Detección temprana de cáncer de piel mediante clasificador de imágenes basado en Inteligencia Artificial

David Martin Tinaquero

Máster Universitario en Ciencia de Datos
Medicina · Análisis de imágenes para determinar potenciales tumores

Tutor/a de TF
Albert Solé Ribalta
Profesor/a responsable de la asignatura
Carlos Luís Sánchez Bocanegra
Luís Fernández Luque

15 de enero de 2023

Universitat Oberta de Catalunya

Ficha del Trabajo Final

Título del trabajo:	Detección temprana de cáncer de piel mediante clasificador de imágenes basado en Inteligencia Artificial
Nombre del autor/a:	David Martin Tinaquero
Nombre del Tutor/a de TF:	Albert Solé Ribalta
Nombre del/de la PRA:	Carlos Luís Sánchez Bocanegra Luis Fernández Luque
Fecha de entrega:	01/2023
Titulación o programa:	Máster Universitario en Ciencia de Datos
Área del Trabajo Final:	Medicina · Análisis de imágenes para determinar potenciales tumores
Idioma del trabajo:	Castellano
Palabras clave	Cáncer de piel, inteligencia artificial, *deep learning*

Resumen del Trabajo

Detectar el cáncer de piel cuando se encuentra en sus etapas iniciales a menudo permite la posibilidad de contar con más opciones de tratamiento. En algunos casos de la enfermedad en etapa inicial surgen signos y síntomas, pero esto no siempre es así. En la actualidad, los métodos más extendidos para detectar el cáncer en sus etapas iniciales son el autoexamen y el examen realizado por un médico especialista (dermatólogo).

El primer método tiene como inconveniente que la mayoría de la ciudadanía no sabe distinguir una lesión cutánea benigna de una cancerosa. El inconveniente principal de la segunda alternativa son los largos tiempos de espera para ser atendido por un dermatólogo, ya que se encuentran entre los tres especialistas más demandados, superados tan solo por los traumatólogos y oftalmólogos.

Este trabajo tiene por objetivo diseñar e implementar una herramienta de diagnóstico temprano que pueda ser utilizada en la práctica clínica como herramienta de apoyo al médico, haciendo de asistente al diagnóstico para

optimizar tiempos, y también ayude a personas sin conocimientos en dermatología haciendo de asesor para la detección de tumores cancerosos durante el autochequeo de la piel.

El trabajo investiga la viabilidad de uso de *deep learning*, concretamente, redes neuronales convolucionales (CNN) y *transformer*, para la clasificación de lesiones cutáneas a partir de imágenes.

Abstract

Finding skin cancer early often opens up more treatment options. Signs and symptoms do occur in some cases of early-stage disease, but this is not always the case. Currently, the most widespread methods to detect cancer in its early stages are self-examination and examination by a specialist doctor (dermatologist).

The first method has the drawback that most citizens do not know how to distinguish a benign skin lesion from a cancerous one. The main drawback of the second alternative is the long waiting times to be seen by a dermatologist, since they are among the three most in-demand specialists, surpassed only by traumatologists and ophthalmologists.

The objective of this work is to design and implement an early diagnosis tool that can be used in clinical practice as a support tool for the doctor, acting as a diagnostic assistant to optimize times, and also help people without dermatology knowledge by acting as an advisor for the detection of cancerous tumors during the self-examination of the skin.

The work investigates the feasibility of using deep learning, specifically, convolutional and transformer neural networks, for the classification of skin lesions from images.

Índice

Lista de Figuras

Lista de Tablas

1. Introducción

1.1. Contexto y justificación del Trabajo

El cáncer de piel es el tipo de cáncer más común. En todo el mundo:

- Cada año se diagnostican más de 13 millones de casos
- Uno de cada tres cánceres que se diagnostican es cáncer de piel
- Más de 65.000 personas mueren cada año por culpa de cáncer de piel [1]

Detectar el cáncer de piel cuando se encuentra en sus etapas iniciales a menudo permite la posibilidad de contar con más opciones de tratamiento. En algunos casos de la enfermedad en etapa inicial surgen signos y síntomas, pero esto no siempre es así [2]. En la actualidad, los métodos más extendidos para detectar el cáncer en sus etapas iniciales son el autoexamen cutáneo y el examen llevado a cabo por un médico especialista. El primer método tiene como inconveniente que la mayoría de la población no sabe distinguir una lesión cutánea benigna de una cancerosa. El inconveniente principal de la segunda alternativa son los largos tiempos de espera para ser atendido por un dermatólogo, ya que se encuentran entre los tres especialistas con tiempos de espera más largos, superados tan solo por los traumatólogos y oftalmólogos.

El proyecto se realiza en colaboración con la iniciativa de promoción de la salud de la Agencia Sanitaria Costa del Sol, Soludable, y nace de la necesidad de optimizar los tiempos necesarios para diagnosticar las lesiones cutáneas en la práctica clínica, así como de la necesidad de dotar a la ciudadanía no especializada de una herramienta que actúe como asesor en el diagnóstico del autochequeo. Un tercer objetivo, fuera del alcance de este proyecto, sería ayudar a eficientar las colas de espera de los especialistas, priorizando los casos en los que un diagnóstico previo realizado por la herramienta hubiera detectado tumores cancerosos.

1.2. Objetivos del Trabajo

Este trabajo tiene por objetivo principal diseñar e implementar una herramienta de diagnóstico temprano que pueda ser utilizada en la práctica clínica como herramienta de apoyo al médico, haciendo de asistente al diagnóstico para optimizar tiempos, y también ayude a personas sin conocimientos en dermatología haciendo de asesor para la detección de tumores cancerosos durante el autochequeo de la piel.

Tratándose de un tema tan delicado como es el diagnóstico médico, es imperativo que la herramienta ofrezca una alta fiabilidad en los resultados de sus predicciones. Es esencial que su uso sea sencillo, sobre todo para que los posibles usuarios no especializados puedan utilizarla sin dificultades. Por último, tiene que ser capaz de clasificar la imagen en el menor de tiempo posible.

Para cumplir con el objetivo principal, el trabajo tendrá los siguientes objetivos parciales:

- Usar *deep learning* para clasificar imágenes de lesiones cutáneas en las diferentes clases benignas y cancerosas.
- Investigar en trabajos previos de otros autores las redes neuronales convolucionales y *transformer*, así como las diferentes técnicas para incrementar su desempeño.
- Investigar redes neuronales convolucionales y *transformer* en su aplicación para clasificar imágenes médicas.
- Investigar el estado del arte de las diferentes librerías usadas en *deep learning*, tales como Tensorflow, Keras y PyTorch, y seleccionar las mejores candidatas para este proyecto.
- Diseñar e implementar modelos neuronales convolucionales y *transformer* simples desde cero y/o ajustar redes preentrenadas, haciendo uso de transferencia de aprendizaje.
- Realizar la búsqueda de los hiperparámetros que devuelven mejores resultados para cada red y ajustarlas con ellos.
- Evaluar los modelos comparando las métricas más importantes.
- Comparar todos los modelos tratados y seleccionar los que mejor se ajusten a la problemática del proyecto.
- Investigar la forma de implementar el modelo clasificador de imágenes como servicio.
- Crear una inteligencia colectiva uniendo las mejores redes entrenadas.
- Diseñar e implementar el clasificador de imágenes como servicio web y/o API.

1.3. Motivación personal

Los aspectos que me motivan a realizar este trabajo son su vinculación con la inteligencia artificial, concretamente con el aprendizaje profundo, que es la especialidad de la informática que más me apasiona. En la actualidad, la IA está presente en muchas áreas, tales como el markéting digital o la educación, aportando mejoras significativas, pero considero que en el área de la medicina es donde las aportaciones de la IA pueden ser más significativas, ya que trata lo más importante para todos los seres humanos, su salud.

Cuando se habla de enfermedades graves a todos nos viene a la cabeza en primer lugar el cáncer, y aunque el cáncer de piel no es el más mortal, sí que es del que más casos hay en todo el mundo. Como en todos los cánceres, la detección temprana es clave para aumentar las posibilidades de tratamiento, con lo que la posibilidad de crear una herramienta que pueda ayudar a diagnosticar la enfermedad en su etapa más temprana es algo muy motivador para mí.

Como último motivo para realizar este trabajo está mi interés en ampliar el trabajo de investigación de la inteligencia artificial aplicada a la medicina en una tesis doctoral.

1.4. Impacto en sostenibilidad, ético-social y de diversidad

A continuación, se detallan los impactos positivos y/o negativos de este trabajo en las tres dimensiones de la competencia transversal UOC “Compromiso ético y global”:

I. **Sostenibilidad**. Este trabajo tiene un impacto positivo en el ODS 12 – *Responsible consumption and production*, en el sentido que la herramienta para clasificar las imágenes pretende optimizar los tiempos y recursos necesarios, como el consumo energético de dispositivos clínicos, empleados en la práctica clínica para diagnosticar las lesiones cutáneas. También tiene un impacto positivo con el ODS – 13 *Climate action*, ya que contribuye a reducir la degradación ambiental, evitando las emisiones de CO2 de los desplazamientos a los centros clínicos de un número importante de personas a las que la herramienta diagnostica que la lesión no es peligrosa.

II. **Comportamiento ético y responsabilidad social (RS)**. El trabajo podría tener un impacto negativo con el ODS – 2 *Zero hunger* ya que, si se consigue alargar la vida de las personas por la aplicación de tratamientos efectivos a tiempo gracias a la detección temprana de la herramienta, habrá más personas en el planeta a las que alimentar. No obstante, el impacto no será significativo,

comparado con otros como los conflictos provocados por los humanos, el cambio climático o las recesiones económicas.

III. **Diversidad (género entre otros) y derechos humanos**. Este proyecto no impacta positiva ni negativamente en esta dimensión, ya que la herramienta podrá ser utilizada por igual por cualquier género, raza, religión, orientación sexual, funcional, etnia, …, sin discriminación por ninguna de ellas.

1.5. Enfoque y método seguido

Para cumplir con el objetivo principal de este proyecto se sigue una estrategia de investigación aplicada, revisando trabajos previos de otros autores para ver la forma como han resuelto problemas similares, analizar la manera de trasladar esas soluciones a este proyecto y aplicarlas haciendo uso de las librerías de *deep learning* que mejor se adapten.

