Práctica

Clasificador Neuronal para la detección de Glaucoma

por

Ander Elkoroaristizabal

Índice

Introducción

El objetivo de esta práctica es la creación de una red neuronal que clasifique ojos en sanos o con Glaucoma a partir de imágenes de la retina. El Glaucoma es una patología que afecta al nervio óptico y cuyos orígenes son diversos, es la segunda causa de ceguera por detrás de la diabetes y los efectos en la pérdida de visión son irreversibles. Las causas que lo producen se pueden tratar si la patología es detectada a tiempo.

Para desarrollar nuestra solución seguiremos la misma estrategia que en el siguiente artículo, el cual hemos utilizado de referencia a lo largo de toda la práctica:

> [1] Diaz-Pinto, A., Morales, S., Naranjo, V. et al. *CNNs for automatic glaucoma assessment using fundus images: an extensive validation*. BioMed Eng OnLine18, 29 (2019). `https://doi.org/10.1186/s12938-019-0649-y`.

En el utilizan las técnicas del *transfer-learning* de modelos preentrenados y su posterior *fine-tuning* para desarrollar redes neuronales convolucionales (*Convolutional Neural Networks*, *CNNs*) partiendo de los mismos datos y con el mismo objetivo. Aunque los autores del artículo no han publicado de en abierto el código utilizado, en el artículo sí revelan información sobre los modelos desarrollados y el entrenamiento de estos que nos ha resultado de utilidad. De este artículo extraemos a su vez la gran mayoría del conocimiento sobre el Glaucoma que reproducimos en este informe.

Estrategias para el diagnóstico del Glaucoma

El Glaucoma se caracteriza por la pérdida de fibra óptica debida al aumento de la presión intraocular (*intraocular pressure, IOP*) y/o la pérdida de flujo sanguíneo al nervio óptico. Aún así, se ha observado que la medición del *IOP* no es lo suficientemente específica ni sensitiva para ser un indicador efectivo de Glaucoma, dado que puede haber daño sin que la IOP haya aumentado.

Otro criterio utilizado para el diagnóstico del Glaucoma es el ratio entre el Disco Óptico (*Optic Disc*) y la Copa Óptica (*Optic Cup*), mostrados sobre dos ejemplos, en rojo y azul respectivamente, en la figura 1. En esta misma imagen se muestra la diferencia entre un ojo sano (**a)**), en el cual la Copa Óptica es claramente más pequeña que el Disco Óptico, y un ojo con Glaucoma (**b)**), donde el tamaño de ambos es mucho más similar.

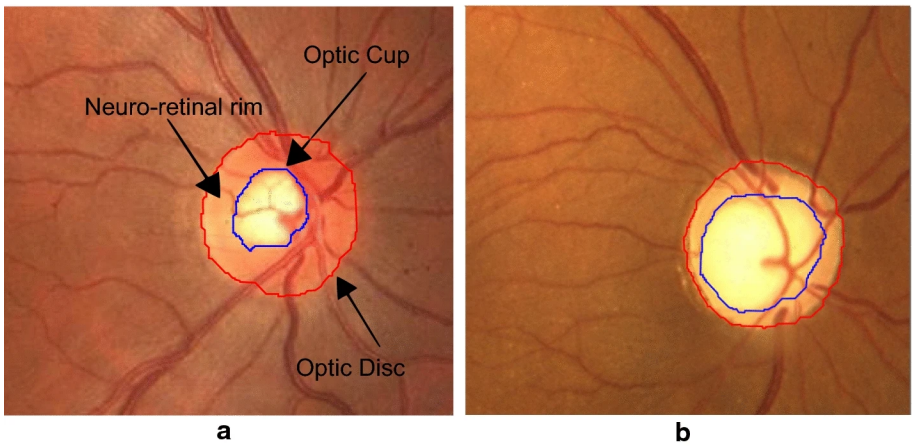

Figura 1: Imágenes digitales de la retina recortadas alrededor del Disco Óptico

Lamentablemente el cálculo del ratio entre Disco Óptico y Copa Óptica, llamado *CDR*, requiere un esfuerzo importante en la difícil tarea de segmentación de estos dos elementos, en la cual hasta los expertos humanos discrepan de manera significativa. Lo mismo ocurre con otras características extraibles manualmente.

En resumen, si bien hay indicadores que se utilizan para el diagnóstico del Glaucoma, estos no son lo suficientemente precisos y/o sensitivos, razón por la cual los autores del artículo [1] optan por la extracción automatizada de características mediante redes neuronales convolucionales, método que nosotros reproducimos.

1. Análisis de datos

Para la realización de esta práctica disponemos de 1707 imágenes en color y de tamaño 224x224, todas ellas clasificadas en normales o anormales. Los datos están públicamente disponibles en Kaggle [2]. Las imágenes se nos dan ya preparadas para aplicar la validación cruzada, divididas en 10 *folds* y particionado cada *fold* en entrenamiento, validación y evaluación. Tal y como indican en [1], estas 1707 imágenes provienen de 5 bases de datos públicas diferentes:

BBDD	**Ojo con Glaucoma**	**Ojo sano**	**Total**
HRF [3]	27	18	45
Drishti-GS1 [4]	70	31	101
RIM-ONE [5]	194	261	455
sjchoi86-HRF [6]	101	300	401
ACRIMA [1]	396	309	705
En total	788	919	1707

Listado de las Bases de Datos utilizadas

Como mencionan en el artículo, esto tiene dos implicaciones importantes: el método de obtención de las imágenes no es necesariamente el mismo, y los expertos que han categorizado las imágenes en sanas o con Glaucoma no son los mismo, por lo que pueden incluso haber seguido criterios diferentes para el diagnóstico.

