre el aprendizaje ininterrumpido que supone ser ingeniero, y un primer contacto con el mundo de la investigación. Ha sido un ejercicio de información continuo sobre los últimos avances en el campo, y sobre aquellos no tan recientes, que han permitido el vertiginoso desarrollo del campo del Deep Learning. Sin embargo, esto no es más que el comienzo de un largo camino.

A1 Apéndices

A1.1. Resultados de las segmentaciones del modelo tras el entrenamiento usando la base de datos RIGA

- Disco óptico:

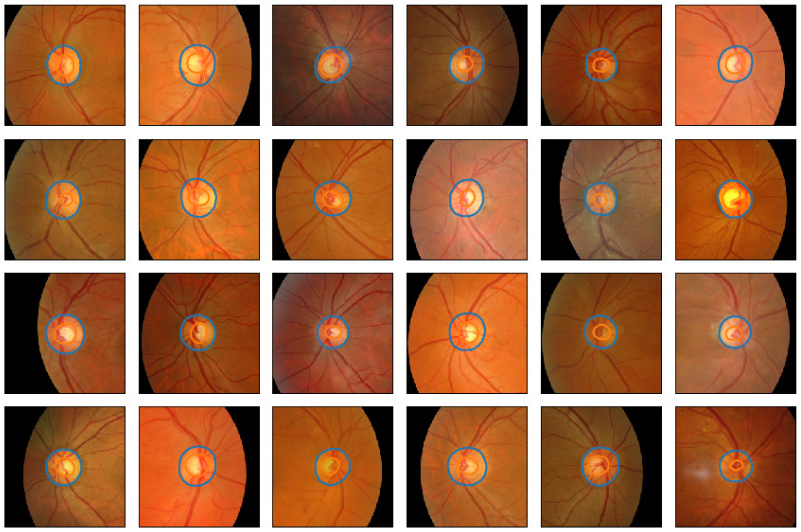
Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
1	0.985	0.960	0.979	1.000
2	0.976	0.930	0.963	1.118
3	0.971	0.917	0.957	1.414
4	0.985	0.956	0.977	1.000
5	0.996	**0.970**	**0.985**	**0.707**
6	0.979	0.926	0.961	1.803
7	0.931	0.845	0.916	3.606
8	0.974	0.927	0.962	1.581
9	0.968	0.930	0.964	1.414
10	0.968	0.925	0.961	1.803
11	0.977	0.950	0.975	**0.707**
12	0.990	0.950	0.974	1.414
13	0.972	0.913	0.955	1.581
14	0.959	0.910	0.953	1.500
15	0.989	0.961	0.980	**0.707**
16	0.971	0.895	0.945	2.000
17	0.990	0.930	0.964	1.581
18	**0.997**	0.940	0.969	1.414
19	0.954	0.902	0.949	2.500
20	0.963	0.925	0.961	1.118
21	0.983	0.950	0.974	1.414
22	0.992	0.961	0.980	1.118
23	0.987	**0.970**	**0.985**	1.000
24	0.993	0.900	0.947	2.000

Tabla A1.1: Métricas del disco óptico (primera tabla)

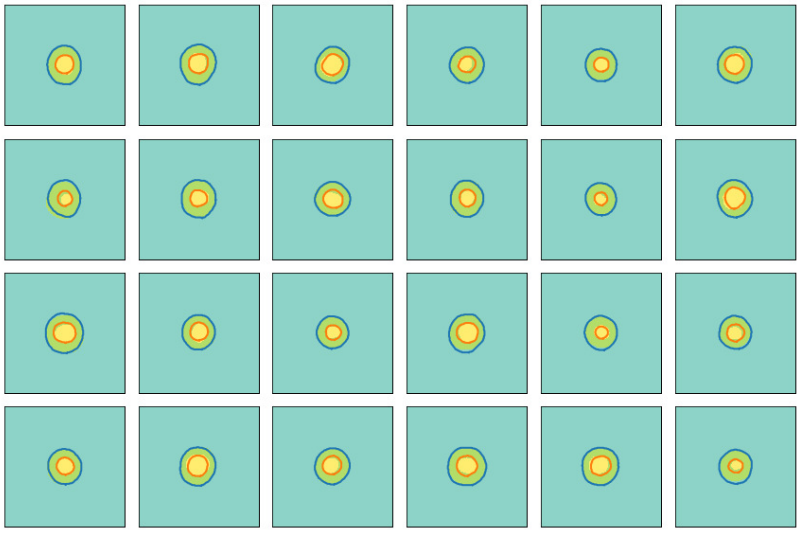
- Excavación:

Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
1	0.928	0.833	0.909	2.236
2	0.938	0.789	0.882	2.000
3	0.909	0.779	0.876	2.121
4	0.951	0.706	0.828	2.500
5	0.962	0.865	0.928	**1.000**
6	0.947	0.852	0.920	1.803
7	0.983	0.638	0.779	2.500
8	0.891	0.764	0.866	1.803
9	0.992	0.849	0.918	2.061
10	0.996	0.938	0.912	1.414
11	0.913	0.681	0.810	2.121
12	0.900	0.801	0.889	2.693
13	0.957	0.823	0.903	1.803
14	0.939	0.785	0.879	2.236
15	0.975	0.806	0.893	1.500
16	0.986	**0.873**	**0.932**	2.000
17	0.809	0.619	0.764	2.062
18	0.970	0.673	0.805	2.236
19	0.928	0.823	0.903	1.581
20	0.889	0.777	0.874	2.693
21	0.985	0.768	0.869	2.062
22	0.982	0.789	0.882	2.000
23	0.923	0.736	0.848	2.236
24	**0.998**	0.598	0.749	2.236

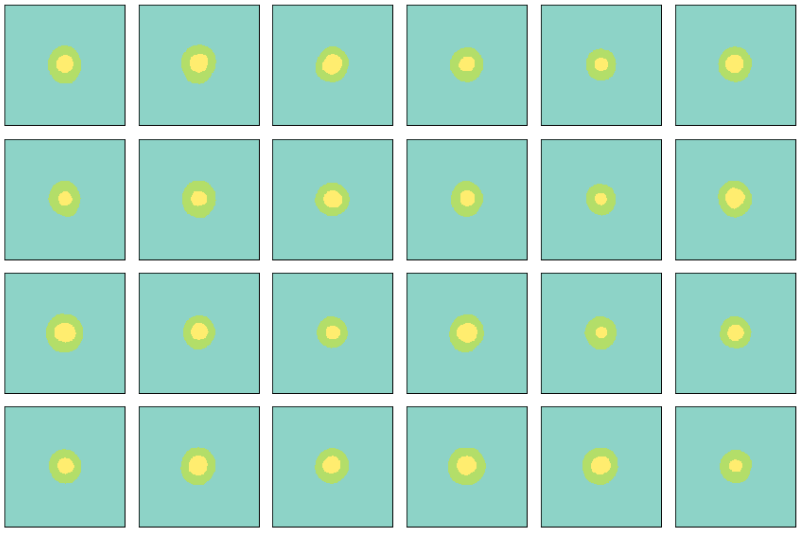
Tabla A1.2: Métricas de la excavación (primera tabla)



(a) Segmentaciones del experto 1 (etiquetas)



(b) Predicciones de la red



(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

Figura A1.1: Comparativa de las segmentaciones de la red (primera tanda)

- Disco óptico:

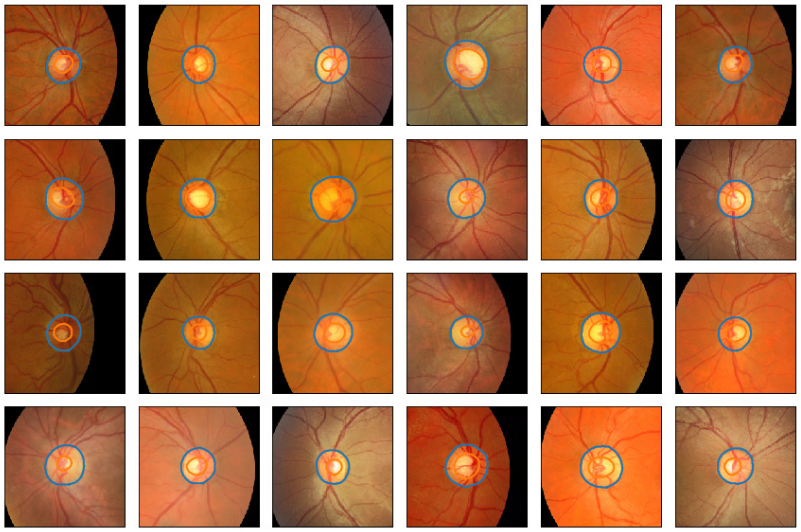
Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
25	0.953	0.898	0.946	2.236
26	0.947	0.893	0.943	2.236
27	0.968	0.921	0.959	1.581
28	**0.993**	0.952	0.976	2.062
29	0.962	0.867	0.929	2.500
30	0.982	0.931	0.964	1.803
31	0.988	**0.966**	**0.983**	1.000
32	0.986	0.950	0.974	1.118
33	0.987	0.961	0.980	1.118
34	0.977	0.952	0.975	1.000
35	0.947	0.893	0.944	1.803
36	0.991	0.965	0.982	**0.707**
37	0.954	0.896	0.945	2.121
38	0.972	0.939	0.968	1.500
39	0.948	0.896	0.945	1.803
40	0.967	0.934	0.966	1.500
41	0.983	0.956	0.978	1.414
42	0.976	0.876	0.934	2.500
43	0.977	0.937	0.967	1.118
44	0.985	0.952	0.976	1.000
45	0.949	0.897	0.946	2.000
46	0.992	0.945	0.972	1.581
47	0.982	0.945	0.972	1.581
48	0.948	0.886	0.940	2.500