El lenguaje de programación que se utiliza en este proyecto es Python, por ser el mejor y el más utilizado en el campo del *deep learning* y en el del aprendizaje automático en general. La preparación del conjunto de datos, ajustes y entrenamientos de las redes neuronales implementadas se realizarán en Jupyter Notebooks ejecutándose en un equipo local (Anexo 1), ya que sus características técnicas son superiores a las minúsculas cuotas de cómputo (CPU y GPU), que ofrecen plataformas en la nube tales como Google Colab o Kaggle en sus versiones gratuitas.

Para el diseño e implementación de la herramienta, primero se seleccionará un conjunto de datos y se preparará para entrenar las redes neuronales. Se seleccionarán las redes neuronales candidatas y se entrenarán, previa búsqueda de los mejores hiperparámetros para cada una. Una vez entrenadas, se evaluará su desempeño midiendo las métricas más importantes y se compararán los resultados conseguidos por los modelos para seleccionar los que mejor se ajusten al problema, formando parte de una inteligencia colectiva. Por último, se implementará el clasificador de imágenes como aplicación de servicio web y/o API.

1.6. Planificación del trabajo

La planificación del trabajo se ha elaborado ajustándose a los hitos de las entregas parciales (PECs), final, presentación y, por último, la defensa pública del TFM, pero la entrega del producto y la versión final de la memoria se hará con la PEC 4.2 el 15/1/2023. A continuación, se muestra el cronograma con la planificación del proyecto y el diagrama de Gantt.

Título	Fecha inicio	Fecha fin	Duración (días)
PEC 1 · Definición y planificación	28/09/2022	09/10/2022	12
Planificación (Diagrama de Gantt)	28/09/2022	02/10/2022	5
Redacción de la parte proporcional de la memoria	03/10/2022	09/10/2022	7
PEC 2 · Estado del arte o análisis de mercado	10/10/2022	23/10/2022	14
Investigación de soluciones · Estado del arte	10/10/2022	19/10/2022	10
Redacción de la parte proporcional de la memoria	20/10/2022	23/10/2022	4
PEC 3 · Diseño e implementación del trabajo	24/10/2022	25/12/2022	63
Preparación del entorno de trabajo	24/10/2022	25/10/2022	2
Elección, análisis y preparación del conjunto de datos	26/10/2022	04/11/2022	10
Diseño de redes y elección de redes preentrenadas	05/11/2022	14/11/2022	10
Optimización de hiperparámetros y ajuste fino	15/11/2022	19/11/2022	5
Entrenamiento y evaluación de las redes	20/11/2022	03/12/2022	14
Puesta en servicio del clasificador de imágenes	04/12/2022	13/12/2022	10
Redacción de la parte proporcional de la memoria	14/12/2022	25/12/2022	12
PEC 4.1 · Redacción de la memoria (1ª entrega)	26/12/2022	08/01/2023	14
Redacción del borrador completo de la memoria	26/12/2022	08/01/2023	14
PEC 4.2 · Redacción de la memoria (entrega final)	09/01/2023	15/01/2023	7
Redacción de la versión final de la memoria	09/01/2023	15/01/2023	7
Preparación de la defensa del proyecto	16/01/2023	22/01/2023	7
Diseño de las diapositivas de la presentación	16/01/2023	20/01/2023	5
Grabación de la versión final de la memoria	21/01/2023	22/01/2023	2
Defensa pública	23/01/2023	03/02/2023	12
Diseño de las diapositivas de la presentación	23/01/2023	25/01/2023	3
Preparación de la defensa pública	26/01/2023	03/02/2023	9

Tabla 1. Tabla con el cronograma del proyecto

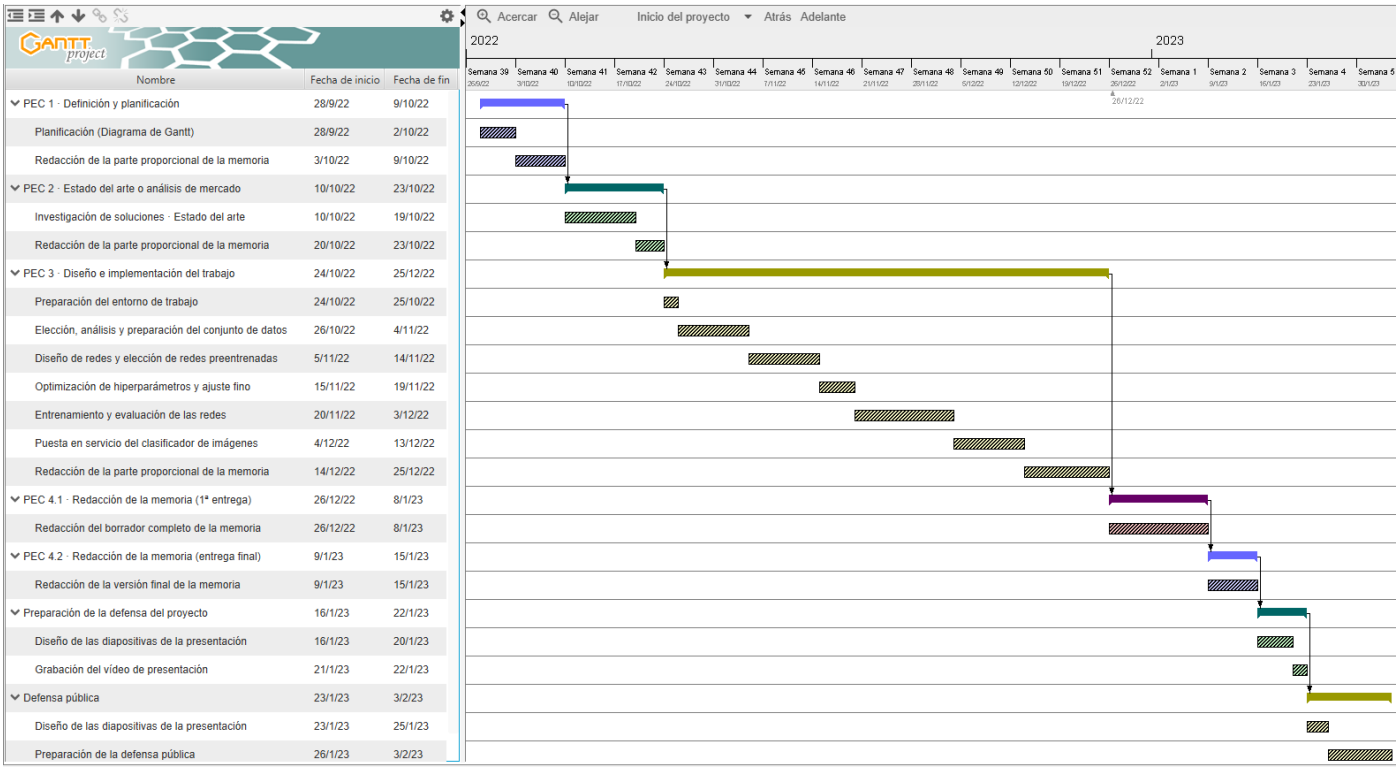


Figura 1. Diagrama de Gantt con la planificación del proyecto

1.7. Breve sumario de productos obtenidos

A continuación, se detallan los productos obtenidos en este proyecto:

- Código fuente de la aplicación web con el clasificador de imágenes para ejecutar en servidor web o remoto.
- Cuadernos Jupyter Notebooks en formato HTML con el código Python utilizado para preparar el conjunto de datos, entrenar y evaluar las redes.

1.8. Breve descripción de otros capítulos de la memoria

En el capítulo 2 se detallan los materiales y métodos utilizados para llevar a cabo este trabajo. Se empieza por la revisión sistemática de artículos científicos realizados por otros autores acerca del uso de técnicas de aprendizaje profundo para la identificación y clasificación de lesiones cutáneas mediante imágenes. En dichos trabajos se obtienen las redes preentrenadas que han conseguido mejores resultados, los mejores conjuntos de imágenes de lesiones cutáneas accesibles públicamente y las métricas más utilizadas para medir el desempeño de los modelos.

Se describen las características del conjunto de datos seleccionado para entrenar las redes neuronales y se detalla el origen y arquitectura de cada una de ellas. Se explica en detalle cómo se construye la inteligencia colectiva, detallando las matrices de pesos y de probabilidades de los nuevos diagnósticos, y la forma de cómo se realiza el consenso del diagnóstico con todas ellas a partir de la multiplicación de las dos matrices. Por último, se trata en detalle la aplicación web y la API.

En el capítulo 3 se entra más en el detalle del conjunto de imágenes utilizado para entrenar las redes neuronales, realizando una exploración, así como diferentes análisis de datos. Se explican las técnicas utilizadas para corregir el desbalanceo de datos y la falta de imágenes de ciertos tipos de lesiones cutáneas. Se detalla la sistemática utilizada para realizar los entrenamientos de cada una de las redes neuronales, así como las métricas utilizadas para evaluar su desempeño.

En este capítulo también se explica cómo se ha llevado a cabo la implementación de la inteligencia colectiva y se evalúa su desempeño, comparando los resultados obtenidos con los conseguidos por cada una de las redes neuronales individualmente. Se explica al detalle la forma como se ha implementado la aplicación web y la API, y se ilustra el uso de ambas. Para terminar este capítulo, se realiza una clasificación, haciendo uso de la API implementada, de 100 imágenes de melanoma de un conjunto de datos diferente al utilizado para entrenar y evaluar las redes neuronales, y se muestran los excelentes resultados obtenidos por la inteligencia colectiva.

En el capítulo 4 se detallan las conclusiones a las que se ha llegado como resultado de este trabajo y se proponen los trabajos futuros que no se han podido llevar a cabo pero que sería interesante abordar.

Por último en este trabajo, se encuentran los apartados de glosario, bibliografía y anexos.

2. Material y métodos

2.1. Diseño y criterio de búsqueda

Se diseña una revisión sistemática de la literatura para resolver las siguientes preguntas de investigación:

- ¿Cuáles son las arquitecturas de redes neuronales más utilizadas en la detección de cáncer de piel mediante la clasificación de imágenes?
- ¿Qué conjuntos de datos se han utilizado para entrenar, validar y evaluar las redes?
- ¿Son estos modelos realmente eficaces para la detección del cáncer de piel?
- ¿Qué métricas son las más utilizadas para medir el desempeño de las redes?