Es importante mencionar también que las imágenes utilizadas ya han sido preprocesadas, en concreto todas han sido recortadas de manera automática utilizando un cuadrado delimitante de $1,5$ veces el radio del Disco Óptico. Este preprocesamiento está fundamentado en el hecho de que el Glaucoma afecta principalmente al Disco Óptico y sus alrededores. Podemos observar este preprocesado en las figuras 2 y 3.

Para el análisis visual de los datos, por el cual comenzamos, utilizaremos el conjunto de entrenamiento del `Fold0`. La figura 2 muestra 10 de las imágenes en este *fold* junto con su etiqueta.

La primera pregunta que hace falta hacerse es si nosotros, como personas no expertas, somos capaces de determinar porque una imagen se ha considerado normal o anormal. Para mí la conclusión es clara: como persona no experta soy incapaz de determinar si una imagen corresponde a un ojo sano o a uno con Glaucoma. La única diferencia que puedo ver entre ambos grupos es el tono más rojizo de las imágenes anormales, lo cual no es un atributo que realmente sirva para dividir ambos casos. También observo que, tal y como ya hemos comentado, utilizar el criterio del CDR (*Cup to Disc Ratio*) es muy difícil, dado que resulta muy complejo dividir que es la Copa y que el Disco. En resumen, se trata de una tarea compleja, dado que sólo personas expertas son capaces de realizarla, y cabe esperar que la modelización no resulte tampoco fácil.

La figura 3 muestra otras 10 imágenes del conjunto de entrenamiento, de donde podemos ver que la variedad entre imágenes es mayor que la mostrada en la figura anterior.

