

Práctica 2: TRANSFERENCIA DE APRENDIZAJE Y EVALUACIÓN DE MODELOS

Álvaro González Sánchez
Modelos de Redes Neuronales

26 de mayo, 2025

Contents

1 Resumen ejecutivo de la práctica	2
2 Evaluar la red neuronal VGG16 entrenada con Imagenet para diagnosticar glaucoma	3
2.1 Determinar el rendimiento de la red VGG16 -con los pesos resultantes de su entrenamiento con Imagenet- para diagnosticar el glaucoma (si/no) en las retinografías de la base de datos ACRIMA	3
3 Transferencia de aprendizaje	4
3.1 Utilizar transferencia de aprendizaje para crear la nueva red VGG16-glaucoma congelando todas las capas convolucionales de VGG16 y entrenando la nueva red con las imágenes de ACRIMA; obtener y aportar descripción gráfica de la red resultante, curvas de aprendizaje y un índice de su rendimiento en ACRIMA.	4
3.1.1 Matriz de confusión	5
3.1.2 Curva AUC-ROC	6
3.1.3 Ejemplos aplicados de predicciones a la base de ACRIMA	7
3.2 Determinar el rendimiento de VGG16-g obtenida en 2.1 para el diagnóstico de glaucoma con las retinografías de Rim-One-R2	7
3.3 Indicar para 2 imágenes elegidas al azar cuáles son las zonas de cada imagen más determinantes del diagnóstico emitido por la red.	10
4 Estudio comparativo de rendimiento	11
4.1 Comparar el rendimiento de VGG16-glaucoma en el diagnóstico de glaucoma en ACRIMA y en Rim- One-R2, utilizando el área bajo la curva ROC como indicador del rendimiento	11
4.2 Ídem utilizando otro índice de rendimiento a elegir (justificando la elección)	12
4.3 Comentar la concordancia o discrepancia en las conclusiones obtenidas con ambos Índices de rendimiento	13
5 Apéndices con la versión de Visual Studio Code y copia de las redes de los ejercicios	15
5.1 Red VGG16 para 1.1	15
5.2 Red VGG16-g para 2.1	15

1 Resumen ejecutivo de la práctica

2 Evaluar la red neuronal VGG16 entrenada con Imagenet para diagnosticar glaucoma

2.1 Determinar el rendimiento de la red VGG16 -con los pesos resultantes de su entrenamiento con Imagenet- para diagnosticar el glaucoma (si/no) en las retinografías de la base de datos ACRIMA

Evaluando directamente la red VGG16 con los pesos del entenamiento de Imagenet, se obtiene lo siguiente:

```
loss, acc = model.evaluate(test_generator, verbose=1)
print(f"Precisión del modelo VGG16 preentrenado sin reentrenar: {acc * 100:.2f}%")

accuracy: 0.5011 - loss: 1.8244
Precisión del modelo VGG16 preentrenado sin reentrenar: 53.19%
```

Al evaluar directamente la red VGG16, sin ningún tipo de reentrenamiento, utilizando únicamente los pesos obtenidos a partir del entrenamiento original con *ImageNet*, se pretende comprobar si el modelo es capaz de distinguir entre retinografías de pacientes con glaucoma y retinografías normales de la base de datos ACRIMA. Esta prueba, aunque sencilla, resulta esencial para determinar si las características aprendidas en un dominio general como ImageNet son transferibles al ámbito específico del diagnóstico oftalmológico.

En primer lugar, tal y como puede observarse en el bloque de código que acompaña a la Figura ??, el modelo alcanza una precisión de únicamente **53.19%**. Este resultado, que se sitúa apenas por encima del azar, ya nos está indicando que el modelo no está siendo capaz de identificar patrones visuales relevantes para la detección del glaucoma. De hecho, el valor de `accuracy` es de 0.5011 y la función de pérdida (`loss`) se mantiene relativamente alta (1.8244), lo cual refuerza esta idea inicial.

No obstante, para entender mejor qué está ocurriendo en términos de clasificación, es útil observar la matriz de confusión que se presenta en la matriz de confusión. En ella se aprecia cómo el modelo tiende a confundir ambas clases: por ejemplo, 44 imágenes normales son clasificadas erróneamente como glaucomas, y 40 glaucomas son clasificadas como normales; es decir, existe un grado de confusión importante entre las clases, lo que pone de manifiesto la falta de capacidad discriminativa del modelo en este nuevo dominio:

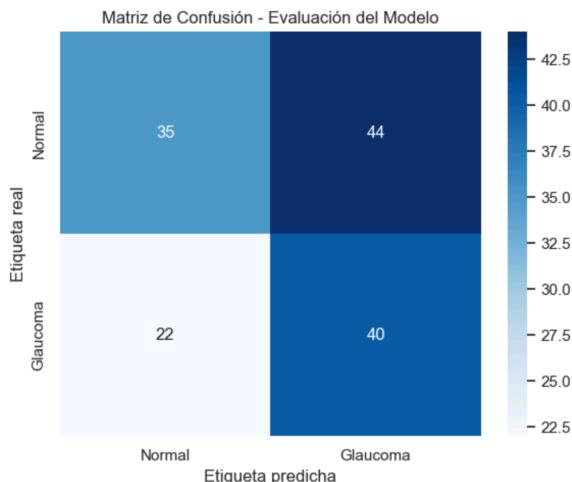


Figure 1: Matriz de confusión: Evaluación directa con ACRIMA desde Imagenet

Esta dificultad del modelo para adaptarse a las nuevas imágenes también se refleja en la curva ROC mostrada en la figura justo de abajo. En ella se observa una curva muy próxima a la diagonal aleatoria, lo cual se traduce en un área bajo la curva (*AUC*) de apenas **0.5274**, lo cual es un valor es prácticamente igual que se obtendría si el modelo realizase predicciones al azar, confirmando que no está siendo capaz de extraer características útiles para la tarea.

```
Área bajo la curva ROC (AUC): 0.5274
```