Se realiza una búsqueda bibliográfica el 10 de octubre de 2022 en las bases de datos de PubMed con todos los artículos publicados desde el 12 de febrero de 2016 hasta el 20 de septiembre de 2022, siguiendo las directrices de PRISMA, utilizando el algoritmo de búsqueda: *("skin cancer" OR "melanoma") AND ("detection" OR "classifier") AND ("deep learning" OR "transformer")*.

Se criban los artículos cuyo texto completo no se encuentra a libre disposición, mediante los filtros disponibles en PubMed. Con el objetivo de ver las redes más utilizadas recientemente, se acota el intervalo de artículos a los publicados en los años 2021 y 2022. Una vez descartados los artículos que no cumplen con los criterios mencionados, se procede a la lectura de los artículos restantes y se descartan aquellos que no están centrados en el cáncer de piel o en la detección de este, así como aquellos que no utilizan el aprendizaje profundo en la detección de cáncer de piel.

La búsqueda devuelve un total de 186 artículos que se convierten en 120, una vez excluidos 66 cuyo texto completo no está disponible de forma abierta. Éstos a su vez se reducen a 44, una vez filtradas las publicaciones realizadas en los años 2021 y 2022, y descartadas las que no están centradas en la detección del cáncer de piel o en la utilización de *deep learning*.

En los 44 artículos revisados los modelos de redes neuronales tratados demuestran su alta eficacia en la detección de cáncer de piel a través del análisis de imágenes. Todos los modelos tratados superaron el 85% de exactitud (*accuracy*) en la clasificación de lesiones cutáneas de nuevas imágenes.

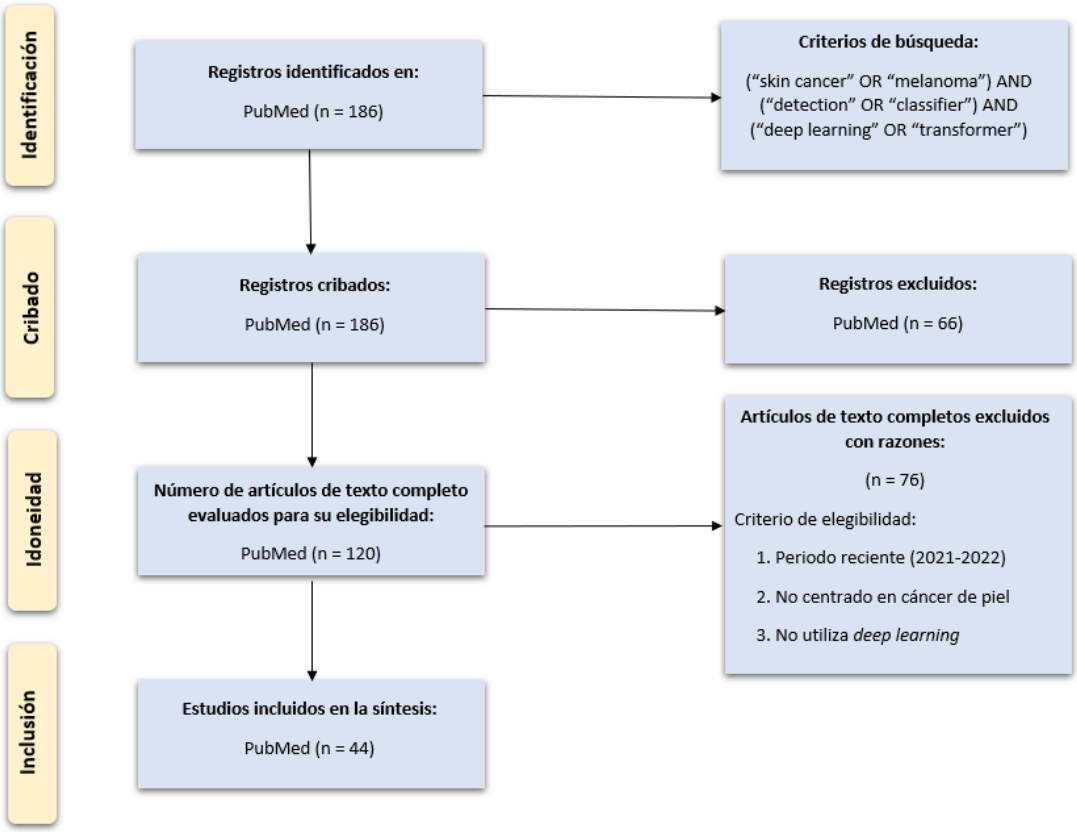


Figura 2. Diagrama de flujo PRISMA de artículos incluidos en la revisión

Con la intención de responder a las dos primeras preguntas objeto de la revisión, se confeccionan las tablas 2 y 3, con las arquitecturas de redes neuronales y los conjuntos de datos más utilizados en los artículos revisados.

Como se puede observar en la tabla 2, las familias de arquitecturas de redes neuronales, en su forma estándar o personalizadas, más utilizadas en los artículos revisados son: ResNet (28,05%), VGG (15,85%), GoogleLeNet (Inception) (14,63%), DenseNet (13,41%), Xception (9,76%), AlexNet (8,54%), MobileNet (7,32%) y EfficientNet (2,44%).

En cuanto a los conjuntos de datos más utilizados en los artículos revisados, hay dos fuentes principales: ISIC (*International Skin Imaging Collaboration*) [43] y HAM10000 [44] (*Human Against Machine*). Del conjunto HAM10000 solo hay una versión disponible con 10.015 imágenes y 7 etiquetas. Del conjunto ISIC, en cambio, hay múltiples versiones disponibles, la mayoría de ellas utilizadas en los desafíos propuestos por ISIC en los años comprendidos entre 2016 y 2020. El desafío del 2018 incluye el conjunto de datos HAM10000.

El conjunto de datos HAM10000 se ha utilizado en 22 de los artículos revisados (50%), mientras que los diferentes conjuntos de datos de ISIC aparecen en 27 artículos

(61,33%), siendo la versión del 2018 con 16.564 imágenes y 8 etiquetas la más utilizada (18,18%).

Familia	Arquitectura	Número de artículos	Artículos
ResNet	ResNet-50 (19) ResNet-18 (2) ResNet-32 (1) ResNet-50V2 (1) Resnet-101 (4) ResNet-152 (2)	23	[3][4][5][6][7][8][9][10][11][12][13][14][15] [16][17][18][19][20][21][22][23][24][25]
VGG	VGG16 (10) VGG19 (6)	13	[3][9][10][12][13][14][18][21][29][30][32][33][34]
GoogleLeNet	InceptionV3 (10) InceptionResNetV2 (5) InceptionV1 (2) InceptionV2 (2)	12	[3][4][5][6][7][11][14][17][28][29][30][31]
DenseNet	DenseNet-201 (6) DenseNet-121 (3) DenseNet-161 (1) DenseNet-169 (1)	11	[6][7][8][10][13][15][17][18][23][26][27]
Xception	Xception (8)	8	[6][7][8][9][13][14][30][35]
AlexNet	AlexNet (7)	7	[3][7][16][18][22][28][33]
MobileNet	MobileNetV2 (4) MobileNetV1 (2)	6	[6][7][8][11][14][26]
EfficienNet	EfficienNet B0 (2)	2	[6][11]

Tabla 2. Arquitecturas de redes CNN más utilizadas en los artículos revisados

La respuesta a la tercera pregunta planteada al inicio de la revisión, que pretende saber si las redes neuronales son eficaces en la detección del cáncer de piel, la obtenemos de la revisión realizada por Sarah Haggenmuller y varios autores más [45], publicada en octubre de 2021, en la que se analizaron 19 artículos previos en los que los clasificadores de redes neuronales convolucionales (CNN) mostraron un rendimiento superior o equivalente al de

los expertos. 11 estudios compararon la eficacia CNN con expertos en clasificación de imágenes dermatoscópicas, 6 en clasificación de imágenes clínicas y los 2 restantes en clasificación de WSI hispatológicos.

Conjunto de datos	Número de imágenes	Número de clases	Número de artículos	Artículos
HAM10000	10.015	7	22	[6][7][9][11][13][14][15][19][21][22][23][24] [27][28][30][32][33][35][37][38][41][42]
ISIC 2018	16.564	7	8	[21][22][23][24][30][33][37][38]
ISIC 2018	3.297	2	6	[5][8][9][12][26][37]
ISIC 2019	33.569	8	5	[3][4][30][39][41]
ISIC 2017	2000	2	4	[10][25][38][40]
ISIC 2016	1.279	2	2	[38][40]
ISIC 2020	44.108	2	2	[39][40]

Tabla 3. Conjuntos de datos más utilizados en los artículos revisados

El artículo de Tri-Cong Pham [17] de septiembre de 2021, explica como una CNN optimizada con lógica de mini lotes y función de pérdida superó en el diagnóstico dermatoscópico del melanoma a 157 dermatólogos de 12 hospitales de Alemania. Mientras que el mejor valor del área bajo la curva (AUC) obtenido por los dermatólogos más expertos alcanzó el 71,30%, la CNN consiguió un 93,80%.

La respuesta a la cuarta pregunta es casi unánime, ya que en la mayoría de los artículos se han utilizado las mismas métricas para evaluar el desempeño de las redes neuronales:

- ***Accuracy* (exactitud)**. Cantidad de predicciones que fueron correctas sobre el total. Se calcula como:

  (VP+VN) / (VP+FP+FN+VN)

- ***Precision* (precisión)**. Porcentaje de casos positivos detectados. Se calcula como:

VP / (VP+FP)

- ***Recall* (sensibilidad)**. Tasa de verdaderos positivos. Proporción de casos positivos correctamente identificados. En el área de la salud es la capacidad de poder detectar correctamente la enfermedad entre los enfermos. Se calcula como:

  VP / (VP+FN)

- ***Especificity* (especificidad)**. Tasa de verdaderos negativos. Proporción de casos negativos correctamente identificados. En el área de la medicina es la capacidad de poder identificar los casos de pacientes sanos entre todos los sanos. Se calcula como:

  VN / (VN+FP)

- **F1 *Score***. Resumen de la precisión y la sensibilidad. Se calcula como:

  (2 * *Precision* * *Recall*) / (*Precision* + *Recall*)

  La forma más utilizada para obtener las métricas anteriores es la matriz de confusión, donde:

  - **TP · *True positive* (verdadero positivo)**. El valor real es positivo y el clasificador predijo que era positivo.
  - **TN · *True negative* (verdadero negativo)**. El valor real es negativo y el clasificador predijo que era negativo.
  - **FN · *False negative* (falso negativo)**. El valor real es positivo y el clasificador predijo que era negativo.
  - **FP · *False positive* (falso positivo)**. El valor real es negativo y el clasificador predijo que era positivo.