Tabla A1.3: Métricas del disco óptico (segunda tabla)

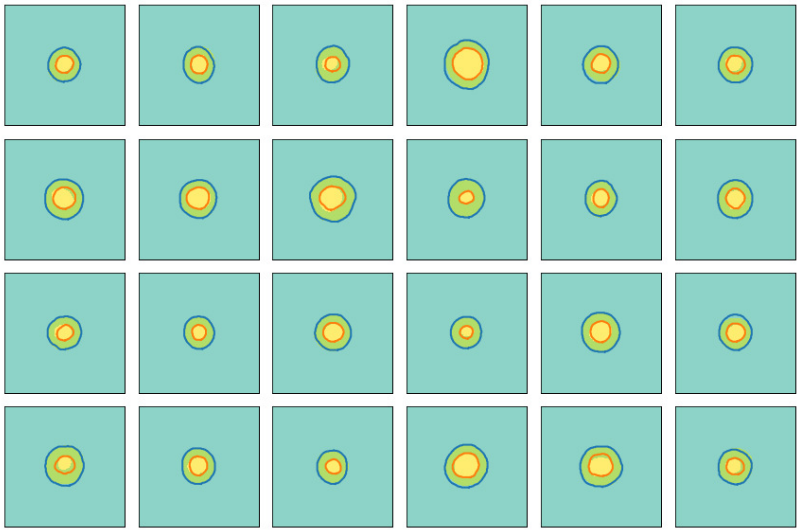
- Excavación:

Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
25	0.959	0.861	0.925	1.500
26	0.977	**0.940**	**0.969**	**1.000**
27	0.870	0.621	0.766	3.162
28	0.968	0.891	0.943	2.000
29	0.972	0.856	0.922	1.414
30	0.931	0.704	0.826	2.915
31	0.945	0.737	0.849	2.915
32	0.943	0.884	0.938	1.500
33	0.937	0.799	0.888	2.500
34	0.919	0.780	0.876	1.500
35	0.904	0.771	0.871	2.062
36	0.935	0.812	0.896	1.803
37	0.859	0.679	0.809	3.162
38	0.919	0.835	0.910	1.414
39	0.937	0.808	0.894	1.581
40	0.931	0.679	0.809	1.500
41	0.916	0.830	0.907	1.581
42	0.962	0.873	0.932	1.118
43	**0.995**	0.589	0.742	3.606
44	0.939	0.796	0.886	2.550
45	0.942	0.844	0.916	1.118
46	0.975	0.871	0.931	1.500
47	0.941	0.759	0.863	3.041
48	0.928	0.679	0.809	3.041

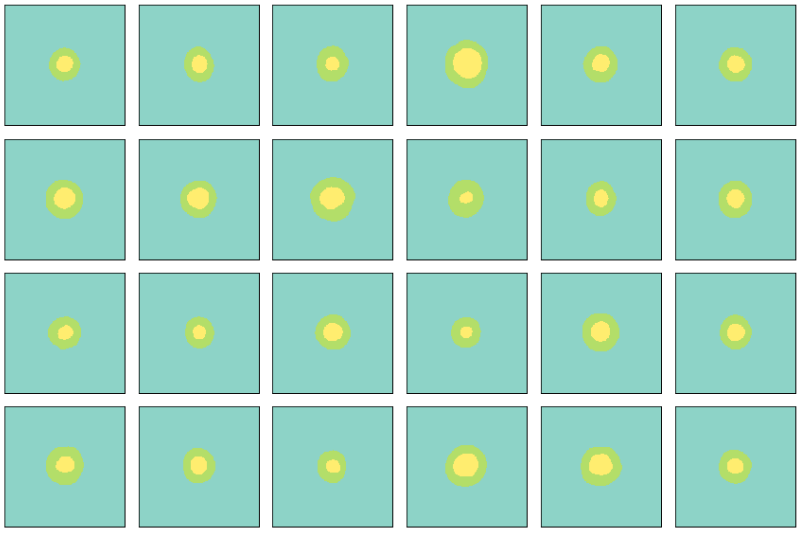
Tabla A1.4: Métricas de la excavación (segunda tabla)



(a) Segmentaciones del experto 1 (etiquetas)



(b) Predicciones de la red



(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

Figura A1.2: Comparativa de las segmentaciones de la red (segunda tanda)

- Disco óptico:

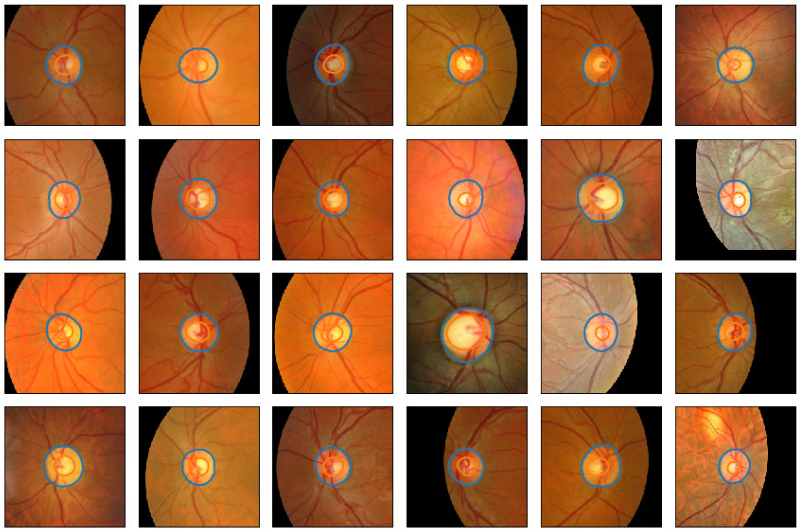
Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
49	0.979	0.949	0.974	**1.000**
50	0.968	0.893	0.943	1.581
51	0.961	0.910	0.953	1.803
52	0.967	0.933	0.965	1.414
53	0.984	0.950	0.974	**1.000**
54	0.920	0.839	0.913	2.121
55	0.974	0.908	0.952	1.803
56	0.988	0.915	0.956	2.500
57	**0.997**	0.922	0.959	1.118
58	0.982	0.929	0.963	2.000
59	0.980	0.946	0.972	1.803
60	0.956	0.897	0.946	2.500
61	0.974	0.922	0.959	1.581
62	0.972	0.939	0.969	1.803
63	0.983	**0.955**	**0.977**	**1.000**
64	0.935	0.862	0.926	5.385
65	0.931	0.860	0.924	2.550
66	0.967	0.934	0.966	1.118
67	0.992	0.946	0.972	1.581
68	0.996	0.938	0.968	1.581
69	0.981	0.925	0.961	1.414
70	0.976	0.932	0.965	1.414
71	0.939	0.878	0.935	2.500
72	0.955	0.906	0.951	1.803

Tabla A1.5: Métricas del disco óptico (tercera tabla)

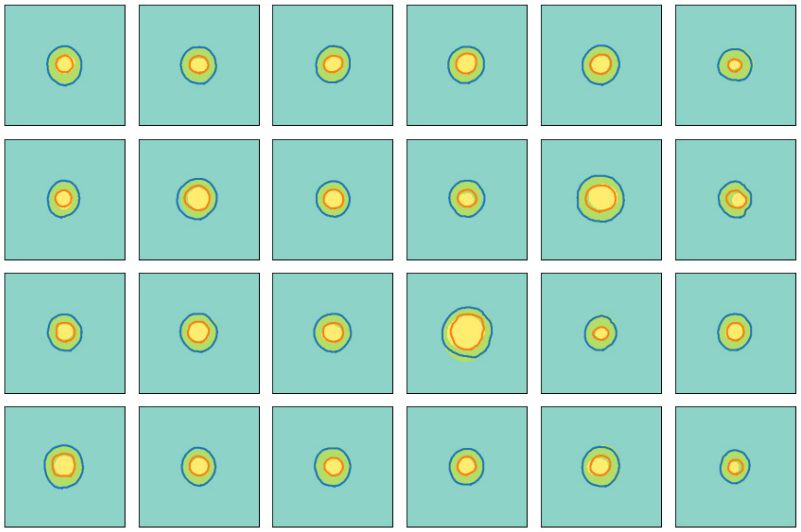
- Excavación:

Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
49	0.862	0.724	0.840	2.500
50	0.997	0.798	0.888	2.000
51	0.947	0.885	0.939	1.581
52	0.949	0.898	0.946	**1.000**
53	**0.998**	0.797	0.887	1.803
54	0.972	0.737	0.848	1.581
55	0.885	0.770	0.870	2.500
56	0.989	0.808	0.894	2.500
57	0.965	0.905	0.950	**1.000**
58	0.994	0.702	0.825	2.500
59	0.931	0.764	0.866	3.162
60	0.996	0.584	0.738	4.610
61	0.991	0.743	0.852	2.121
62	0.958	0.874	0.933	1.803
63	0.955	0.773	0.872	2.121
64	0.925	0.821	0.901	3.808
65	0.945	0.764	0.866	2.500
66	0.933	0.787	0.881	2.000
67	0.996	0.752	0.858	2.500
68	0.990	**0.910**	**0.956**	1.803
69	0.940	0.879	0.936	1.414
70	0.970	0.805	0.892	1.803
71	0.910	0.684	0.813	3.000
72	0.961	0.558	0.717	3.041