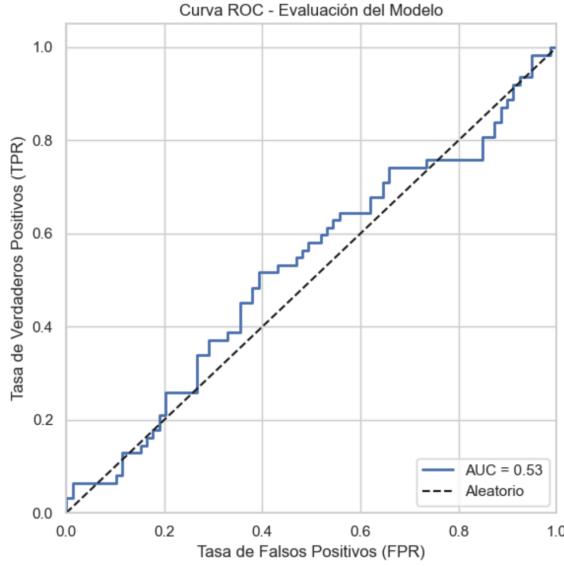


Figure 2: Curva ROC: Evaluación directa con ACRIMA desde Imagenet

En definitiva, estos resultados ponen de manifiesto una limitación importante: aunque VGG16 es una arquitectura potente, y aunque sus pesos preentrenados han sido ampliamente utilizados en muchos contextos, el hecho de **no reentrenar ni una sola de sus capas**, unido al cambio de dominio entre **ImageNet y ACRIMA**, hace que su capacidad de generalización se vea claramente comprometida. Por ende, el modelo no logra adaptarse a las particularidades concretas y únicas a nivel visual de las retinografías.

3 Transferencia de aprendizaje

3.1 Utilizar transferencia de aprendizaje para crear la nueva red VGG16-glaucoma congelando todas las capas convolucionales de VGG16 y entrenando la nueva red con las imágenes de ACRIMA; obtener y aportar descripción gráfica de la red resultante, curvas de aprendizaje y un índice de su rendimiento en ACRIMA.

Después de haber comprobado que la red VGG16 preentrenada sobre ImageNet no logra generalizar adecuadamente al contexto específico del diagnóstico de glaucoma en la base de datos ACRIMA, resulta razonable plantearse un enfoque más específico: el de la **transferencia de aprendizaje**. En particular, el objetivo ahora es mantener las capas convolucionales de VGG16 congeladas —es decir, sin modificar sus pesos originales— y entrenar únicamente una nueva cabeza de clasificación sobre las imágenes de ACRIMA.

Esta estrategia parte de la suposición de que las capas inferiores del modelo (ya entrenadas) contienen representaciones útiles para muchas tareas visuales, y que, por tanto, puede ser más eficaz ajustar solo las capas superiores para adaptarlas al nuevo dominio. De esta forma, se preservan los filtros generales aprendidos y se reduce tanto el coste computacional como el riesgo de sobreajuste, especialmente en contextos donde la cantidad de datos disponibles es limitada, como ocurre en muchas bases médicas.

El gráfico que se muestra a continuación refleja la evolución de la precisión del modelo a lo largo del proceso de entrenamiento, en el que únicamente se han ajustado los pesos de la nueva cabeza añadida, lo que permite no solo observar el grado de aprendizaje que está alcanzando el modelo, sino también detectar posibles problemas de sobreajuste o inestabilidad durante el proceso.