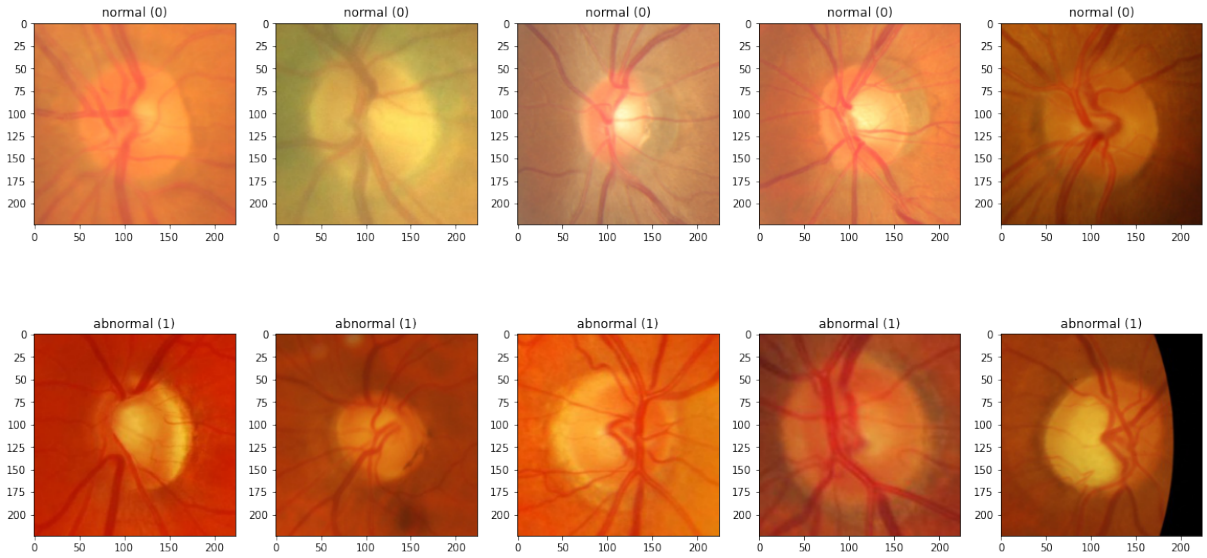

Figura 2: 10 imágenes del conjunto de entrenamiento del *Fold0*

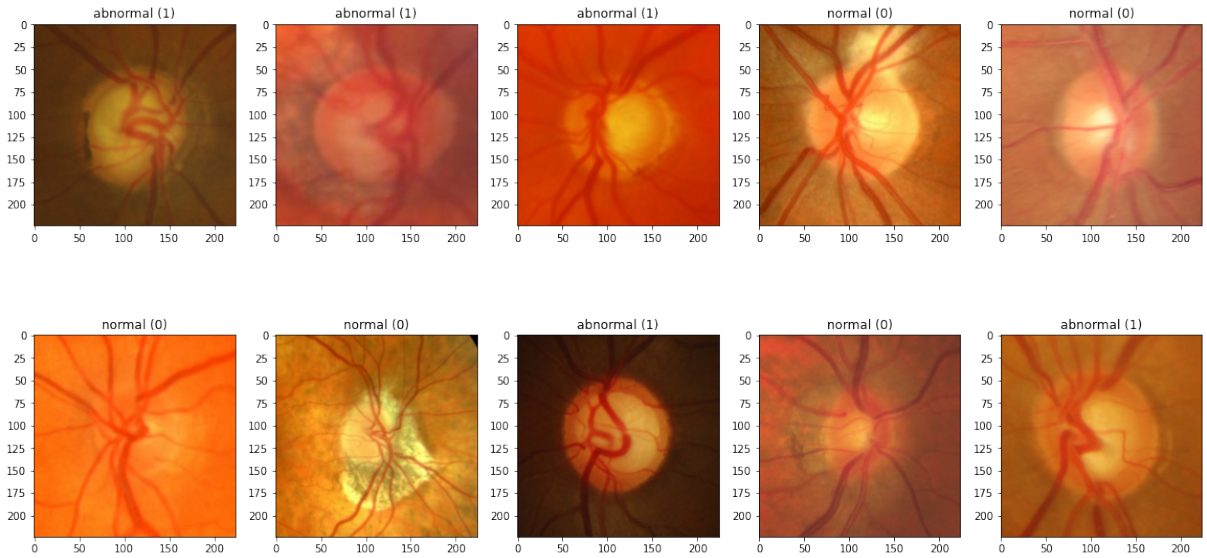

Figura 3: 10 imágenes diferentes del conjunto de entrenamiento del *Fold0*

Por otra parte, como en muchas imágenes el color es muy homogéneo hemos probado de aplicar la técnica de ecualización de histogramas (*Histogram Equalization* [7]) a las imágenes de la figura 3, para así aumentar el contraste de las imágenes y ver si esto nos ayuda a distinguirlas y por lo tanto podría ayudar al clasificador. Aunque el mayor contraste facilita la distinción de la venas del Disco Óptico, la separación de Copa Óptica y Disco Óptico sigue siendo igual de difícil, por lo que concluimos que no merece la pena aplicar esta transformación.

Para concluir el análisis visual es interesante observar que de querer aumentar los datos (hacer *data augmentation*) tendrían sentido tanto las rotaciones y volteos aleatorios como la aplicación de un pequeño zoom, que son las operaciones utilizadas por los autores del artículo de referencia. Las traslaciones, por el contrario, corren el riesgo de sacar de la imagen la parte de interés, la más cercana al Disco Óptico, por lo que no parece que sea buena idea utilizarlas.

Respecto al análisis numérico, el *Fold0* contiene 1379 imágenes para el entrenamiento con un 45 % de ojos con Glaucoma, 154 imágenes para la validación con un 46 % de ojos con Glaucoma y 174 imágenes para la evaluación con un 54 % de ojos con Glaucoma. Todas las imágenes son de tamaño 224×224 y a color, por lo que las dimensiones de cada imagen son $(224, 224, 3)$. Analizando todos los *folds* se observa que en todos ellos hay el mismo número de imágenes de validación y evaluación, pero en los *fold*s 1 y 2 hay respectivamente 3 y 7 imágenes de entrenamiento más, probablemente por error. En todos los *folds* las imágenes son en color y de tamaño 224×224. Por otra parte, si se ve cierta variabilidad en la proporción de casos anormales, especialmente en el conjunto de evaluación, que varía desde un 40 % para el *fold* 9 hasta un 53 % para los *folds* 1 y 6. Esto contrasta con las proporciones en los conjuntos de entrenamiento, más concentrados alrededor del 46 %. En consecuencia se tendrá que tener esto en cuenta al analizar los resultados de la validación cruzada, ya que estos pueden variar ligeramente de un *fold* a otro, al ser las distribuciones de los conjuntos de evaluación diferentes.

Evaluación de los modelos

La métrica que utilizaremos para evaluar los modelos será el ***F1-Score***. La *F1-Score* es la media armónica de la precisión P (de *precision*) y la exhaustividad R (de *recall*):

$$\text{F1-Score} := 2 \cdot \frac{P \cdot R}{P + R}.$$

En consecuencia toma valores altos cuando ambas métricas son altas, pero baja rápidamente cuando alguna de las dos es baja. La precisión y la exhaustividad se definen a su vez como

$$P := \frac{TP}{TP + FP} \quad \text{y} \quad R := \frac{TP}{TP + FN},$$

donde TP representa los verdaderos positivos (*True Positives*), FP los falsos positivos (*False Positives*) y FN los falsos negativos (*False Negatives*). Puesto por escrito, la precisión responde a la pregunta "¿Qué proporción de las predicciones positivas han sido realmente correctas?", mientras que al exhaustividad responde a la pregunta "¿Que proporción de los positivos reales han sido identificados?" [9]. La *F1-Score* es por lo tanto no-simétrica, dado que evalúa que proporción de los casos positivos hemos conseguido identificar, y a coste de cuantas predicciones positivas erróneas.

En aplicaciones médicas el coste de no detectar la enfermedad suele ser mucho mayor que el de detectarla erróneamente, pero tampoco es posible hacerle pruebas a todo los pacientes. En otras palabras, necesitamos detectar el máximo de casos con enfermedad, pero también debemos reducir al máximo la proporción de casos donde habiendo dicho que había enfermedad, nos equivocamos. Tal y como explicamos en la introducción, este es también nuestro caso.

Si fijamos la clase formada por las imágenes de ojos con Glaucoma como clase positiva, esto es exactamente lo que la *F1-Score* mide, por lo que esta métrica es adecuada para nuestro caso.

Técnicas y tecnologías utilizadas

Entrenar una CNN de cero requiere grandes cantidades de datos etiquetados, de los cuales nosotros carecemos. Afortunadamente en tareas de visión por computador existe una opción alternativa, el ***transfer-learning*** [8, Sección 9.1.2], utilizando algunas capas de modelos ya entrenados para extraer las características de la imagen y añadir al final algunas capas nuevas que utilicen estas características para nuestro propósito concreto. Como paso extra existe también la opción de hacer ***fine-tuning***, consistente en entrenar las capas extraídas del otro modelo, para adaptarlas también a nuestro problema.