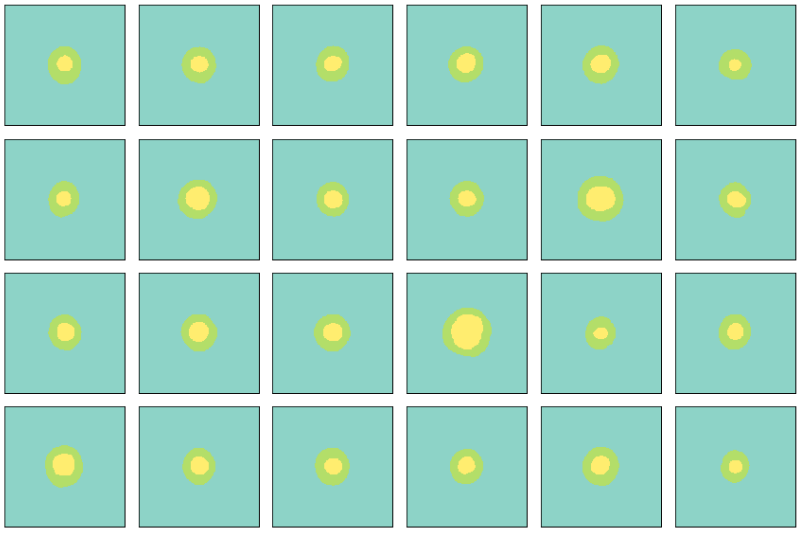
Tabla A1.6: Métricas de la excavación (tercera tabla)



(a) Segmentaciones del experto 1 (etiquetas)



(b) Predicciones de la red



(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

Figura A1.3: Comparativa de las segmentaciones de la red (tercera tanda)

- Disco óptico:

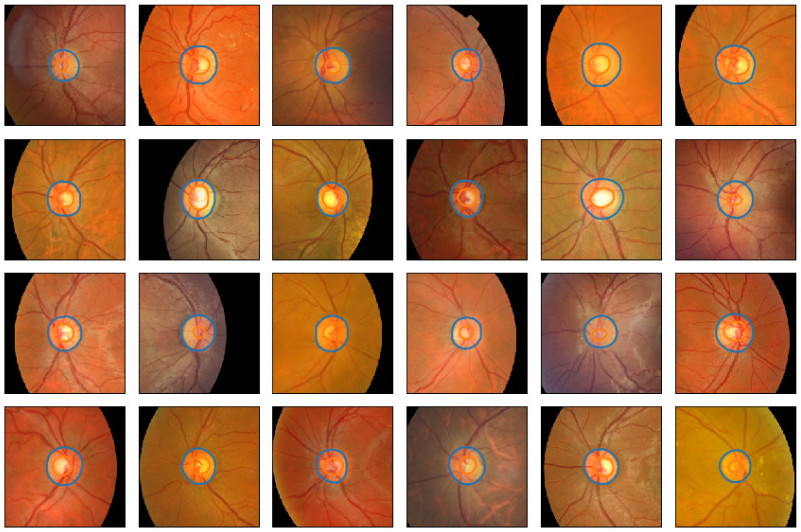
Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
73	0.984	0.918	0.957	1.414
74	0.983	**0.959**	**0.979**	**0.707**
75	0.995	0.942	0.970	1.414
76	**0.997**	0.930	0.964	1.118
77	0.987	0.937	0.968	2.062
78	0.994	0.931	0.964	2.236
79	0.977	0.938	0.968	1.118
80	0.973	0.921	0.959	1.414
81	0.995	0.925	0.961	1.414
82	0.965	0.929	0.963	1.414
83	0.978	0.903	0.949	2.500
84	0.962	0.924	0.960	1.581
85	0.979	0.931	0.965	1.581
86	0.970	0.937	0.967	1.414
87	0.966	0.918	0.957	2.121
88	0.995	0.908	0.952	1.118
89	0.978	0.921	0.959	1.500
90	0.972	0.938	0.968	1.414
91	0.978	0.956	0.977	1.414
92	0.983	0.925	0.961	1.000
93	0.984	0.877	0.934	2.121
94	0.969	0.920	0.958	1.500
95	0.979	0.951	0.975	1.000
96	0.970	0.933	0.965	1.000

Tabla A1.7: Métricas del disco óptico (cuarta tabla)

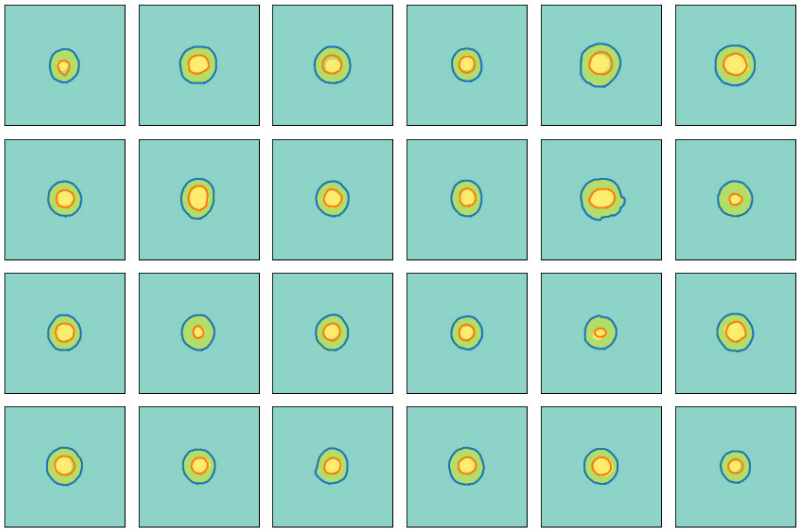
- Excavación:

Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
73	0.903	0.567	0.723	3.606
74	0.931	0.838	0.912	1.803
75	0.987	0.593	0.745	3.808
76	0.943	0.743	0.853	1.803
77	0.980	0.741	0.851	3.000
78	0.949	0.816	0.898	2.062
79	0.945	0.868	0.929	1.500
80	0.970	0.913	0.954	**1.000**
81	0.903	0.785	0.879	1.581
82	0.940	0.865	0.927	1.500
83	0.980	0.866	0.928	1.414
84	0.938	**0.913**	**0.987**	1.803
85	0.959	0.875	0.933	1.500
86	0.924	0.752	0.858	1.500
87	0.960	0.849	0.919	1.803
88	**0.998**	0.727	0.842	2.000
89	0.777	0.514	0.679	3.202
90	0.923	0.834	0.910	2.236
91	0.922	0.716	0.835	2.500
92	0.889	0.777	0.875	2.000
93	0.900	0.789	0.882	2.000
94	0.967	0.900	0.947	**1.000**
95	0.898	0.756	0.861	2.500
96	0.996	0.659	0.795	2.121

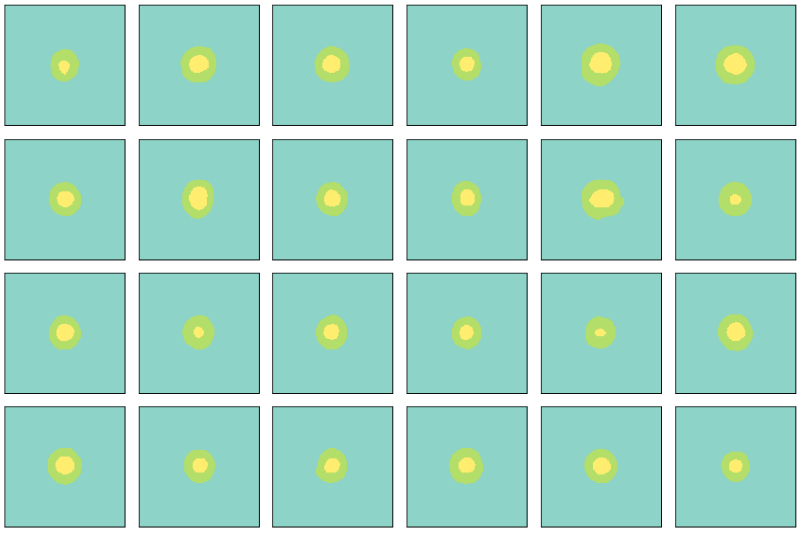
Tabla A1.8: Métricas de la excavación (cuarta tabla)



(a) Segmentaciones del experto 1 (etiquetas)



(b) Predicciones de la red



(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

Figura A1.4: Comparativa de las segmentaciones de la red (cuarta tanda)

- Disco óptico:

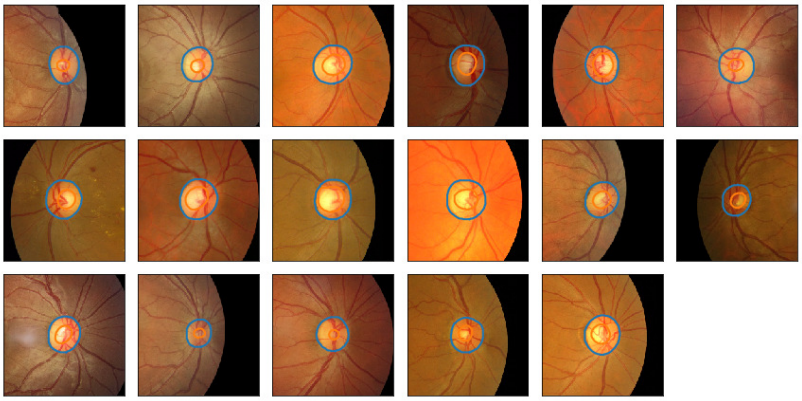
Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
97	0.970	0.940	0.969	1.118
98	0.916	0.832	0.908	2.500
99	0.979	0.947	0.973	1.414
100	0.943	0.885	0.939	2.236
101	0.988	0.950	0.974	1.118
102	0.977	0.934	0.966	**1.000**
103	0.989	**0.963**	**0.981**	1.118
104	0.984	0.951	0.975	1.118
105	0.973	0.946	0.972	1.803
106	0.957	0.905	0.950	2.121
107	0.978	0.944	0.971	1.118
108	**0.997**	0.885	0.939	2.000
109	0.979	0.941	0.970	1.118
110	0.995	0.786	0.880	2.693
111	0.969	0.927	0.962	1.581
112	0.974	0.937	0.967	1.500
113	0.974	0.925	0.961	1.500
Media	0.974 $\pm$0,017	0.924 $\pm$0,031	0.960 $\pm$0,017	1.613 $\pm$0,636