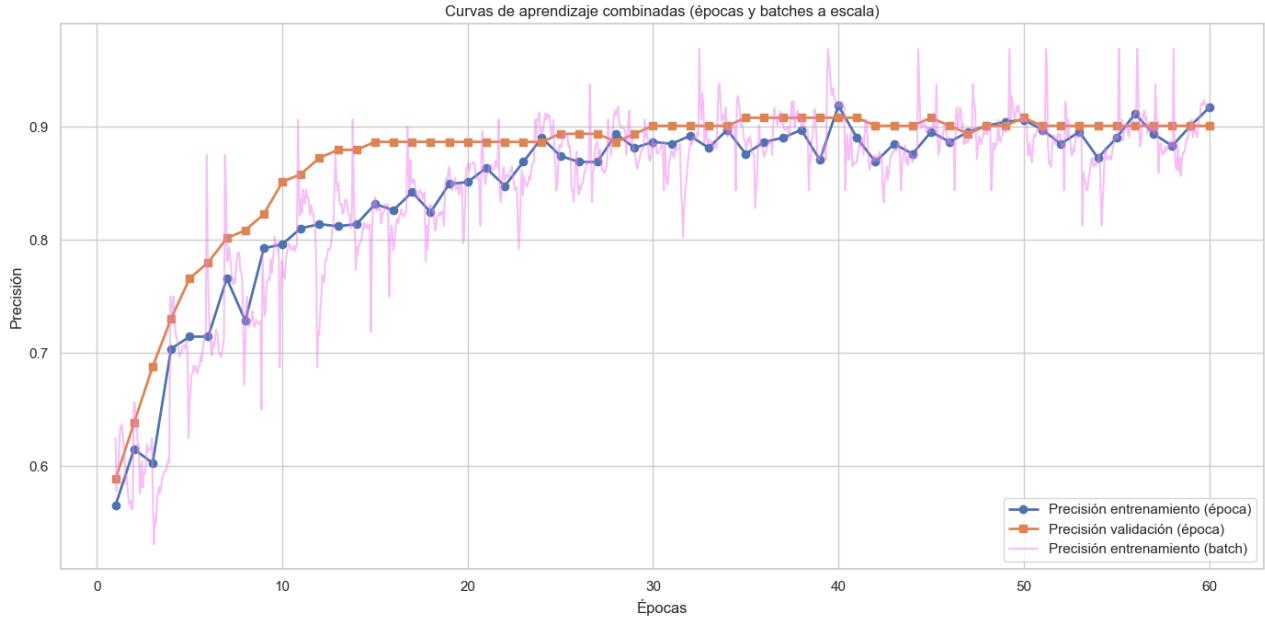


Figure 3: Red de curvas de aprendizaje. Entrenamiento con congelación de capas

El gráfico, como se observa, muestra la evolución de la **precisión del modelo** a lo largo del entrenamiento con **congelación de las capas convolucionales de la red VGG16**. Se representan tres curvas: la precisión por época sobre los datos de entrenamiento (línea azul con marcadores circulares), la precisión por época sobre el conjunto de validación (línea naranja con marcadores cuadrados) y, además, la precisión por batch a lo largo del entrenamiento (línea rosa), que refleja las oscilaciones internas durante el aprendizaje.

Como se puede observar, el modelo mejora de forma consistente en las primeras 5-15 épocas, alcanzando rápidamente valores de precisión en validación cercanos al 90%. Sin embargo, a partir de ahí, la curva de validación se estabiliza y se vuelve casi asintótica, con oscilaciones muy leves de los valores en el `val accuracy`, mientras que la de entrenamiento sigue una progresión más lenta pero continua, acercándose progresivamente a la misma línea. Este patrón es indicativo de un modelo que generaliza bien, sin signos evidentes de sobreajuste. Eso sí, la ligera diferencia entre la precisión de entrenamiento y validación (donde la valdación tiende a ser superior, aunque no mucho), se puede atribuir al uso de técnicas de regularización como `Dropout` y `data augmentation`, que introducen variabilidad durante el entrenamiento pero no afectan a la evaluación en validación.

La curva que representa la precisión batch a batch, muestra una alta variabilidad esperable en los primeros ciclos de aprendizaje, que se va reduciendo con el paso de las épocas a medida que el modelo se estabiliza, lo cual es también indicativo de una convergencia adecuada del proceso de entrenamiento.

3.1.1 Matriz de confusión

```

Normal clasificado como normal (TN): 68
Normal clasificado como glaucoma (FP): 11
Glaucoma clasificado como normal (FN): 3
Glaucoma clasificado como glaucoma (TP): 59
Accuracy: 0.90
Precisión (PPV): 0.84
Sensibilidad (TPR): 0.95
Especificidad (TNR): 0.86

```

Una vez finalizado el entrenamiento del modelo con congelación de capas, se procedió a evaluar su rendimiento sobre el conjunto de prueba. Esta evaluación incluyó tanto métricas numéricas clásicas como visualizaciones interpretables.

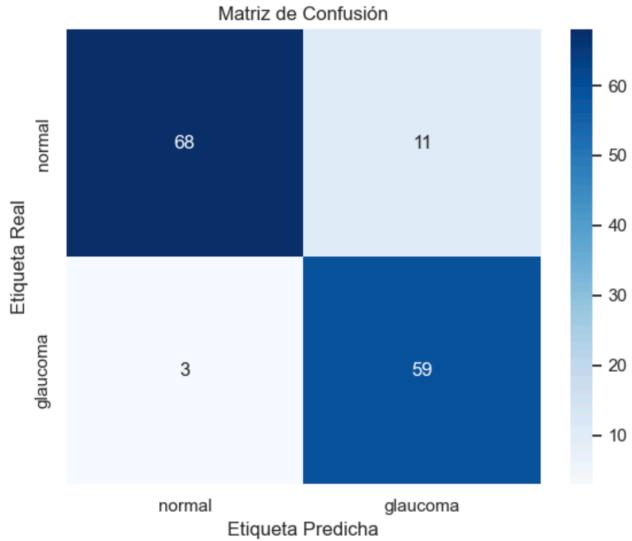


Figure 4: Matriz de confusión. Modelo reentrenado con capas congeladas

Los resultados mostraron una *Sensibilidad* ($TPR = 0.95$) y *Especificidad* ($TNR = 0.86$), siendo ambas elevadas, lo que indica que el modelo es capaz de detectar con gran fiabilidad tanto los casos positivos como los negativos. Sin embargo, se produjeron 11 falsos positivos (normales clasificados como glaucoma) y 3 falsos negativos (glaucomas no detectados), lo que implica que, si bien el rendimiento general es muy bueno, sigue siendo necesario tener en cuenta el posible impacto clínico de estos errores residuales.