Por otra parte, cuando se dispone de pocos datos es habitual que los modelos obtenidos hagan *overfitting*, que extraigan supuestos patrones del conjunto de entrenamiento que no ocurren en el conjunto de validación y evaluación, lo cual lleva a un peor rendimiento del modelo. Una de las técnicas utilizadas para paliar este problema es el ***early-stopping*** [8, Sección 4.3.2], consistente en dejar de entrenar el modelo cuando alguna métrica que estemos monitorizando no mejore durante un número de épocas predeterminado, típicamente la función de pérdida o la métrica de validación. Otra de las técnicas utilizadas es el ***data-augmentation*** [8, Sección 9.1.3], que consiste en aplicar transformaciones aleatorias a los datos de forma que las imágenes de entrenamiento no sean siempre las mismas, pero las imágenes transformadas no destaquen como parte del conjunto de entrenamiento, es decir, que no sean distinguibles del resto de imágenes.

Respecto a las tecnologías, para el entrenamiento de las redes neuronales se ha utilizado `Tensorflow` y su distribución de `Keras`. También se han utilizado las librerías `pandas`, `numpy` y `matplotlib`, y el servicio *Weights & Biases* para el seguimiento de los experimentos realizados y la visualización de las curvas de aprendizaje durante el entrenamiento. Finalmente, para la ejecución del código se ha utilizado la plataforma de computación en la nube *Google Colab*.

2. Modelado sobre un único *fold*

Para poder obtener conclusiones preliminares en un plazo razonable de tiempo comenzamos evaluando 5 modelos distintos en un único *fold*, el *Fold0*.

Al tratarse de un problema de clasificación binaria la función de pérdida utilizada ha sido la *binary cross-entropy*, y la capa de salida una única neurona con función sigmoide. Además, todos los modelos que hemos entrenado están basados en el *transfer-learning* de modelos entrenados sobre el conjunto de datos *ImageNet* y distribuidos como parte de `Keras` [10]. Los primeros tres modelos que hemos entrenado están basados en la red *EfficientNet B0*, y los dos últimos en la red *VGG-19*, que es la red con la cual los autores del artículo obtienen un mayor *F1-Score*. En los modelos 2, 3 y 5 hemos aplicado también el *fine-tuning* del modelo base, dado que sin este paso extra los modelos sufrían de *underfitting*, de no ser capaces de obtener muy buenos resultados ni siquiera sobre el conjunto de entrenamiento.

En el entrenamiento de los dos últimos modelos hemos utilizado también la expansión artificial del conjunto de datos (*data augmentation*) utilizada en el artículo original, consistente en rotaciones aleatorias, *zoom* en un rango de 0 a 0,2 y volteos horizontales y verticales, obteniendo una reducción del *overfiting* de los modelos.

Para el *transfer-learning* en todos los modelos hemos prescindido de las capas de clasificación originales del modelo base, y la hemos sustituido por una capa de *GlobalAveragePolling2D*, una capa de *BatchNormalization*, una capa de *dropout* con probabilidad del 20 %, y finalmente una capa *fully connected* para la clasificación.

Como disponemos de pocos datos, en todos los modelos utilizamos el *early-stopping*, monitorizando para ello la función de pérdida sobre el conjunto de validación, y quedándonos con la mejor época (determinada esta sí mediante el *F1-Score*) del modelo. Para calcular este *F1-Score* es necesario fijar el umbral de probabilidad entre clases, la probabilidad por encima de la cual se predecirá que la imagen es de la clase positiva, es decir con Glaucoma. Para todos los modelos hemos fijado esta cantidad en 0,5, y al escoger el modelo optimizando el *F1-Score* ya obtenemos un modelo para el cual este umbral es suficientemente bueno[1].

Para todos los modelos se ha realizado una búsqueda de los hiperparámetros óptimos. Para el entrenamiento de los modelos sin *fine-tuning* se han probado los optimizadores *Adam* y *SGD*, mientras que para el *fine-tuning* se ha utilizado el optimizador *SGD*, como recomiendan en [12] junto con el recorte de gradiente (*gradient clipping*) según norma. De esta manera se facilita

[1] Otra opción es escoger la época del modelo con menor pérdida, pero de esta manera haría falta escoger después el umbral utilizando los datos de validación [11], que con pocos datos sería fácil que introdujese cierto *overfitting*.

que las modificaciones de los pesos se mantenga pequeñas, de forma que los pesos ya entrenados no sufran cambios bruscos en las primeras iteraciones del refinamiento. Se ha probado con diferentes valores del *learning rate* y el tamaño del *batch* (este último de 1 a 32 [13]), y se han hecho pruebas con y sin *momentum* al usar el optimizador *SGD*. Como referencia, en en artículo utilizan siempre el optimizador *SGD* con *learning rate* igual a 0,0001 y *momentum* de 0,9.

A continuación resumimos los resultados obtenidos por los mejores modelos de cada tipo.