- **AUC · *Area Under the ROC Curve* (Área bajo la curva ROC)**. Medida agregada del rendimiento en todos los umbrales de clasificación posibles. Se interpreta como la probabilidad de que el modelo clasifique un ejemplo positivo aleatorio más alto que un ejemplo negativo aleatorio. Conviene utilizar AUC porque:

  - **Es invariable con respecto a la escala**. Mide que tan bien se clasifican las predicciones, en lugar de sus valores absolutos.
  - **Es invariable con respecto al umbral de clasificación**. Mide la calidad de las predicciones del modelo, independientemente del umbral de clasificación elegido.

Matriz de confusión		Estimado por el modelo			
		Negativo (N)	*Positivo (P)*		
Real	*Negativo*	a: (TN)	b: (FP)		
	Positivo	c: (FN)	d: (TP)	**Precisión** ***("precision")*** Porcentaje predicciones positivas correctas:	d/(b+d)
		Sensibilidad, exhaustividad ***("Recall")*** Porcentaje casos positivos detectados	**Especifidad** ***(Specifity)*** Porcentaje casos negativos detectados	**Exactitud ("accuracy")** Porcentaje de predicciones correctas *(No sirve en datasets poco equilibrados)*	
		d/(d+c)	a/(a+b)	(a+d)/(a+b+c+d)	

Figura 3. Matriz de confusión con sus métricas (Fuente: 57)

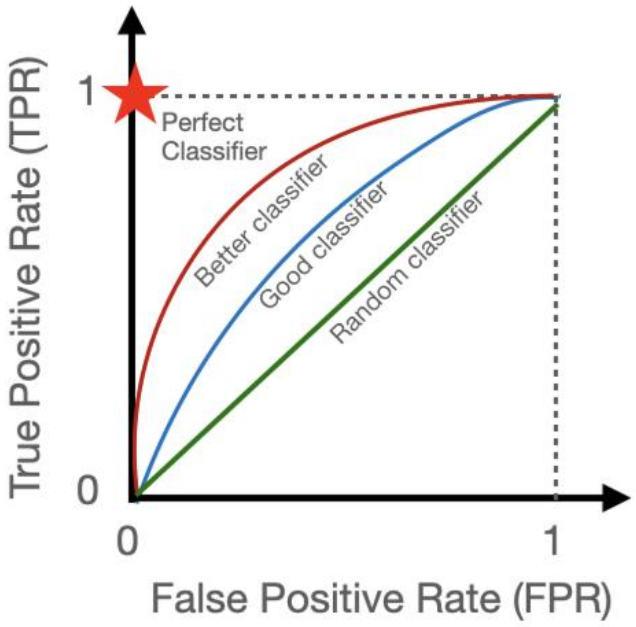


Figura 4. AUC ROC. Área bajo la curva (Fuente: 58)

2.2. Trabajos relacionados

Es una práctica habitual entre los autores de los artículos utilizar diferentes arquitecturas de redes neuronales y comparar los resultados obtenidos por ellas, como en el artículo de Ahmad Naeem [3], publicado en julio de 2022, en el que utiliza un modelo al que denomina SCDNet, que combina VGG16 con una CNN, y lo compara con los cuatro clasificadores preentrenados: Resnet50, InceptionV3, AlexNet y VGG19.

Otro artículo de referencia en el que se han comparado diferentes arquitecturas de redes neuronales es el del autor Satin Jain [14], publicado en diciembre de 2021, en el que se utilizaron las redes de transferencia de aprendizaje: VGG19, InceptionV3, InceptionResNetV2, ResNet50, Xception y MobileNet.

Hay autores que proponen estructuras de diseño propio, basadas en otros modelos de aprendizaje profundo, como en el trabajo de Nancy Girdhar [27], publicado en agosto de 2022, en el que utiliza un modelo personalizado denominado DenseNet II, basado en los modelos DenseNet, VGG16, InceptionV3 y ResNet.

A menudo, el autor selecciona únicamente la arquitectura que ha obtenido mejores resultados en las métricas evaluadas, pero no siempre es así, ya que hay autores que proponen modelos híbridos formados por varias redes trabajando juntas, ya sea combinando sus arquitecturas en una sola o tomando una decisión final en función de las decisiones individuales. Son estos últimos tipos los me parecen más interesantes, por su parecido con los comités de expertos que se realizan en el área de la medicina para obtener un diagnóstico consensuado ante ciertos casos de difícil diagnóstico. El hecho de que sean modelos con estructuras diferentes los asemeja aún más con la realidad, ya que los expertos también son diferentes, siendo mejores diagnosticando ciertas lesiones que otras.

El artículo realizado por Dan Popescu [7], publicado en junio de 2022, crea una inteligencia colectiva formada por las 9 redes neuronales preentrenadas: AlexNet, GoogLeNet, GoogLeNet-Places365, MobileNet-V2, Xception, ResNet-50, ResNet-101, InceptionResNet-V2 y DenseNet201. Estas 9 redes clasifican las nuevas imágenes individualmente y, a partir de una matriz de pesos, toma la decisión final y más precisa relacionada con la predicción basada en los pesos asociados de cada salida de la red.

Por último, mencionar el trabajo de Suliman Aladhadh [42], publicado en mayo de 2022, en el que se utiliza el modelo Vision Transformer, perteneciente al tipo de redes *transformer*, que aplican un conjunto en evolución de técnicas matemáticas, llamadas atención o atención propia, para detectar formas sutiles en que los elementos de datos en una serie se influencian y dependen entre sí. Hay poca literatura por el momento de redes *transformer* aplicadas a la detección de cáncer de piel, pero este tipo de redes han demostrado buen desempeño en el tratamiento de secuencias de texto, imágenes y vídeo.

2.3. Conjunto de datos

El conjunto de datos seleccionado para este trabajo es HAM10000, del Skin Cancer MNIST. Tal y como se ha visto en los artículos revisados, es el conjunto más utilizado y ha proporcionado buenos resultados para técnicas de diagnóstico de cáncer utilizando *deep learning*. Su número de licencia es CCBY-NC-SA 4.0.

El conjunto se ha recopilado de un archivo público de Kaggle [46]. Incluye 10.015 imágenes dermatoscópicas en formato JPEG con 8 bits de profundidad. Las imágenes fueron recolectadas durante un período de 20 años del Departamento de Dermatología de la Universidad Médica de Viena, Austria, y de la práctica de cáncer de piel de Cliff Rosendahl en Queensland, Australia.

Las imágenes están divididas en los 7 tipos de lesiones siguientes:

- Queratosis actínica y carcinoma intraepitelial/enfermedad de Bowen (akiec)
- Carcinoma de células basales (bcc)
- Lesiones benignas tipo queratosis (bkl)
- Dermatofibroma (df)
- Melanoma (mel)
- Nevos melanocíticos (nv)
- Lesiones vasculares (vasc)

El conjunto de datos está muy desequilibrado, con la mayor cantidad de imágenes de clase benigna: df (115 / 1,15%), vasc (142 / 1,42%), akiec (327 / 3,27%), bcc (514 / 5,13%), bkl (1.099 / 10,97%), melanoma (1.113 / 11,11%) y nv (6.705 / 66,95%).

Las técnicas de sobremuestreo (*oversampling*) para igualar la cantidad de imágenes de las clases y el aumento de datos (*data augmentation*) para añadir más variedad de imágenes, puede ser útil en la fase de entrenamiento, especialmente en las clases para las que hay pocas imágenes. Esto puede reducir el sobreajuste (*overfitting*) de las redes neuronales, mejorando su generalización.

Algunos métodos de augmento de datos considerados son:

- Rotación del eje vertical y horizontal
- Reflejo
- Zoom
- Desplazamiento de píxeles vertical y horizontal
- Adición de ruido
- Cortes de secciones

Las redes preentrenadas admiten imágenes con dimensiones específicas. Las más comunes en la actualidad son 224x224, aunque algunas redes requieren tamaños diferentes. Ciertas redes incorporan sus propias funciones de redimensión de imágenes, pero otras no, con lo que será necesario crear una función que adapte el tamaño de las imágenes para entrenar las redes de transferencia de aprendizaje, según los tamaños admitidos en sus capas de entrada.

2.4. Modelos *deep learning*

Este trabajo contempla 10 modelos de redes neuronales. Los modelos bien entrenados, trabajando como una inteligencia colectiva pueden ofrecer un alto grado de fiabilidad, superior a cuando lo hacen individualmente. Se considera 1 CNN creada *adhoc* y 9 modelos de transferencia de aprendizaje, constituidos por 7 CNN y 2 de redes *transformer*. En los modelos preentrenados se sustituyen las capas top de clasificación por otras capas *adhoc*. A continuación, se listan todos los modelos:

- **CNN adhoc**. Red neuronal convolucional creada específicamente con la siguiente estructura:

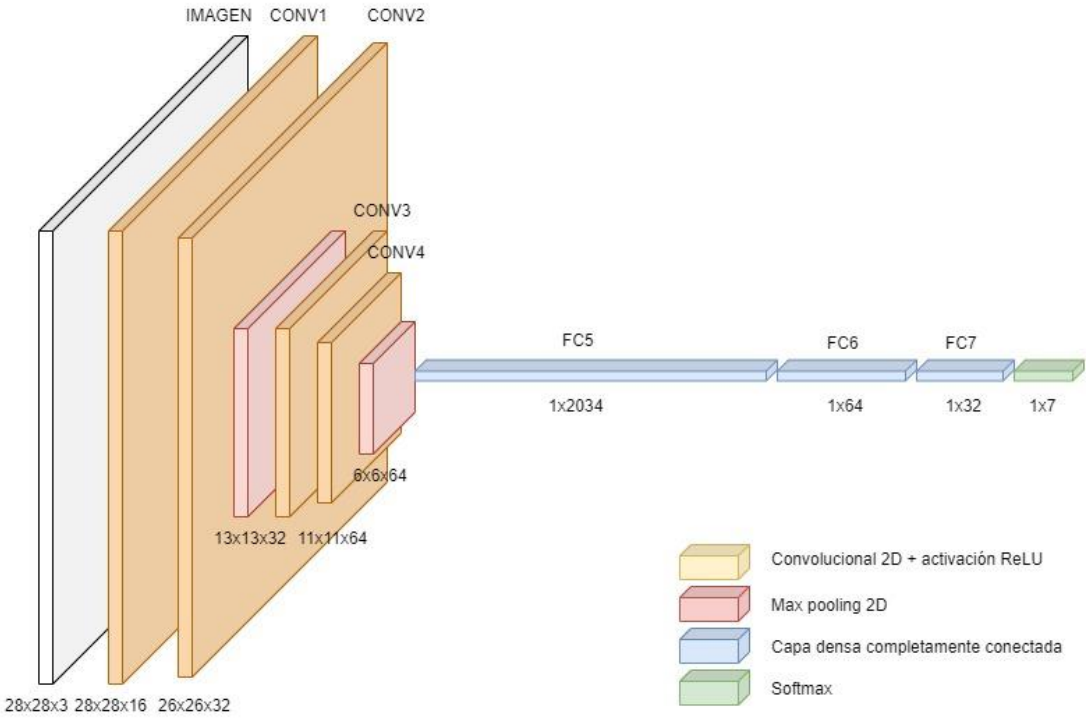


Figura 5. Estructura de CNN *adhoc*

- **Vision Transformer ViT B16**. Propuesto en el artículo de Alexey Dosovitskiy [47], publicado en junio de 2021, siendo el primer artículo que entrena un codificador *transformer* en ImageNet, logrando buenos resultados en comparación con las arquitecturas convolucionales familiares. Este tipo de redes carecen de los sesgos inductivos de las CNN, como la invariancia, pudiendo reconocer un objeto en una imagen incluso cuando su apariencia o posición varía, o la traslación, cuando una cantidad fija de píxeles de la imagen se ha movido una cantidad fija en una dirección particular. Una desventaja frente a las CNN es que no puede procesar datos estructurados en cuadrícula, tienen que

ser secuencias, lo que requiere convertir señales espaciales no secuenciales en secuencias. Los pasos que realiza Vision Transformer son:

1. Dividir la imagen en trozos más pequeños
2. Aplanado de los trozos
3. Creación de incrustaciones lineales de menor dimensión a partir de los trozos aplanados
4. Agregar incrustaciones posicionales
5. Alimentar la secuencia como entrada a un codificador *transformer* estándar
6. Preentrenar el modelo con etiquetas de imagen
7. Ajuste fino en el conjunto de datos para la clasificación de imágenes

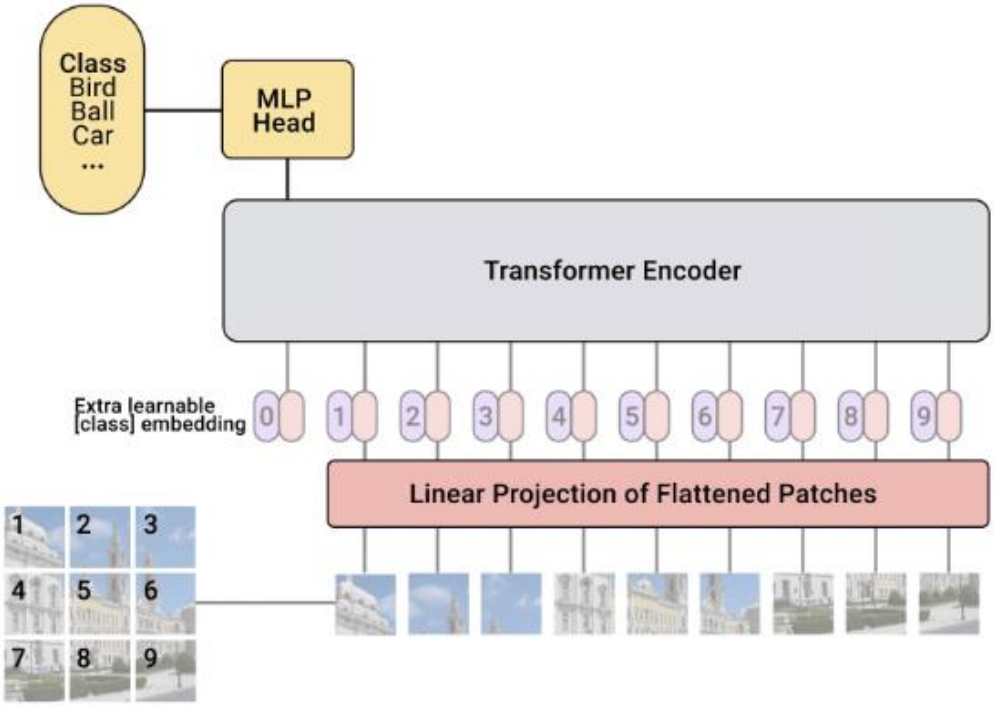


Figura 6. Arquitectura ViT. Tomada del artículo original [47]

- **Swin Transformer**. Es un modelo de aprendizaje profundo basado en *transformer* con un rendimiento de última generación en tareas de visión. A diferencia de Vision Transformer (ViT) que la precede, es altamente eficiente y tiene una mayor precisión. Swin Transformer introduce dos conceptos clave para abordar los problemas a los que se enfrenta ViT: mapas de características jerárquicas y atención de ventana desplazada. Su propio nombre proviene de transformador de ventana desplazada.

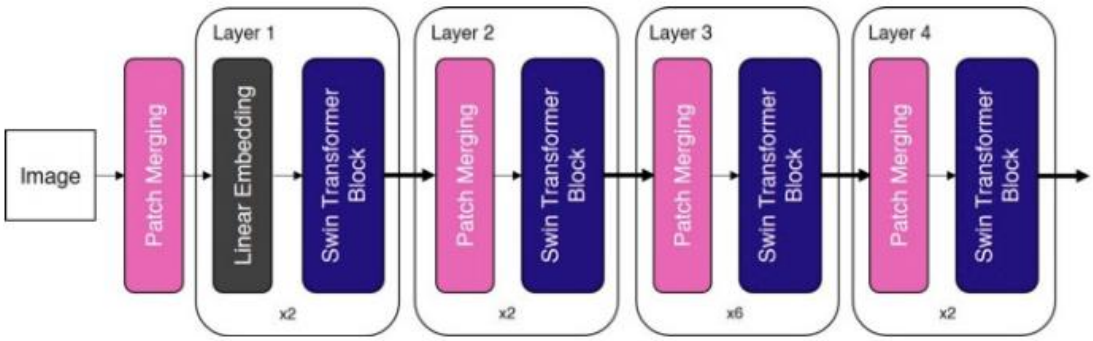


Figura 7. Arquitectura de Swin Transformer. Adaptación del artículo original [48]

- **EfficientNet B0**. EfficientNet [49] es un grupo de 8 modelos de redes neuronales convolucionales (B0 a B7) que, debido a los puntos que se detallan a continuación, es más eficiente que la mayoría de sus predecesores:

    - **Convolución en profundidad + convolución en punto**: divide la convolución original en dos etapas para reducir significativamente el coste de cálculo, con una pérdida mínima de precisión.

    - **Resolución inversa**: los bloques consisten en una capa que primero extiende los canales y después los comprime, de modo que las capas con menos canales se saltan la conexión.

    - **Cuello de botella lineal**: utiliza la activación lineal en la última capa de cada bloque para evitar la pérdida de información de ReLU.

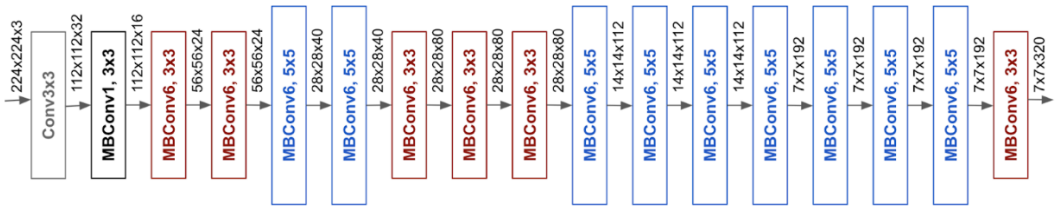


Figura 8. Arquitectura de EfficientNet B0 (Fuente: 55)

- **Xception**. Es una red neuronal convolucional creada por Google que tiene 71 capas de profundidad. Tiene una convolución separable en profundidad modificada, que es la convolución puntual seguida de una convolución en profundidad. Esta modificación está motivada por el módulo de inicio de InceptionV3, en el que la convolución 1x1 se realiza antes de cualquier convolución espacial NxN.

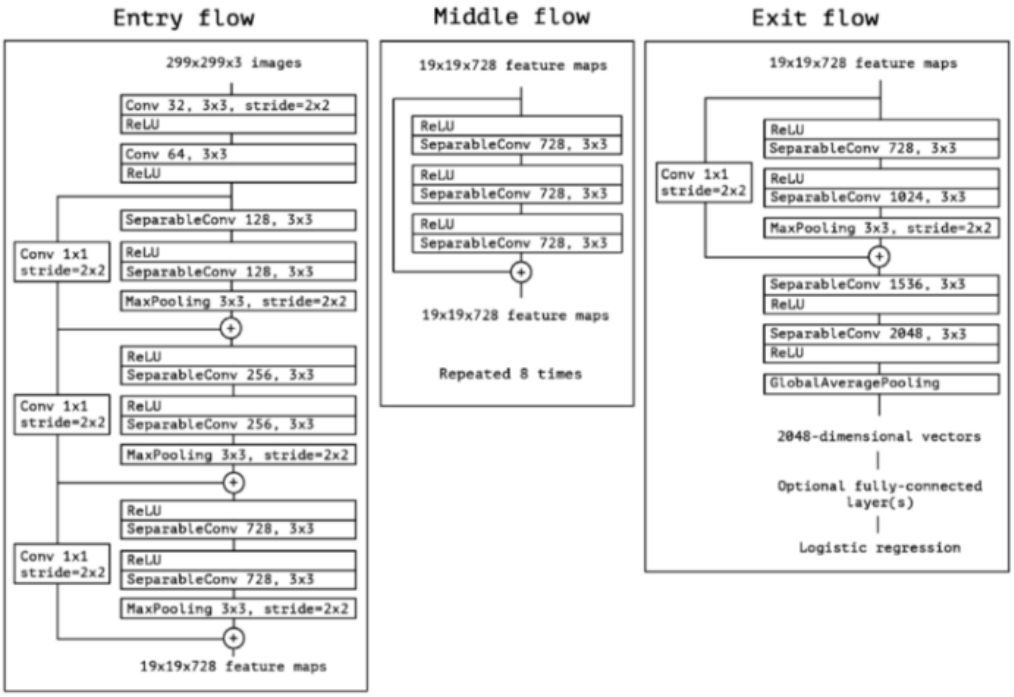


Figura 9. Arquitectura de Xception. Tomada del artículo original [50]