3.1.2 Curva AUC-ROC

```
AUC-ROC: 0.9831
Umbral óptimo (Youden): 0.83
```

En este caso, el área bajo la curva (AUC) fue de 0.9831, lo que indica un rendimiento muy óptimo, pues demuestra que el modelo tiene una alta probabilidad de asignar una mayor probabilidad de “glaucoma” a un ojo realmente afectado que a uno sano. Por otro lado, el punto rojo sobre la curva indica el **umbral de decisión óptimo según el índice de Youden** (0.83), que maximiza la diferencia entre sensibilidad y tasa de falsos positivos, donde elegir el umbral adecuado puede depender de si se prioriza evitar falsos negativos (riesgo de no diagnosticar) o falsos positivos (sobrediagnóstico).

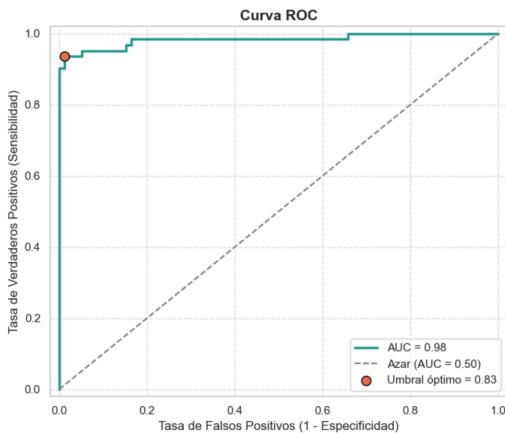


Figure 5: Curva ROC-AUC

3.1.3 Ejemplos aplicados de predicciones a la base de ACRIMA

A continuación se muestran algunos ejemplos prácticos de predicciones que el modelo final ha realizado sobre imágenes del conjunto de prueba perteneciente a la base de datos ACRIMA. Para ello, se utiliza la función:

```
display_predictions_from_generator(best_model, test_generator, class_names=["normal", "glaucoma"])
```

Que permite seleccionar algunas imágenes reales del generador `test_generator` (de forma aleatoria) y mostrar, junto a cada una, tanto la etiqueta verdadera como la predicción generada por el modelo (`best_model`). Las etiquetas se presentan en formato legible ("normal" o "glaucoma"), lo que facilita una evaluación rápida e intuitiva de la capacidad del modelo para distinguir entre casos normales y casos patológicos.



Figure 6: Predicciones de imágenes aleatorias del dataset de test de ACRIMA

Como se observa, el modelo ha acertado en prácticamente todos los casos, prediciendo correctamente la presencia de glaucoma. Solo se aprecia un error (marcado en rojo) en la última columna, donde el modelo clasifica como "normal" una imagen que en realidad corresponde a un caso de glaucoma. Este tipo de error, aunque minoritario, es importante tenerlo en cuenta en contextos clínicos, donde una predicción incorrecta podría tener consecuencias relevantes.

Así, en conjunto, los resultados visuales apoyan los indicadores numéricos obtenidos previamente (AUC y precisión en test), mostrando que el modelo es capaz de distinguir con alta fiabilidad entre ojos normales y con glaucoma.

3.2 Determinar el rendimiento de VGG16-g obtenida en 2.1 para el diagnóstico de glaucoma con las retinografías de Rim-One-R2

Se utilizó la siguiente base de datos de **Rim-One-R2**:

```
import os

# Rutas originales dentro de mi ordenador
```

```

base_path = '/Users/alvarogonzalez/Downloads/RIMONE-db-r2' # ESTA ES LA BASE DE DATOS, "RIMONE-db-
r2"
normal_path = os.path.join(base_path, 'Normal')
glaucoma_path = os.path.join(base_path, 'Glaucoma')

# Función para listar imágenes .jpg en una carpeta
def listar_jpgs(directorio):
    return sorted([
        f for f in os.listdir(directorio)
        if f.lower().endswith('.jpg') and os.path.isfile(os.path.join(directorio, f))
    ])

# Listado de archivos que sean válidos
jpgs_normal = listar_jpgs(normal_path)
jpgs_glaucoma = listar_jpgs(glaucoma_path)

# Resultados
print(f"Imágenes .jpg en 'Normal': {len(jpgs_normal)}")
print(f"Imágenes .jpg en 'Glaucoma': {len(jpgs_glaucoma)}")

# Ejemplos de las imágenes para contrastar que se haya redireccionado bien la ruta:
print("\nEjemplos de archivos:")
print("Normal:", jpgs_normal[:5])
print("Glaucoma:", jpgs_glaucoma[:5])

```

Cargamos el mejor modelo y, directamente, evaluamos dicho modelo a cómo clasificaría en la nueva base de datos de RIM-ONE-R2:

```

from tensorflow.keras.models import load_model

# Cargar el mejor modelo guardado automáticamente por ModelCheckpoint
model = load_model("./checkpoints/best_model.keras")

# Evaluar en el conjunto RIM-ONE-r2
loss, accuracy = model.evaluate(rimone_generator, verbose=1)
print(f"Precisión en RIM-ONE-r2: {accuracy * 100:.2f}%")

```

Los resultados de clasificación del modelo sobre el conjunto RIM-ONE-r2 muestran un rendimiento claramente deficiente: a pesar de que el valor de *accuracy* alcanza un 51%, comparándolo con el azar, es prácticamente como contar con él directamente a la hora de clasificar, ocultando así la red un desequilibrio importante entre sensibilidad y especificidad.