Modelos basados en EfficientNet B0

Modelo 1: *transfer-learning* de EfficientNet B0

En este modelo entrenamos exclusivamente las capas de clasificación añadidas sobre el modelo base sin capas de clasificación. La configuración que nos ha dado los mejores resultados es el uso del optimizador *Adam* con un *learning rate* de 0,001 y *batch size* de 32, además de una paciencia de 20 épocas en el *early-stopping* y un máximo de 100 épocas en total. Los resultados de entrenamiento, validación y test correspondientes a la mejor época, la número 35, se muestran en la tabla 1, y la figura 4 muestra la matriz de confusión obtenida sobre el conjunto de test.

Conjunto	Binary cross-entropy	Accuracy	F1-Score
Entrenamiento	0,2183	0,9094	0,8991
Validación	0,3501	0,8701	0,8485
Test	0,4116	0,8448	0,8421

Tabla 1: Resultados del Modelo 1

Como podemos ver el rendimiento alcanzado en entrenamiento no es especialmente alto (comparado con los valores que obtendremos más adelante), y observando la curva de aprendizaje parece que el entrenamiento se estaba estancando ya, por lo que cabe pensar que este modelo padece de *underfitting*, al no tener la flexibilidad suficiente para predecir los datos de entrenamiento. Al mismo tiempo, se ve un ligero descenso en el rendimiento del modelo de entrenamiento a validación, y de validación a evaluación, aunque probablemente no lo suficientemente grande para hablar de *overfitting*[2].

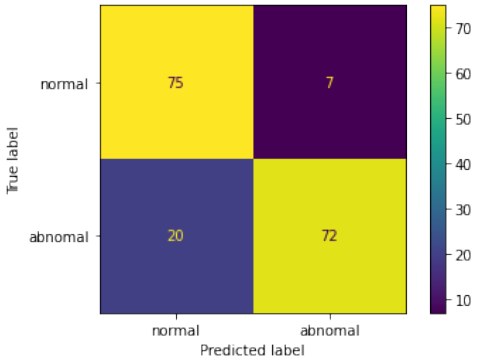

Figura 4: Matriz de confusión del Modelo 1 sobre datos de test

Observando la matriz de confusión vemos también que el modelo obtiene una gran cantidad de Falsos Negativos, por lo que no es un modelo adecuado para la tarea que tenemos entre manos.

[2]Aunque pueda parecer contradictorio, *underfitting* y *overfitting* pueden ocurrir de manera simultánea.

Modelo 2: *fine-tuning* del Modelo 1

En este modelo partimos de los pesos obtenidos en el Modelo 1 y entrenamos las capas de clasificación añadidas y las 20 últimas capas del modelo base, con la excepción de las capas de *BatchNormalization*, dado que modificar estas capas empeoraría en gran medida el rendimiento del modelo [15]. La configuración que nos ha dado los mejores resultados es el uso del optimizador *SGD* con un *learning rate* de 0,01, *momentum* igual a 0,9 y *batch size* de 16, además de una paciencia de 15 épocas en el *early-stopping* y un máximo de 50 épocas en total. Los resultados de entrenamiento, validación y test correspondientes a la mejor época, la número 22, se muestran en la tabla 2, y la figura 5 muestra la matriz de confusión obtenida sobre el conjunto de test.

Conjunto	Binary cross-entropy	Accuracy	F1-Score
Entrenamiento	0,0354	0,9920	0,9912
Validación	0,3776	0,8961	0,8857
Test	0,4225	0,8851	0,8901

Tabla 2: Resultados del Modelo 2

En este caso el evaluación del modelo es clara, padece de un visible *overfitting*: es capaz de clasificar casi a la perfección los datos de entrenamiento, pero su rendimiento baja mucho tanto sobre el conjunto de validación como en evaluación. Aún así, los resultados son mejores que los del Modelo 1, debido a la mayor flexibilidad del modelo, que le ha permitido aprender mejor los patrones presentes en los datos. Para mejorar el modelo necesitaríamos reducir el *overfitting*, para lo cual tendría sentido utilizar el *data-augmentation* o aumentar el ratio de *dropout*.

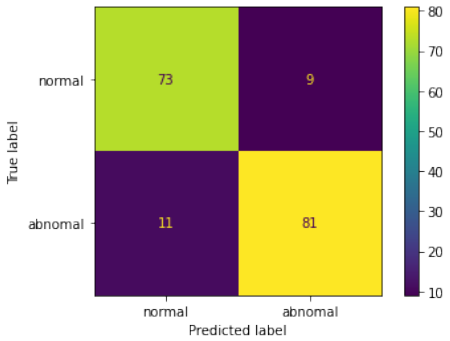

Figura 5: Matriz de confusión del Modelo 2 sobre datos de test

En la matriz de confusión podemos ver reflejada la mejora observada en la tabla: aunque el número de Falsos Positivos aumenta en 2, el número de Falsos Negativos desciende de 20 a 11, por lo que el Modelo 2 no sólo obtiene mejores métricas y más aciertos, si no que además es más adecuado para nuestra tarea.

Modelo 3: *fine-tuning* del Modelo 2

En este modelo partimos de los pesos obtenidos en el Modelo 2 y entrenamos las capas de clasificación añadidas y todas las capas del modelo base (con la excepción de las capas de *BatchNormalization*). La configuración que nos ha dado los mejores resultados es el uso del optimizador *SGD* con un *learning rate* de 0,0001, *momentum* igual a 0,9, *gradient clipping* según norma de 0,5 y *batch size* de 32, además de una paciencia de 10 épocas en el *early-stopping* y un máximo de 25 épocas en total. Los resultados de entrenamiento, validación y test correspondientes a la mejor época, la número 13, se muestran en la tabla 3, y la figura 6 muestra la matriz de confusión obtenida sobre el conjunto de test.