- **ResNet-50**. En el artículo [51] de 2015, se propone una nueva aproximación de diseño para las CNN: la optimización basada en residuos. Los autores lanzaron las tres arquitecturas: ResNet-50, ResNet-101 y ResNet-152. El número en el sufijo indica el número de capas de la red. A pesar de que tiene mayor número de capas que las redes predecesoras, el número de operaciones que realiza es menor. Esto es así porque se implementa una nueva forma de aprendizaje basada en la minimización de residuos, que consiste en saltar desde la entrada de la primera de las capas a la salida de la última cada vez que se añade a la red una serie de capas que incluyan pesos. En la siguiente imagen, extraída del artículo original, se muestra la idea propuesta por los autores:

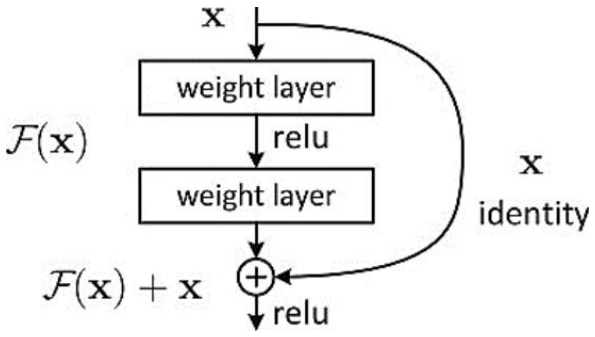


Figura 10. Muestra de bloque de una red de aprendizaje por residuos. Tomada del artículo original [51]

- **VGG16**. es una red neuronal convolucional propuesta por K. Simonyan y A. Zisserman en el artículo [52], y adquirió notoriedad al ganar el Desafío de Reconocimiento Visual a Gran Escala de ImageNet (ILSVRC) en 2014. El modelo

alcanzó una precisión del 92,7% en ImageNet, que es una de las puntuaciones más altas logradas. Supone una mejora respecto a los modelos anteriores al proponer núcleos de convolución más pequeños (3×3) en las capas de convolución de lo que se había hecho anteriormente. El modelo se entrenó durante semanas utilizando tarjetas gráficas de última generación.

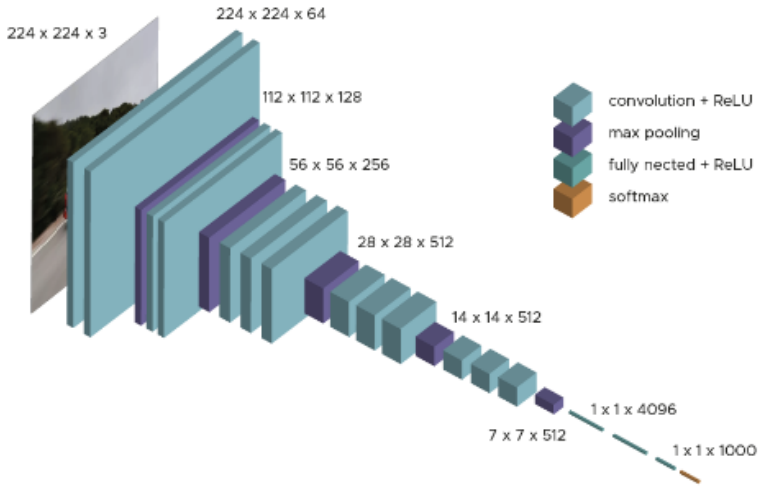


Figura 11. Arquitectura de VGG16 (Fuente: 56)

- **DenseNet**. Esta red se propone en el artículo [53] y se compara con ResNet e Inception, pero tiene una estructura nueva. Su estructura no es complicada, pero es muy efectiva. Cada capa obtiene entradas adicionales de todas las capas anteriores y pasa sus propios mapas de características a todas las capas posteriores, es decir, que se concatenan cada una de las salidas de las capas anteriores con las posteriores, con menor cantidad de parámetros y mayor exactitud que redes como ResNet.

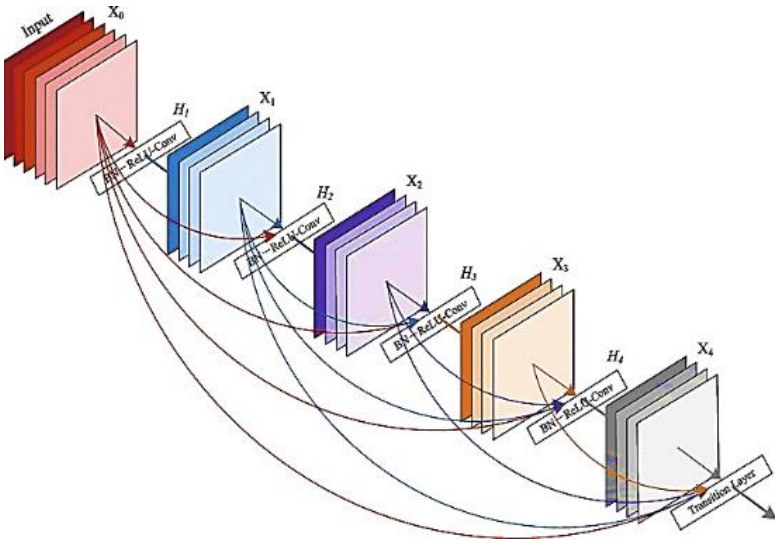


Figura 12. Arquitectura de DenseNet (Fuente: 59)

- **MobileNetV3**. Presentada en el artículo [54], permite utilizar las CPUs de los dispositivos móviles directamente a un rendimiento considerablemente mayor que su versión anterior, MobileNetV2. Permite aumentar la exactitud en ImageNet en comparación con sus redes antecesoras. Se trata de una combinación de las tecnologías MobileNetV2 y *Squeeze and Excite*. Esa última está diseñada para mejorar los resultados que se obtienen a través de la utilización de capas adicionales en puntos estratégicos de la arquitectura de la red, sin llegar a alterarla en su forma. En este caso se añaden estas capas a continuación de la convolución de *Deepwise*, manteniendo las dimensiones y tamaño entre ellas.

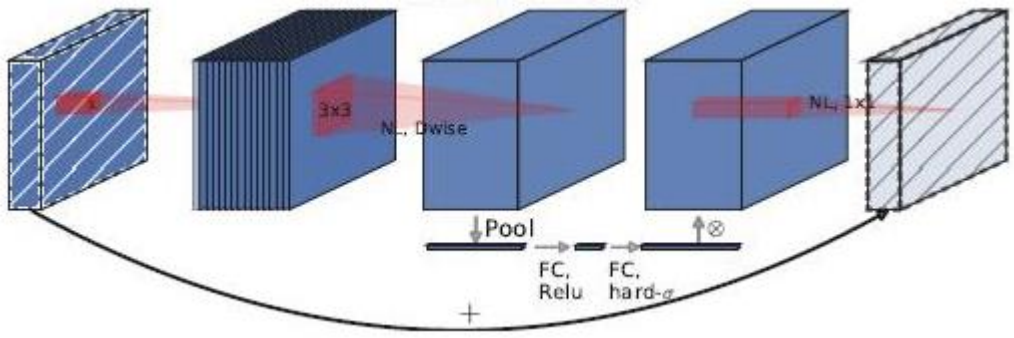


Figura 13. Bloque MobileNetV3. MobileNetV2 + Squeeze-and-Excite. Tomada del artículo original [54]

- **InceptionResNetV2**. Es una red neuronal convolucional que entrenada con más de un millón de imágenes de la base de datos de ImageNet. Su estructura consta de 164 capas de profundidad y puede clasificar imágenes en 1000 categorías de objetos, como el teclado, el ratón, el lápiz y muchos animales. Como resultado, la red ha aprendido representaciones ricas en funciones para una amplia gama de imágenes. Tiene un tamaño de entrada de imagen de 299 por 299 y la salida es una lista de probabilidades de clase estimadas. Está formulada en base a una combinación de la estructura Inception y la conexión Residual. En el bloque Inception-Resnet, se combinan filtros convolucionales de varios tamaños con conexiones residuales. El uso de conexiones residuales no solo evita el problema de degradación causado por estructuras profundas, sino que también reduce el tiempo de entrenamiento.