Analizando la matriz de confusión se observa que el modelo tiende a clasificar muchas imágenes de glaucoma como si fueran normales (165 falsos negativos), mientras que detecta correctamente solo 90 casos verdaderos de glaucoma: esto es, una sensibilidad muy baja (0.35), lo cual es especialmente preocupante en el contexto clínico, donde los falsos negativos pueden tener consecuencias graves. Por otro lado, la especificidad es considerablemente mayor (0.70), lo que indica que el modelo se muestra **conservador**, prefiriendo etiquetar imágenes como “normales” aunque corra el riesgo de pasar por alto casos reales de la enfermedad. De hecho, se ve más abajo en las imágenes de clasificación directa de ejemplos cómo la red se equivoca muy frecuentemente.

En definitiva, este patrón de errores sugiere que **el modelo no se adapta bien a las características del conjunto RIM-ONE-r2** y que su capacidad para **identificar correctamente casos de glaucoma en este dominio es limitada** y como si se tratase del azar, junto con el elevado número de falsos negativos, que refuerza la idea de una **falta de generalización**.

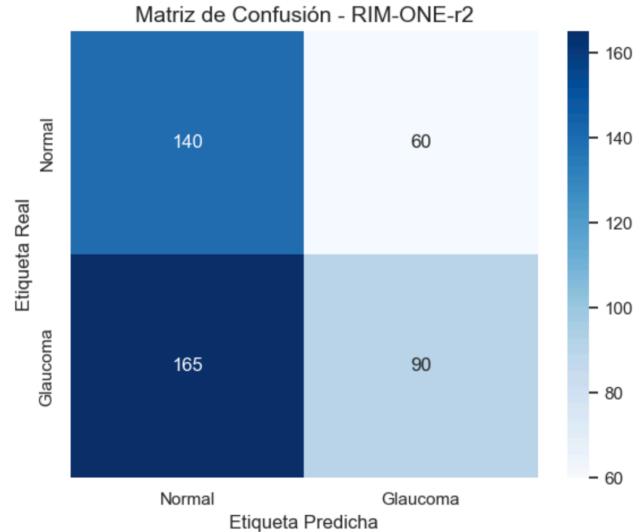


Figure 7: Matriz de confusión RIM-ONE-R

```
Normal clasificado como normal (TN): 140
Normal clasificado como glaucoma (FP): 60
Glaucoma clasificado como normal (FN): 165
Glaucoma clasificado como glaucoma (TP): 90
```

```
Accuracy: 0.51
Precisión (PPV): 0.60
Sensibilidad (TPR / Recall): 0.35
Especificidad (TNR): 0.70
```



Figure 8: Clasificación mínimamente mejor que el azar

Y también podemos obtener la siguiente curva ROC individual solamente de la evaluación directa del modelo de red neuronal aplicado a RIM, que refuerza totalmente los argumentos que se acaban de presentar

en contra de la viabilidad del modelo a los datos:

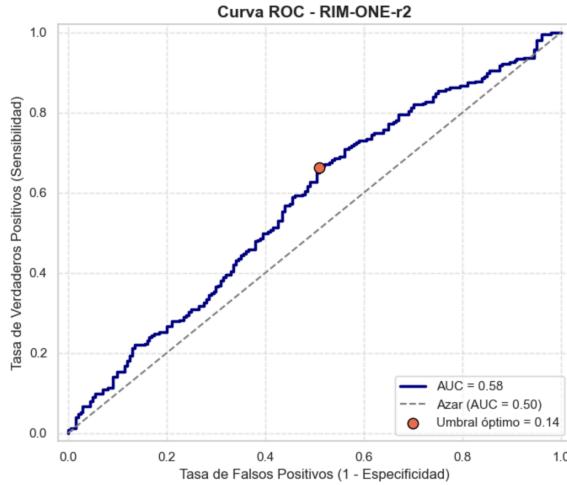


Figure 9: Curva ROC individual aplicado a RIM-ONE-R2

```
AUC-ROC RIM-ONE-r2: 0.5755
Umbral óptimo (Youden): 0.14
```

Como se observa, el valor del AUC-ROC en RIM-ONE-r2 es de 0.5755, lo que indica que el modelo apenas supera el rendimiento esperado por azar, mientras que el umbral óptimo calculado mediante el índice de Youden se sitúa en 0.14, lo que sugiere que, para maximizar la sensibilidad y especificidad combinadas, es necesario emplear un umbral muy bajo, probablemente debido a la fuerte descompensación en las predicciones originales del modelo.

3.3 Indicar para 2 imágenes elegidas al azar cuáles son las zonas de cada imagen más determinantes del diagnóstico emitido por la red.