Conjunto	Binary cross-entropy	Accuracy	F1-Score
Entrenamiento	0,0244	0,9884	0,9872
Validación	0,4370	0,9091	0,9079
Test	0,4622	0,8851	0,8969

Tabla 3: Resultados del Modelo 3

En este caso los resultados del modelo son muy similares a los del Modelo 2, al igual que su evaluación: hace un claro *overfitting* de los datos de entrenamiento, que es capaz de clasificar casi a la perfección, pero su rendimiento baja mucho sobre el conjunto de validación, y un poco más aún sobre el conjunto de evaluación. Esto es lo lógico, al haber aumentado en gran medida la flexibilidad de un modelo (el Modelo 2) que ya hacía un *overfitting* tan claro.

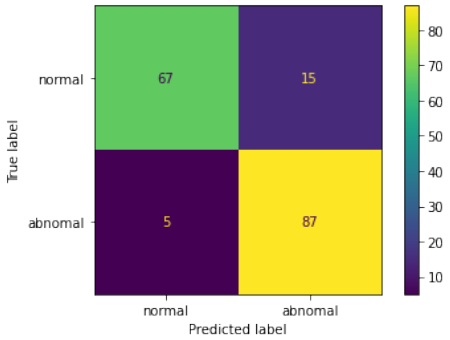

Figura 6: Matriz de confusión del Modelo 2 sobre datos de test

En la matriz de confusión sí podemos ver cierta diferencia respecto al Modelo 2: este modelo comete menos Falsos Negativos, si bien a cambio de cometer más Falsos Positivos. Con estos resultados la elección de si elegir este modelo o el Modelo 2 no es obvia, por lo que lo ideal sería que un experto valorase cual prefiere. Como veremos a continuación esto no será necesario, dado que el Modelo 5 obtenido es claramente superior.

Modelos basados en la red VGG-19

Modelo 4: *transfer-learning* de VGG-19

En este modelo entrenamos exclusivamente las capas de clasificación añadidas sobre el modelo base sin capas de clasificación. La configuración que nos ha dado los mejores resultados es el uso del optimizador *Adam* con un *learning rate* de 0,001 y *batch size* de 16, además de una paciencia de 10 épocas en el *early-stopping* y un máximo de 100 épocas en total. Los resultados de entrenamiento, validación y test correspondientes a la mejor época, la número 5, se muestran en la tabla 4, y la figura 7 muestra la matriz de confusión obtenida.

Conjunto	Binary cross-entropy	Accuracy	F1-Score
Entrenamiento	0,4617	0,7883	0,7641
Validación	0,4415	0,8506	0,8189
Test	0,4582	0,8218	0,8075

Tabla 4: Resultados del Modelo 4

Notemos que debido al *data-augmentation* los resultados de entrenamiento son ligeramente peores que los de validación y evaluación. Por otra parte, el rendimiento de este modelo es inferior al visto en otros modelos, razón por la cual le realizamos un *fine-tuning* en el Modelo 5.

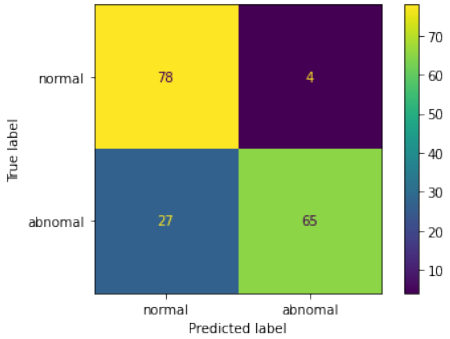

Figura 7: Matriz de confusión del Modelo 4 sobre datos de test

Como podemos ver este modelo comete una gran cantidad de errores, entre los cuales la inmensa mayoría son Falsos Negativos, por lo que no es un modelo adecuado para nuestra tarea.

Modelo 5: *fine-tuning* del Modelo 4

En este modelo partimos de los pesos obtenidos en el Modelo 4 y entrenamos las capas de clasificación añadidas y los 2 últimos bloques del modelo base. La configuración que nos ha dado los mejores resultados es el uso del optimizador *SGD* con un *learning rate* de 0,001, *momentum* igual a 0,9 y *batch size* de 16, además de una paciencia de 10 épocas en el *early-stopping* y un máximo de 50 épocas en total. Los resultados de entrenamiento, validación y test correspondientes a la mejor época, la número 39, se muestran en la tabla 5, y la figura 8 muestra la matriz de confusión obtenida sobre el conjunto de test.

Conjunto	**Binary cross-entropy**	**Accuracy**	**F1-Score**
Entrenamiento	0,1087	0,9594	0,9552
Validación	0,3063	0,9221	0,9189
Test	0,2324	0,9310	0,9362

Tabla 5: Resultados del Modelo 5

Como podemos ver este modelo obtiene los mejores resultados hasta ahora sobre los conjuntos de validación y test, claramente superiores a los de los modelos anteriores. Además el *overfitting* es mucho menor que el de los modelos 2 y 3, los más cercanos en resultados. También parece que este modelo aún podría refinarse un poco más, entrenando un bloque más, por ejemplo.