Tabla A1.9: Métricas del disco óptico (quinta tabla)

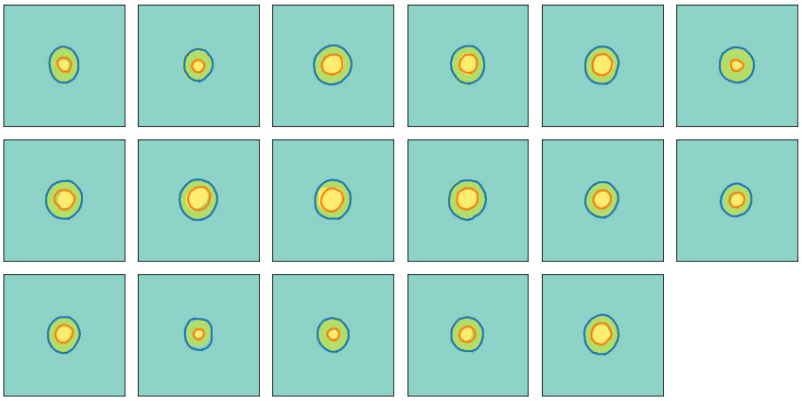
- Excavación:

Imagen	AUC	Índice Jaccard	Índice Dice	Distancia Hausdorff (%)
97	0.961	0.609	0.757	2.062
98	0.933	0.827	0.905	1.118
99	0.933	0.822	0.902	2.000
100	0.887	0.735	0.848	2.121
101	0.936	0.839	0.913	1.500
102	0.883	0.746	0.854	2.062
103	0.957	0.733	0.846	3.354
104	0.927	0.735	0.847	2.500
105	0.857	0.713	0.833	4.272
106	0.947	0.846	0.917	2.062
107	0.984	**0.897**	**0.946**	**1.000**
108	0.940	0.775	0.873	1.581
109	0.970	0.824	0.904	1.414
110	0.933	0.715	0.834	1.803
111	**0.999**	0.644	0.784	1.803
112	0.942	0.672	0.804	3.041
113	0.995	0.829	0.907	1.414
Media	0.943 $\pm$0,040	0.775 $\pm$0,088	0.887098 $\pm$0,058	2.101 $\pm$0,722

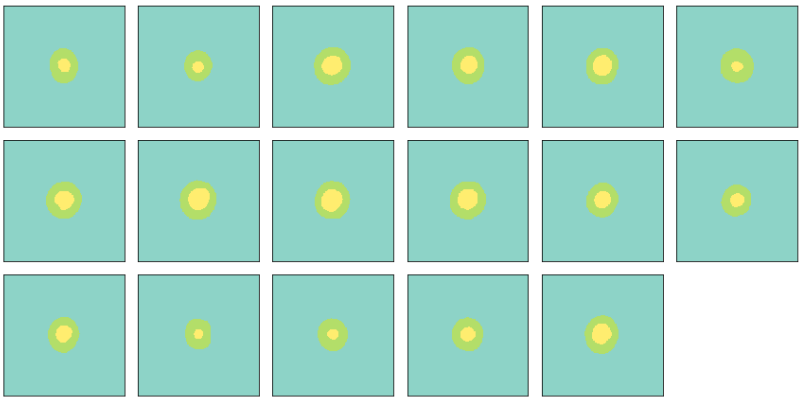
Tabla A1.10: Métricas de la excavación (quinta tabla)



(a) Segmentaciones del experto 1 (etiquetas)



(b) Predicciones de la red



(c) Resultados de la red superpuestas con las segmentaciones del experto 1 (amarillo)

Figura A1.5: Comparativa de las segmentaciones de la red (quinta tanda)

Referencias

Aggarwal, C. C. (2018). *Neural networks and deep learning* (1st ed.). Yorktown Heights, NY, USA: Springer.