2 imágenes al azar de no diagnóstico: Normal

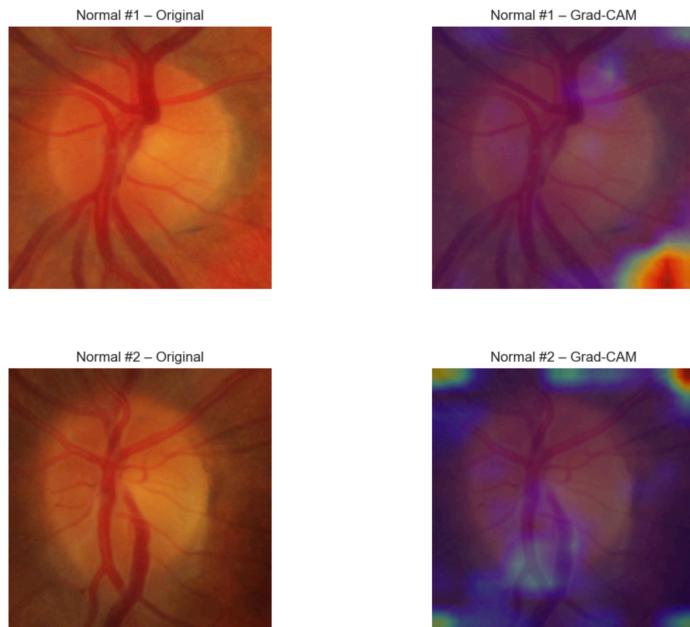


Figure 10: Zonas clave (heatmap) en Normal

2 imágenes al azar de diagnóstico: Glaucoma

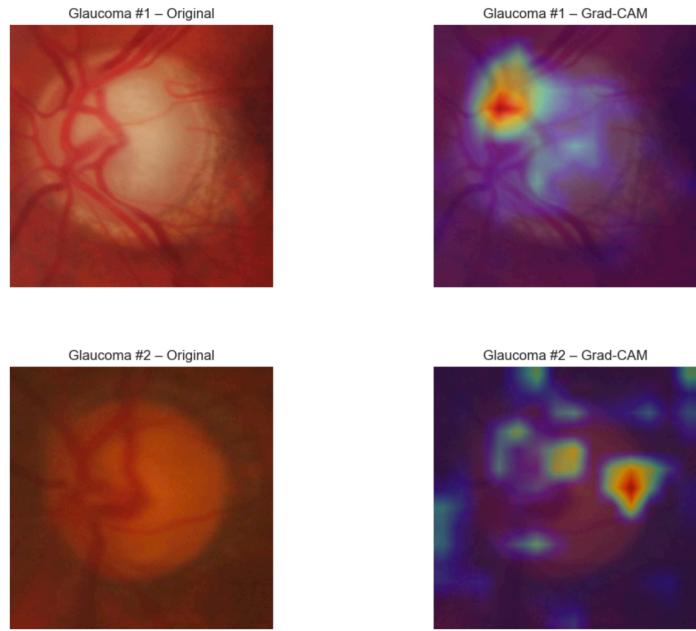


Figure 11: Zonas clave (heatmap) en Glaucoma

Lo destacable en este apartado es que, en las imágenes etiquetadas como “Normal”, las zonas resaltadas tienden a distribuirse de forma más difusa en torno al nervio óptico, sin concentraciones muy marcadas. En cambio, en las imágenes clasificadas como “Glaucoma”, se observan activaciones mucho más localizadas, especialmente alrededor del centro del ojo.

4 Estudio comparativo de rendimiento

4.1 Comparar el rendimiento de VGG16-glaucoma en el diagnóstico de glaucoma en ACRIMA y en Rim- One-R2, utilizando el área bajo la curva ROC como indicador del rendimiento

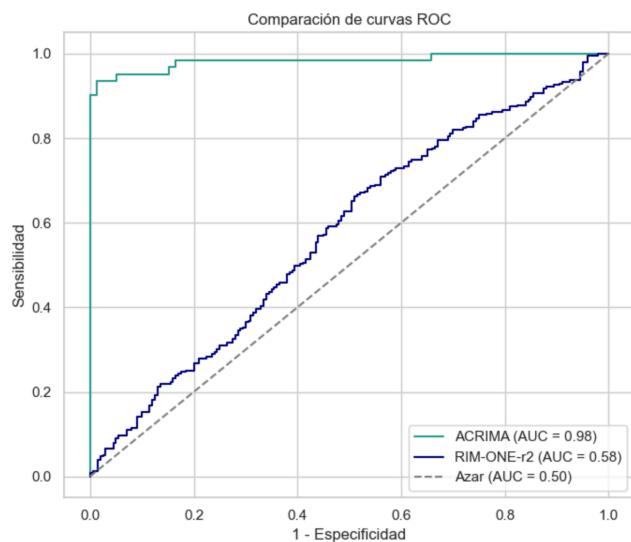


Figure 12: Enter Caption

```
AUC en ACRIMA: 0.9831
AUC en RIM-ONE-r2: 0.5755
```

Al comparar juntas ambas curvas ROC, tanto las de la red VGG16 con ACRIMA como con RIM-ONE-r2, se observan resultados claramente muy distintos. En ACRIMA, el modelo alcanza un valor de AUC de 0.9831, lo cual refleja un rendimiento excelente, prácticamente perfecto. Sin embargo, en el conjunto RIM-ONE-r2, el AUC desciende hasta 0.5755, apenas por encima del azar ($AUC = 0.5$), lo que indica que el modelo apenas es capaz de distinguir entre las clases en este segundo dominio más de lo que lo haría el propio azar prácticamente.