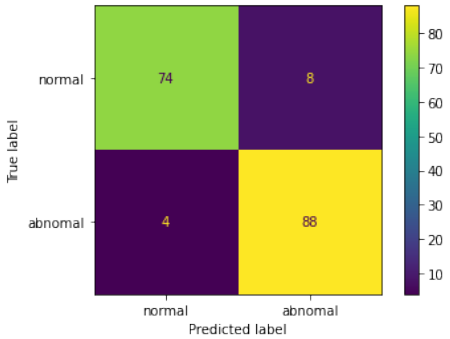

Figura 8: Matriz de confusión del Modelo 5 sobre datos de test

Los buenos resultados del modelo se ven reflejados en la matriz de confusión, donde podemos ver que obtenemos sólo 12 errores en total, siendo además la mayoría de ellos Falsos Positivos.

3. Validación cruzada y discusión

Como hemos visto en la sección anterior el mejor modelo obtenido ha sido el Modelo 5, resultado de hacer *transfer-learning* y *fine-tuning* de la red *VGG-19*.

Para disponer de una evaluación menos dependiente de sesgos y variaciones estadísticas realizamos una validación cruzada de este modelo, utilizando para ello los 10 *folds* predeterminados de los que disponemos. La tabla 6 muestra la media y desviación estándar de cada una de las métricas estudiadas a lo largo de todos los *folds*.

Conjunto	**Loss**	**Accuracy**	**F1-Score**
Entrenamiento	$0{,}13 \pm 0{,}05$	$0{,}95 \pm 0{,}02$	$0{,}95 \pm 0{,}02$
Validación	$0{,}26 \pm 0{,}06$	$0{,}92 \pm 0{,}02$	$0{,}92 \pm 0{,}02$
Test	$0{,}28 \pm 0{,}07$	$0{,}90 \pm 0{,}02$	$0{,}90 \pm 0{,}03$

Tabla 6: Resultados de la validación cruzada

Como podemos ver los resultados son ligeramente peores que los obtenidos en el *fold0*, lo cual nos indica que quizás la configuración con la que hemos entrenado el modelo está demasiado afinada para este *fold*. Aún así, con estos resultados el Modelo 5 es un modelo robusto y con muy buenos resultados, con *accuracy* y *F1-Score* sólo ligeramente inferior a los obtenidos por los autores del artículo para la misma arquitectura (ambos iguales a $0{,}91 \pm 0{,}03$). Es importante considerar también en esta comparación que nosotros sólo hemos hecho *fine-tuning* de los últimos dos bloques de la *VGG-19*, y durante un máximo de 150 épocas (que además en la práctica no ha superado las 50 épocas), mientras que en el artículo han modificado la red completa y la han entrenado durante 250 épocas, por lo que nuestro entrenamiento ha sido computacionalmente menos costoso.

Reducción de Falsos Negativos

El Glaucoma es una patología muy grave, por lo que es importante reducir los falsos negativos del modelo, como ya hemos comentado en la sección 2. Hay diversas estrategias que podríamos seguir para tratar de reducirlos al máximo:

- Podemos reducir el umbral de probabilidad para considerar un caso como positivo [11]. De esta manera seremos menos exigentes a la hora de predecir un caso como positivo, por lo que se reducirá el número de falsos negativos.
- Podemos utilizar una función de pérdida con pesos [16, Sección *Class weights*] que de mayor peso a los ejemplos de la clase positiva, y así "le preste más atención" a esta clase, reduciendo el número de falsos negativos.
- Podemos hacer *oversampling* de la clase que queramos reconocer mejor [16, Sección *Oversampling*], en nuestro caso la clase positiva.

La parte negativa de todas estas estrategias es el aumento de falsos negativos, inevitable debido a la balanza existente entre falsos positivos y falsos negativos.

En nuestro caso las 3 estrategias podrían funcionar, si bien debido a la escasez de datos la primera estrategia se debe aplicar con cuidado. En caso de querer utilizar esta estrategia, podría ser también interesante cambiar el criterio según el cual estamos escogiendo el mejor modelo, como quizás el *AUC* bajo la curva *precision-recall*, de manera que garantizásemos un mayor abanico de posibilidades a la hora de escoger el umbral. Utilizada junto con *data-augmentation* la tercera estrategia podría dar buenos resultados. Por lo tanto, si quisiese minimizar aún más el número de falsos negativos trataría de combinar estas dos estrategias.

4. Análisis crítico

4.1. Diseño de los *folds* para la validación cruzada

Para la realización de la práctica se nos han entregado las *folds* preparadas para el entrenamiento. El principal problema de los *folds* entregados es la variabilidad de la proporción de casos anormales en el conjunto de test a lo largo de los diferentes *folds* y su diferencia para con la proporción en entrenamiento y validación, tal y como hemos comentando en la Sección 1.

En caso de que se nos hubiesen dado las 1707 imágenes directamente, y tuviese que diseñar yo los *folds*, habría aplicado un diseño de *stratified K-fold cross-validation*. De esta manera podríamos garantizar que la proporción de casos positivos y casos negativos se mantiene similar en cada *fold* y entre los conjuntos de entrenamiento, validación y evaluación de cada iteración, lo cual suele mejorar la capacidad de generalización de los modelos entrenados.

Además, como nuestros datos provienen de bases de datos diferentes, también trataría de que en cada *fold* hubiese una proporción similar de imágenes de cada base de datos. Como comentan en el artículo [1] la capacidad de generalización cae en gran medida cuando se trata de predecir imágenes de una base de datos no vista en el entrenamiento, por lo que garantizar que durante el entrenamiento siempre se viesen imágenes de todas las bases de datos aumentará la capacidad de generalización de los modelos obtenidos.