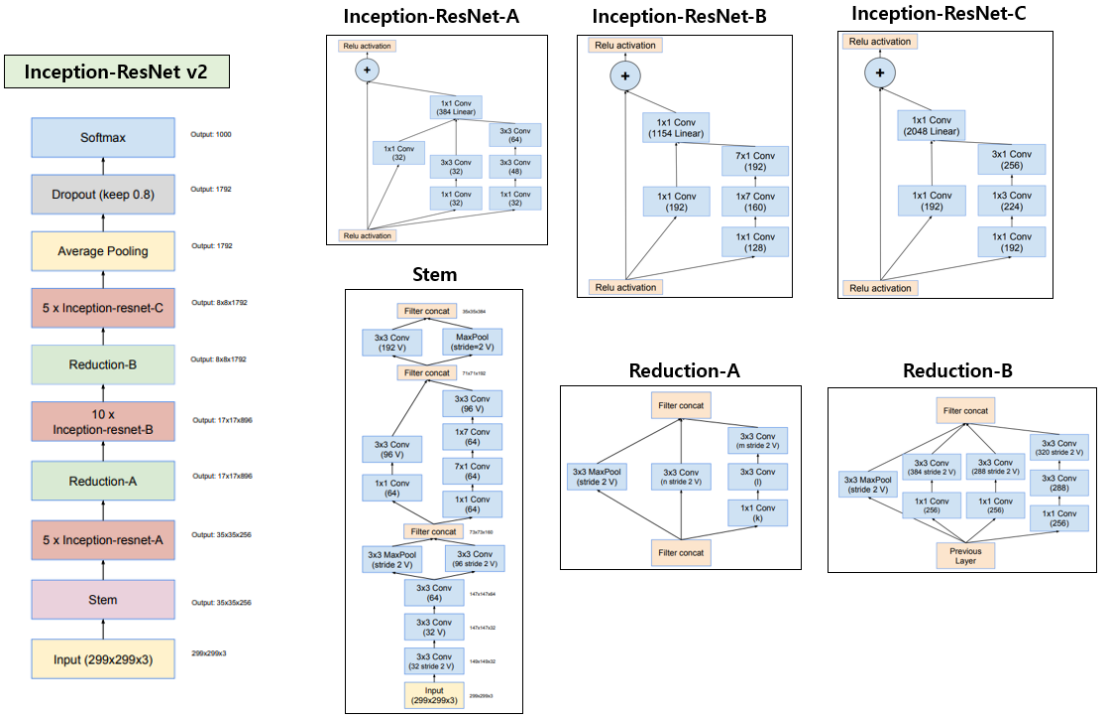


Figura 14. Arquitectura de InceptionResNetV2 (Fuente: 61)

2.5. Configuración y métricas de evaluación de rendimiento

Para evaluar el desempeño de las redes se consideran las métricas más utilizadas por los investigadores en sus artículos, explicadas en el punto 2.1 de este documento. Se establece:

- División del conjunto de datos en: 60% entrenamiento / 20% validación / 20% test.
- Búsqueda de los mejores hiperparámetros. Entrenamientos con diferentes valores para:
  - Número de épocas. Estableciendo un número máximo y configurando parada temprana (*early stopping*) cuando la pérdida (*loss*) en la validación no mejore en un número determinado de épocas.
  - Tamaño del lote (*batch*)
  - Tasa de aprendizaje (*learning rate*)

- Optimizador. Siempre Adam.

- Se configura la realización de puntos de control (*checkpoints*) que guardan los pesos del mejor modelo, entendido como el que obtiene mayor exactitud (*accuracy*) durante la validación.

- Uso de los modelos con los mejores pesos guardados de cada arquitectura de red para realizar las predicciones con el conjunto de pruebas (*test*).

- Con las predicciones realizadas con el conjunto de pruebas se obtendrán las métricas para evaluar el desempeño de cada red.

2.6. Configuración de la inteligencia colectiva

Una vez entrenados y evaluados los diez modelos, se procede a la configuración de la inteligencia colectiva para la toma de decisiones consensuadas, basadas en las predicciones individuales de cada una de las redes que la forman. Con el objetivo de mejorar la exactitud de las decisiones finales, el sistema de consenso contempla ponderaciones calculadas sobre la exactitud (*accuracy*) de los modelos en la clasificación de cada clase.

Las nuevas imágenes son clasificadas por cada una de las redes individualmente y los pesos asignados a cada clase en la predicción se multiplican por los porcentajes de acierto de las clases correspondientes de la matriz de confusión obtenida con el conjunto de pruebas. La tabla 4 muestra una matriz con las redes (R_1 a R_{10}) en las filas y las clases (C_1 a C_7) en las columnas y los valores de los porcentajes de acierto ($\%ACC_{R1C1}$ a $\%ACC_{R10C7}$) para cada una, extraídos de las matrices de confusión de cada modelo.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
R_1	$\%ACC_{R1C1}$	$\%ACC_{R1C2}$	$\%ACC_{R1C3}$	$\%ACC_{R1C4}$	$\%ACC_{R1C5}$	$\%ACC_{R1C6}$	$\%ACC_{R1C7}$
R_2	$\%ACC_{R2C1}$	$\%ACC_{R2C2}$	$\%ACC_{R2C3}$	$\%ACC_{R2C4}$	$\%ACC_{R2C5}$	$\%ACC_{R2C6}$	$\%ACC_{R2C7}$
R_3	$\%ACC_{R3C1}$	$\%ACC_{R3C2}$	$\%ACC_{R3C3}$	$\%ACC_{R3C4}$	$\%ACC_{R3C5}$	$\%ACC_{R3C6}$	$\%ACC_{R3C7}$
R_4	$\%ACC_{R4C1}$	$\%ACC_{R4C2}$	$\%ACC_{R4C3}$	$\%ACC_{R4C4}$	$\%ACC_{R4C5}$	$\%ACC_{R4C6}$	$\%ACC_{R4C7}$
R_5	$\%ACC_{R5C1}$	$\%ACC_{R5C2}$	$\%ACC_{R5C3}$	$\%ACC_{R5C4}$	$\%ACC_{R5C5}$	$\%ACC_{R5C6}$	$\%ACC_{R5C7}$
R_6	$\%ACC_{R6C1}$	$\%ACC_{R6C2}$	$\%ACC_{R6C3}$	$\%ACC_{R6C4}$	$\%ACC_{R6C5}$	$\%ACC_{R6C6}$	$\%ACC_{R6C7}$
R_7	$\%ACC_{R7C1}$	$\%ACC_{R7C2}$	$\%ACC_{R7C3}$	$\%ACC_{R7C4}$	$\%ACC_{R7C5}$	$\%ACC_{R7C6}$	$\%ACC_{R7C7}$
R_8	$\%ACC_{R8C1}$	$\%ACC_{R8C2}$	$\%ACC_{R8C3}$	$\%ACC_{R8C4}$	$\%ACC_{R8C5}$	$\%ACC_{R8C6}$	$\%ACC_{R8C7}$
R_9	$\%ACC_{R9C1}$	$\%ACC_{R9C2}$	$\%ACC_{R9C3}$	$\%ACC_{R9C4}$	$\%ACC_{R9C5}$	$\%ACC_{R9C6}$	$\%ACC_{R9C7}$
R_{10}	$\%ACC_{R10C1}$	$\%ACC_{R10C2}$	$\%ACC_{R10C3}$	$\%ACC_{R10C4}$	$\%ACC_{R10C5}$	$\%ACC_{R10C6}$	$\%ACC_{R10C7}$

Tabla 4. Matriz de pesos con las exactitudes de clasificación de las redes para cada clase

La tabla 5 muestra una matriz, también con las redes en las filas y las clases en las columnas, pero los valores se corresponden con las probabilidades ($\%PROB_{R1C1}$ a $\%PROB_{R10C7}$) asignadas por las redes a cada clase en la predicción de una imagen nueva.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
R_1	$\%PROB_{R1C1}$	$\%PROB_{R1C2}$	$\%PROB_{R1C3}$	$\%PROB_{R1C4}$	$\%PROB_{R1C5}$	$\%PROB_{R1C6}$	$\%PROB_{R1C7}$
R_2	$\%PROB_{R2C1}$	$\%PROB_{R2C2}$	$\%PROB_{R2C3}$	$\%PROB_{R2C4}$	$\%PROB_{R2C5}$	$\%PROB_{R2C6}$	$\%PROB_{R2C7}$
R_3	$\%PROB_{R3C1}$	$\%PROB_{R3C2}$	$\%PROB_{R3C3}$	$\%PROB_{R3C4}$	$\%PROB_{R3C5}$	$\%PROB_{R3C6}$	$\%PROB_{R3C7}$
R_4	$\%PROB_{R4C1}$	$\%PROB_{R4C2}$	$\%PROB_{R4C3}$	$\%PROB_{R4C4}$	$\%PROB_{R4C5}$	$\%PROB_{R4C6}$	$\%PROB_{R4C7}$
R_5	$\%PROB_{R5C1}$	$\%PROB_{R5C2}$	$\%PROB_{R5C3}$	$\%PROB_{R5C4}$	$\%PROB_{R5C5}$	$\%PROB_{R5C6}$	$\%PROB_{R5C7}$
R_6	$\%PROB_{R6C1}$	$\%PROB_{R6C2}$	$\%PROB_{R6C3}$	$\%PROB_{R6C4}$	$\%PROB_{R6C5}$	$\%PROB_{R6C6}$	$\%PROB_{R6C7}$
R_7	$\%PROB_{R7C1}$	$\%PROB_{R7C2}$	$\%PROB_{R7C3}$	$\%PROB_{R7C4}$	$\%PROB_{R7C5}$	$\%PROB_{R7C6}$	$\%PROB_{R7C7}$
R_8	$\%PROB_{R8C1}$	$\%PROB_{R8C2}$	$\%PROB_{R8C3}$	$\%PROB_{R8C4}$	$\%PROB_{R8C5}$	$\%PROB_{R8C6}$	$\%PROB_{R8C7}$
R_9	$\%PROB_{R9C1}$	$\%PROB_{R9C2}$	$\%PROB_{R9C3}$	$\%PROB_{R9C4}$	$\%PROB_{R9C5}$	$\%PROB_{R9C6}$	$\%PROB_{R9C7}$
R_{10}	$\%PROB_{R10C1}$	$\%PROB_{R10C2}$	$\%PROB_{R10C3}$	$\%PROB_{R10C4}$	$\%PROB_{R10C5}$	$\%PROB_{R10C6}$	$\%PROB_{R10C7}$

Tabla 5. Matriz de probabilidades aplicadas por las redes a cada clase en nuevas predicciones