Esta discrepancia sugiere un problema claro de generalización: **el modelo ha aprendido a clasificar eficazmente las imágenes de ACRIMA, pero no es capaz de mantener ese rendimiento cuando se enfrenta a imágenes procedentes de otra base de datos.** Muy posibles causas pueden incluir el sobreajuste a las de ACRIMA, junto con diferencias en la calidad de imagen, el pre o postprocesamiento de las retinografías mediante métodos distintos, el etiquetado, las condiciones de adquisición entre ambos conjuntos, la propia diferencia entre las imágenes de base o bien por utilizar instrumentos distintos. En cualquier caso, los resultados dejan claro que el modelo, tal como está entrenado, **no puede considerarse robusto frente a cambios en el dominio de entrada;** esto es, **no generaliza bien.**

4.2 Ídem utilizando otro índice de rendimiento a elegir (justificando la elección)

Además del área bajo la curva ROC, un índice complementario especialmente útil para evaluar el rendimiento en clasificación binaria es el *Matthews Correlation Coefficient* (MCC). Este coeficiente puede interpretarse como una medida de la correlación entre las predicciones del modelo y las etiquetas verdaderas, integrando en su cálculo todos los elementos de la matriz de confusión: verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN).

La utilidad del MCC radica en que proporciona una evaluación más equilibrada del rendimiento del clasificador, incluso en situaciones de desbalance entre clases, algo común en problemas clínicos como el diagnóstico del glaucoma. A diferencia del *accuracy*, que puede ser engañoso si una clase es mucho más frecuente que la otra, el MCC penaliza adecuadamente tanto los errores por omisión como por comisión. Su rango de valores va de -1 (clasificación completamente errónea) a $+1$ (clasificación perfecta), siendo 0 indicativo de un rendimiento equivalente al azar. En este contexto, por ende, el MCC permite valorar de forma más exigente si el modelo es verdaderamente capaz de distinguir entre ojos sanos y con glaucoma. Tras calcularlo, se obtuvieron los siguientes resultados an nivel descriptivo y gráfico:

```
MCC ACRIMA @0.5 = 0.806
MCC RIM-ONE @0.5 = 0.056
```

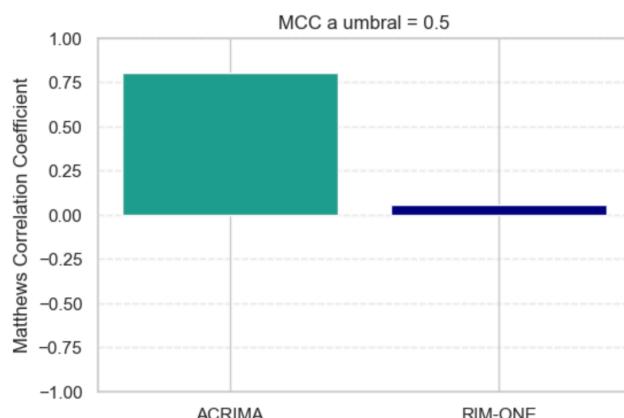


Figure 13: Enter Caption

En la Figura 13, se muestra el valor del MCC en ambos casos al utilizar un umbral fijo de 0.5. En este caso,

se observa un valor alto en ACRIMA ($MCC = 0.806$), lo que indica una predicción robusta en ese conjunto. En cambio, el valor muy cercano al 0 absoluto en RIM-ONE-r2 ($MCC = 0.056$) sugiere que no solamente el modelo no es capaz de superar una predicción mejor que el azar, sino que no generaliza bien a este segundo dominio, algo que el MCC capta con mayor sensibilidad que otros indicadores como el *accuracy*.

Por otro lado, la Figura 14 muestra **cómo evoluciona el MCC** al variar el **umbral de decisión**. En el caso de ACRIMA, se observa una evolución amplia de valores que son, en todo los casos, altos, lo que indica que el modelo **mantiene un buen rendimiento en un rango considerable de umbrales**. Sin embargo, en RIM-ONE-r2, el MCC se mantiene en torno a cero sin mostrar mejoras destacables, lo que refuerza la idea de que **el modelo no se adapta bien a este segundo conjunto**.

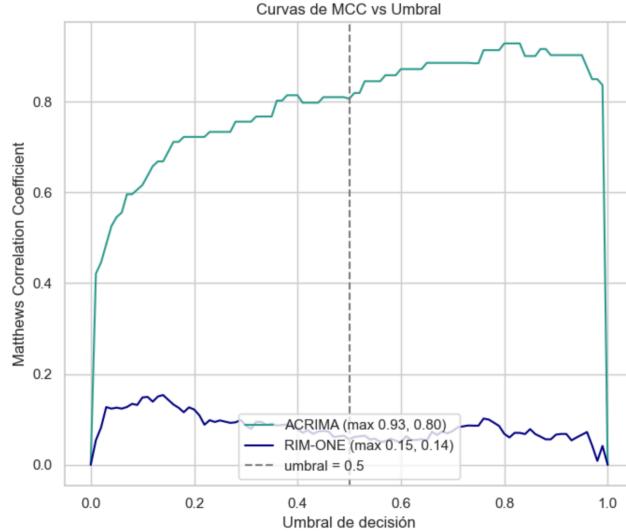


Figure 14: Enter Caption

Por tanto, el uso del MCC como índice alternativo está plenamente justificado, ya que ofrece una medida global del rendimiento del clasificador que es resistente al desbalance de clases y refleja de forma más precisa la calidad real de las predicciones del modelo.