4.2. Resultados y conclusiones

Comenzamos por los resultados, de los cuales las conclusiones dependen. Como ya hemos comentado en la Sección 3, los resultados que hemos obtenido con el Modelo 5, donde hemos obtenido un *F1-Score* de $0{,}90 \pm 0{,}03$ mediante validación cruzada sobre los datos de test, son sólo ligeramente inferiores a los obtenidos en [1], por lo que los podemos considerar muy buenos, aunque no sean punteros. Por otra parte, hay puntos de mejora, como pueden ser el diseño estratificado en la validación cruzada y la reducción de falsos negativos. Además es probable que un refinamiento del *data-augmentation* y un mayor *fine-tuning* pudiesen mejorar aún un poco los resultados. Por último, y aunque esto ya no está en nuestras manos, tal y como comentan en el artículo, la obtención de más imágenes y con mayor calidad mejoraría sin duda los resultados.

Siguiendo con las conclusiones, la principal es la sorprendente potencia de las técnicas de *transfer-learning* y *fine-tuning*, que sin duda alguna tendré muy presentes en el futuro. Aún siendo la nueva tarea difícil y nuestros datos pocos y no muy similares a los del conjunto *ImageNet*, mediante estas técnicas hemos conseguido obtener modelos con muy buenos resultados en un tiempo (relativamente) reducido. Respecto al *fine-tuning* concretamente creo que es muy importante ir descongelando progresivamente las capas de los modelos, dado que de hacerlo de golpe, como al pasar del Modelo 2 al 3, cuesta mucho controlar el tamaño de los gradientes para que el modelo siga mejorando. También es sorprendente el impacto del *data-augmentation*, visible en el Modelo 5, que reduce en gran medida el *overfitting* del modelo, permitiendo que aprenda otros patrones que sí generalicen.

La otra cara de la moneda es la considerable cantidad de horas pasadas tuneando los hiperparámetros de los modelos, debido al gran impacto que los pequeños cambios de estos tienen. Y esto sin la necesidad de diseñar la arquitectura de los modelos a utilizar, un reto por si mismo para facilitar el cual he llegado a la conclusión de que es muy necesario investigar de inicio que arquitecturas se han utilizado previamente en tareas similares. Por último, la inmensidad de configuraciones posibles de una red neuronal pueden convertir su diseño y entrenamiento en un esfuerzo titánico, lo cual obliga a valorar cuidadosamente cuando los resultados obtenidos son los suficientemente buenos, y a medir con cuidado si merece la pena invertir una mayor cantidad de horas y recursos en probar alguna idea nueva.

Referencias

[1] Andres Diaz-Pinto, Sandra Morales, Valery Naranjo, Thomas Köhler, Jose M. Mossi, and Amparo Navea. Cnns for automatic glaucoma assessment using fundus images: an extensive validation. *BioMedical Engineering OnLine*, 18(1):29, 2019. doi: 10.1186/s12938-019-0649-y. URL `https://doi.org/10.1186/s12938-019-0649-y`.

[2] Jordi de la Torre UOC. Datos Practica DL UOC 2022. URL `https://www.kaggle.com/datasets/jordidelatorreuoc/practica-dl-uoc-2022?select=practica_DL_UOC_2022`.

[3] Attila Budai, Rüdiger Bock, Andreas Maier, Joachim Hornegger, and Georg Michelson. Robust vessel segmentation in fundus images. *International journal of biomedical imaging*, 2013, 2013.

[4] Jayanthi Sivaswamy, S. R. Krishnadas, Gopal Datt Joshi, Madhulika Jain, and A. Ujjwaft Syed Tabish. Drishti-gs: Retinal image dataset for optic nerve head(onh) segmentation. In *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pages 53–56, 2014. doi: 10.1109/ISBI.2014.6867807.

[5] F. Fumero, S. Alayon, J. L. Sanchez, J. Sigut, and M. Gonzalez-Hernandez. Rim-one: An open retinal image database for optic nerve evaluation. In *2011 24th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 1–6, 2011. doi: 10.1109/CBMS.2011.5999143.

[6] Qaisar Abbas. Glaucoma-deep: detection of glaucoma eye disease on retinal fundus images using deep learning. *Int J Adv Comput Sci Appl*, 8(6):41–5, 2017.

[7] OpenCV. Histogram Equalization tutorial. URL `https://docs.opencv.org/3.4/d4/d1b/tutorial_histogram_equalization.html`.

[8] Jordi Casas Roma. *Deep learning*. Editorial UOC, 2019.

[9] Google Machine Learning Crash Course. Classification: Precision and Recall. URL `https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall?hl=es_419`.

[10] Keras. Keras Applications. URL `https://keras.io/api/applications/`.

[11] Machine Learning Mastery. A Gentle Introduction to Threshold-Moving for Imbalanced Classification. URL `https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/`.

[12] Francois Chollet - Keras. Transfer learning & fine-tuning. URL `https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html`.

[13] For deep learning, does the statement "batch size should be no more than 32"; make sense from yann lecun? Cross Validated. URL `https://stats.stackexchange.com/q/355324`.

[14] Mingxing Tan - Google AI. EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling. URL `https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html`.

[15] Yixing Fu - Keras. Image classification via fine-tuning with EfficientNet. URL `https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/`.

[16] Tensorflow Tutorials. Classification on imbalanced data. URL `https://www.tensorflow.org/tutorials/structured_data/imbalanced_data`.