Almazroa, A., Alodhayb, S., Osman, E., Ramadan, E., Hummadi, M., Dlaim, M., … Lakshminarayanan, V. (2018, 3 de diciembre). *Riga dataset (retinal fundus images for glaucoma analysis).* https://doi.org/10.7302/Z23R0R29

Anónimo. (s.f.-a). *Cloud tensor processing unit (tpu).* https://cloud.google.com/tpu/docs/tpus

Anónimo. (s.f.-b). *Curva roc.* https://es.wikipedia.org/wiki/Curva_ROC

Anónimo. (s.f.-c). *Hausdorff distance.* https://en.wikipedia.org/wiki/Hausdorff_distance

Anónimo. (s.f.-d). *Jaccard index.* https://en.wikipedia.org/wiki/Jaccard_index

Anónimo. (s.f.-e). *Neuritis óptica: síntomas y tratamiento de la enfermedad.* https://fr-dc.ru/oftalmolog/nevrit-zritelnogo-nerva-simptomy-bolezni-i-sposoby-lecheniya

Anónimo. (s.f.-f). *precision_score.* https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

Anónimo. (s.f.-g). *recall_score.* https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html

Anónimo. (s.f.-h). *Receiver operating characteristic.* https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Anónimo. (s.f.-i). *Roc-auc score.* https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

Anónimo. (2020a). *Diagnóstico de glaucoma.* https://www.oftalmoseo.com/patologias-frecuentes-2/diagnostico-de-glaucoma/

Anónimo. (2020b). *Estadísticas.* https://www.glaucomapatients.org/es/basica/estadisticas/

Anónimo. (2020c, 23 de octubre). *Glaucoma.* https://www.mayoclinic.org/es-es/diseases-conditions/glaucoma/symptoms-causes/syc-20372839

Balan, E., Venkatesan, C., Sumithra, M. G., Akila, M. y Manikandan, M. (2020, octubre). *Method for detecting cup to disk ratio for the prediction of glaucoma disease – a review.*

https://druckhaus-hofmann.de/gallery/7-wj-august-2020.pdf

Berzal, F. (s.f.). *Entrenamiento de redes neuronales.* https://elvex.ugr.es/decsai/deep-learning/slides/NN4%20Training.pdf

Chen, L. C., Papandreou, G., Schroff, F. y Adam, H. (2018, 5 de diciembre). *Rethinking atrous convolution for semantic image segmentation.* https://arxiv.org/pdf/1706.05587.pdf

Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F. y Adam, H. (2018, 30 de enero). *Encoder-decoder with atrous separable convolution for semantic image segmentation.* https://arxiv.org/pdf/1802.02611.pdf

Chollet, F. (2018). *Deep learning with python* (1ª ed.). Shelter Island, NY, USA: Manning Publications.

DelSole, M. (2018, 24 de abril). *What is one hot encoding and how to do it.* https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179

Dertat, A. (2017, 3 de octubre). *Applied deep learning - part 3: Autoencoders.* https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

DiBattista, J. (2020, 8 de julio). *Deep learning on a budget: $450 egpu vs google colab.* https://towardsdatascience.com/deep-learning-on-a-budget-450-egpu-vs-google-colab-494f9a2ff0db

Ertel, W. (2017). *Introduction to artificial intelligence* (2nd ed.). Hochschule Ravensburg-Weingarten, Weingarten, Germany: Springer.

Everingham, M., van Gool, L., Williams, C., Winn, J. y Zisserman, A. (2012). *Visual object classes challenge 2012.* http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

Kan, E. (2018, 11 de diciembre). *Quick ml concepts: Tensors.* https://towardsdatascience.com/quick-ml-concepts-tensors-eb1330d7760f

Kingma, D. P., y Ba, J. L. (2017, 30 de enero). *Adam: A method for stochastic optimization.* https://arxiv.org/pdf/1412.6980.pdf

Lago, M. D., y Bengoa, (2018). *Epidemiología: el glaucoma del siglo xxi.* https://sociedadoftalmologicademadrid.com/revistas/revista-2018/m2018-15a.html

MIAG. (2015). *Rim-one.* http://medimrg.webs.ull.es/research/retinal-imaging/rim-one/

Parsad, N. M. (2018, 14 de junio). *Deep learning in medical imaging v.* https://medium.com/datadriveninvestor/deep-learning-in-medical-imaging-3c1008431aaf

Pradhan, C. (2016, 18 de diciembre). *What is the difference between keras and tensorflow?* https://www.quora.com/What-is-the-difference-between-keras-and-tensorflow

Skansi, S. (2018). *Introduction to deep learning* (1st ed.). University of Zagreb, Zagreb, Croatia:

Springer.

Srivastava, P. (2020, 23 de febrero). *Why use python for ai and machine learning?* https://technative.io/why-use-python-for-ai-and-machine-learning/

Statt, N. (2019, 30 de octubre). *Deepmind's starcraft 2 ai is now better than 99.8 percent of all human players.* https://www.theverge.com/2019/10/30/20939147/deepmind-google-alphastar-starcraft-2-research-grandmaster-level

Torres, J. (2020). *Python deep learning* (1.ª ed.). Barcelona, España: MARCOMBO.

Tsang, S.-H. (2018, 25 de julio). *Review: Xception — with depthwise separable convolution, better than inception-v3 (image classification).* https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568