4.3 Comentar la concordancia o discrepancia en las conclusiones obtenidas con ambos Índices de rendimiento

Ambos índices de rendimiento, tanto el área bajo la curva ROC (AUC) como el coeficiente de correlación de Matthews (MCC), apuntan en la misma dirección y permiten extraer conclusiones claramente similares: los dos reflejan un **rendimiento muy bueno del modelo cuando se evalúa sobre las imágenes del conjunto ACRIMA**, mientras que ambos muestran un **descenso notable cuando se aplica el mismo modelo a las imágenes de RIM-ONE-r2**. Esta coincidencia refuerza la idea de que el modelo es eficaz únicamente en el conjunto con el que ha sido entrenado o en uno similar, pero **no logra generalizar su capacidad de discriminación a nuevos datos provenientes de una base distinta**, sea por el motivo que sea, como se ha comentado antes.

Además, aunque ambos indicadores captan esta diferencia de rendimiento, el MCC lo hace con una mayor sensibilidad, ya que no solo señala una pérdida de precisión en RIM-ONE-r2, sino que muestra valores próximos a cero, lo que sugiere que las predicciones pueden ser equivalentes, como se ha comentado, al azar. Esto permite matizar aún más la interpretación y constatar que el modelo no solo baja de rendimiento, sino que puede volverse contraproducente en contextos distintos a los que conoce.

En definitiva, la lectura conjunta de ambos indicadores permite concluir que las capacidades del modelo para detectar glaucoma se restringen al dominio específico del conjunto ACRIMA. Esta conclusión es coherente y robusta, ya que se sostiene independientemente del índice de evaluación empleado, y pone de relieve la

necesidad de validar los modelos sobre bases heterogéneas antes de asumir que sus resultados son aplicables de forma generalizada.

Este tipo de situaciones pone de manifiesto un problema recurrente en el aprendizaje automático aplicado al ámbito clínico: la **falta de generalización fuera del dominio de entrenamiento**. Es frecuente que un modelo se comporte de forma excelente en los datos con los que fue entrenado y validado, pero que pierda completamente su utilidad cuando se enfrenta a imágenes con ligeras diferencias en calidad, resolución, instrumentos de captura o incluso criterios de etiquetado. Por ello, más allá de alcanzar buenos resultados métricos en una única base, es fundamental adoptar una visión crítica y cauta sobre su aplicabilidad real, y priorizar el uso de bases variadas y representativas en todas las fases del desarrollo.

5 Apéndices con la versión de Visual Studio Code y copia de las redes de los ejercicios

La versión utilizada de Visual Studio Code para macOS (chip M2) es:

```
Version: 1.100.0 (Universal)
Commit: 19e0f9e681ecb8e5c09d8784acaa601316ca4571
Date: 2025-05-07T12:48:53.763Z
Electron: 34.5.1
ElectronBuildId: 11369351
Chromium: 132.0.6834.210
Node.js: 20.19.0
V8: 13.2.152.41-electron.0
OS: Darwin arm64 24.5.0
```

Se ha usado **Python 3.9.18** en un entorno de *keras env*.

5.1 Red VGG16 para 1.1

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.optimizers import Adam

# 1. Cargar VGG16 sin la parte superior, con pesos de ImageNet
base_model = VGG16(include_top=False, weights='imagenet', input_shape=(224, 224, 3))

# 2. Congelar TODAS las capas
for layer in base_model.layers:
    layer.trainable = False # Esto se hace para usar la red como extracto de características fijo

# 3. Añadir una cabeza de clasificación mínima
x = base_model.output
x = Flatten()(x)
output = Dense(1, activation='sigmoid')(x) # Se aplana la salida y se conecta a una única neurona
    con activación sigmoide: clasificación binaria

model = Model(inputs=base_model.input, outputs=output)

# 4. Compilar (pero NO entrenar)
model.compile(optimizer=Adam(1e-4), loss='binary_crossentropy', metrics=['accuracy'])
```

5.2 Red VGG16-g para 2.1

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, GlobalAveragePooling2D, BatchNormalization, Dense,
    Dropout
from tensorflow.keras.applications import VGG16
from tensorflow.keras.optimizers import Adam

# 1. Definir la entrada de imágenes
inputs = Input(shape=(224, 224, 3)) # Imágenes RGB de tamaño 224x224

# 2. Cargar VGG16 como extracto de características
base_model = VGG16(include_top=False, weights='imagenet', input_tensor=inputs)

# 3. Congelar todas las capas convolucionales
for layer in base_model.layers:
    layer.trainable = False # No se actualizan durante el entrenamiento
```

```

# 4. Añadir una cabeza densa personalizada
x = base_model.output
x = GlobalAveragePooling2D()(x) # Reduce el mapa de activaciones a un vector
x = BatchNormalization()(x) # Normaliza para acelerar el entrenamiento
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x) # Previene sobreajuste
x = Dense(64, activation='relu')(x)
x = Dropout(0.3)(x)
outputs = Dense(1, activation='sigmoid')(x) # Salida binaria como antes, para clasificación (
    glaucoma o normal)

# 5. Definir y compilar el modelo completo
model = Model(inputs=inputs, outputs=outputs)
model.compile(
    optimizer=Adam(1e-4),
    loss='binary_crossentropy',
    metrics=['accuracy'])
)

# 6. Mostrar resumen del modelo
model.summary()

```