La tabla 6 muestra la matriz con los cálculos para tomar la decisión final de la inteligencia colectiva, basada en las decisiones ponderadas de todas las redes.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	
R_1	$M1_{R1C1} * M2_{R1C1}$	$M1_{R1C2} * M2_{R1C2}$	$M1_{R1C3} * M2_{R1C3}$	$M1_{R1C4} * M2_{R1C4}$	$M1_{R1C5} * M2_{R1C5}$	$M1_{R1C6} * M2_{R1C6}$	$M1_{R1C7} * M2_{R1C7}$	
R_2	$M1_{R2C1} * M2_{R2C1}$	$M1_{R2C2} * M2_{R2C2}$	$M1_{R2C3} * M2_{R2C3}$	$M1_{R2C4} * M2_{R2C4}$	$M1_{R2C5} * M2_{R2C5}$	$M1_{R2C6} * M2_{R2C6}$	$M1_{R2C7} * M2_{R2C7}$	
R_3	$M1_{R3C1} * M2_{R3C1}$	$M1_{R3C2} * M2_{R3C2}$	$M1_{R3C3} * M2_{R3C3}$	$M1_{R3C4} * M2_{R3C4}$	$M1_{R3C5} * M2_{R3C5}$	$M1_{R3C6} * M2_{R3C6}$	$M1_{R3C7} * M2_{R3C7}$	
R_4	$M1_{R4C1} * M2_{R4C1}$	$M1_{R4C2} * M2_{R4C2}$	$M1_{R4C3} * M2_{R4C3}$	$M1_{R4C4} * M2_{R4C4}$	$M1_{R4C5} * M2_{R4C5}$	$M1_{R4C6} * M2_{R4C6}$	$M1_{R4C7} * M2_{R4C7}$	
R_5	$M1_{R5C1} * M2_{R5C1}$	$M1_{R5C2} * M2_{R5C2}$	$M1_{R5C3} * M2_{R5C3}$	$M1_{R5C4} * M2_{R5C4}$	$M1_{R5C5} * M2_{R5C5}$	$M1_{R5C6} * M2_{R5C6}$	$M1_{R5C7} * M2_{R5C7}$	
R_6	$M1_{R6C1} * M2_{R6C1}$	$M1_{R6C2} * M2_{R6C2}$	$M1_{R6C3} * M2_{R6C3}$	$M1_{R6C4} * M2_{R6C4}$	$M1_{R6C5} * M2_{R6C5}$	$M1_{R6C6} * M2_{R6C6}$	$M1_{R6C7} * M2_{R6C7}$	
R_7	$M1_{R7C1} * M2_{R7C1}$	$M1_{R7C2} * M2_{R7C2}$	$M1_{R7C3} * M2_{R7C3}$	$M1_{R7C4} * M2_{R7C4}$	$M1_{R7C5} * M2_{R7C5}$	$M1_{R7C6} * M2_{R7C6}$	$M1_{R7C7} * M2_{R7C7}$	
R_8	$M1_{R8C1} * M2_{R8C1}$	$M1_{R8C2} * M2_{R8C2}$	$M1_{R8C3} * M2_{R8C3}$	$M1_{R8C4} * M2_{R8C4}$	$M1_{R8C5} * M2_{R8C5}$	$M1_{R8C6} * M2_{R8C6}$	$M1_{R8C7} * M2_{R8C7}$	
R_9	$M1_{R9C1} * M2_{R9C1}$	$M1_{R9C2} * M2_{R9C2}$	$M1_{R9C3} * M2_{R9C3}$	$M1_{R9C4} * M2_{R9C4}$	$M1_{R9C5} * M2_{R9C5}$	$M1_{R9C6} * M2_{R9C6}$	$M1_{R9C7} * M2_{R9C7}$	
R_{10}	$M1_{R10C1} * M2_{R10C1}$	$M1_{R10C2} * M2_{R10C2}$	$M1_{R10C3} * M2_{R10C3}$	$M1_{R10C4} * M2_{R10C4}$	$M1_{R10C5} * M2_{R10C5}$	$M1_{R10C6} * M2_{R10C6}$	$M1_{R10C7} * M2_{R10C7}$	
IC	**SUM(C_1)**	**SUM(C_2)**	**SUM(C_3)**	**SUM(C_4)**	**SUM(C_5)**	**SUM(C_6)**	**SUM(C_7)**	**MAX(IC)**

Tabla 6. Matriz con las decisiones individuales ponderadas y cálculo de la decisión de la inteligencia colectiva

2.7. Configuración de la aplicación web

Para que el clasificador pueda ser utilizado con nuevas imágenes por los usuarios, se considera su configuración como servicio web, con la intención de que no requiera instalación de software adicional ni necesidades de disponer de hardware específico. El único requisito para su uso es un navegador web. La aplicación puede ejecutarse en servidor web local o remoto. Este trabajo utiliza el *framework* Flask [60], debido a su facilidad para crear aplicaciones web escritas en Python de forma rápida con un número mínimo de líneas de código.

3. Resultados

3.1. Exploración del conjunto de datos

Antes de empezar a entrenar las redes neuronales, es necesario conocer bien los datos con los que las entrenaremos. El conjunto HAM10000 contiene 10.015 imágenes en formato JPG y cinco ficheros en formato CSV. Cuatro de los ficheros contienen los valores de los píxeles de las imágenes en dos tamaños diferentes, 8 x 8 y 28 x 28, en formato RGB y escala de grises, además de la etiqueta en formato numérico con la clase a la que pertenece la imagen. El quinto fichero CSV contiene los metadatos siguientes: tipo de lesión en formato de texto con la que está etiquetada la imagen, tipo de confirmación del diagnóstico, edad del paciente, género del paciente y localización en el cuerpo de la lesión. El fichero también incluye los identificadores de las lesiones y de las imágenes. La siguiente tabla muestra las dimensiones de todos los ficheros CSV:

Fichero	# Filas	# Columnas	# Elementos
hmnist_8_8_L.csv	10.015	65	650.975
hmnist_8_8_RGB.csv	10.015	193	1.932.895
hmnist_28_28_L.csv	10.015	785	7.861.775
hmnist_28_28_RGB.csv	10.015	2.353	23.565.295
HAM10000_metadata.csv	10.015	7	70.105

Tabla 7. Dimensiones de los ficheros de metadatos del conjunto HAM10000

Para este trabajo únicamente se utiliza el fichero de metadatos y el fichero que contiene los valores de los píxeles en formato 28 x 28 x 3.

	pixel0000	pixel0001	pixel0002	pixel0003	pixel0004	pixel0005	pixel0006	pixel0007	pixel0008	pixel0009	...	pixel2343
0	192	153	193	195	155	192	197	154	185	202	...	173
1	25	14	30	68	48	75	123	93	126	158	...	60
2	192	138	153	200	145	163	201	142	160	206	...	167
3	38	19	30	95	59	72	143	103	119	171	...	44
4	158	113	139	194	144	174	215	162	191	225	...	209

5 rows × 2353 columns

Figura 15. Cinco primeros registros del fichero hmnist_28_28_RGB.csv

	lesion_id	image_id	dx	dx_type	age	sex	localization
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear

Figura 16. Cinco primeros registros del fichero HAM10000_metadata.csv

3.2. Limpieza de datos

En el fichero de metadatos hay 57 valores nulos en la columna que contiene las edades de los pacientes (*age*). Como representan un porcentaje muy reducido del total (~0,6%), los registros podían haber sido descartados, pero se ha considerado una mejor opción, reemplazar los valores nulos por el valor medio del resto.

3.3. Visualización de imágenes aleatorias

Se realiza una exploración visual de cinco imágenes aleatorias de cada clase:

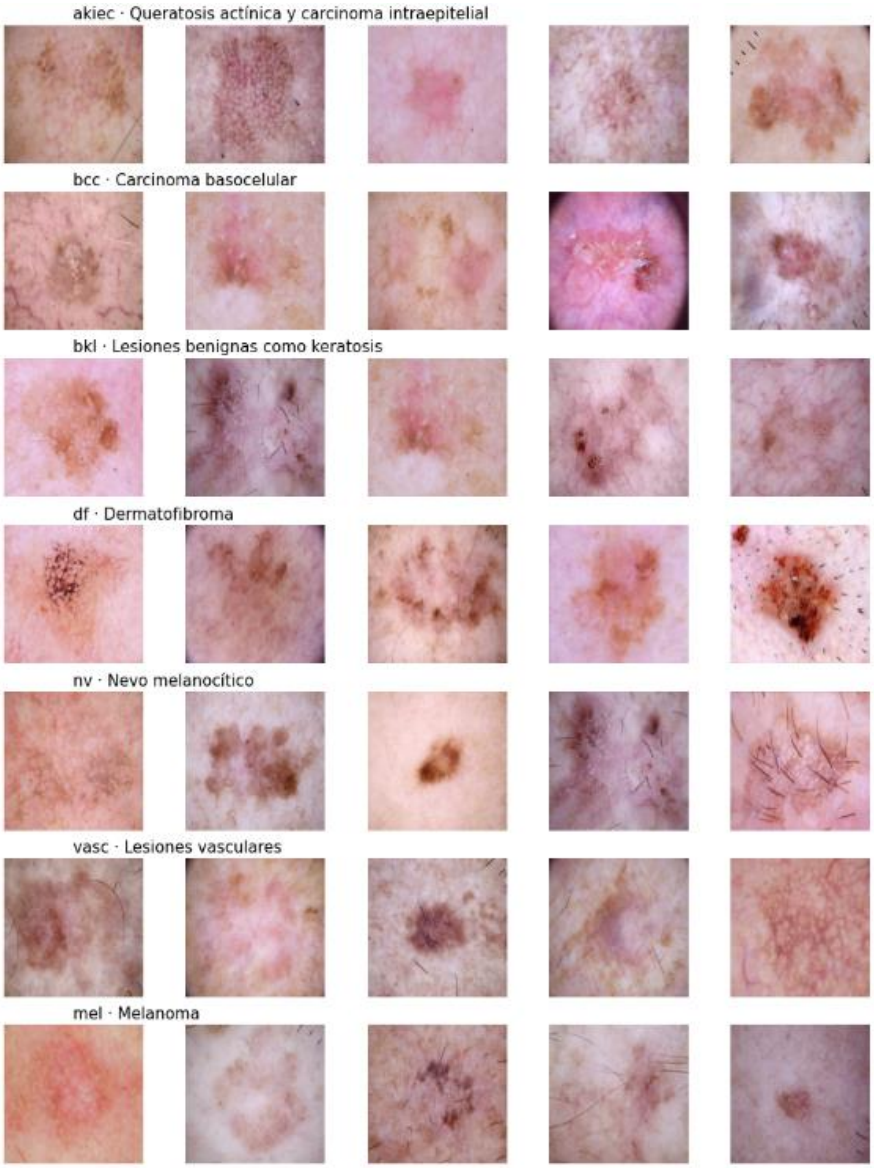


Figura 17. Cinco imágenes aleatorias de tipo de lesión

3.4. Análisis de datos

En el primer análisis se observa que los datos están desbalanceados, predominando la clase “nv”, con 6.705 imágenes, a la que le siguen las imágenes de las clases “mel” y “bkl” con 1.113 y 1.099 imágenes respectivamente. El resto de las clases tienen entre 115 y 514 imágenes. Para corregir el desbalanceo se hará uso de técnicas de sobremuestreo y aumento de datos.

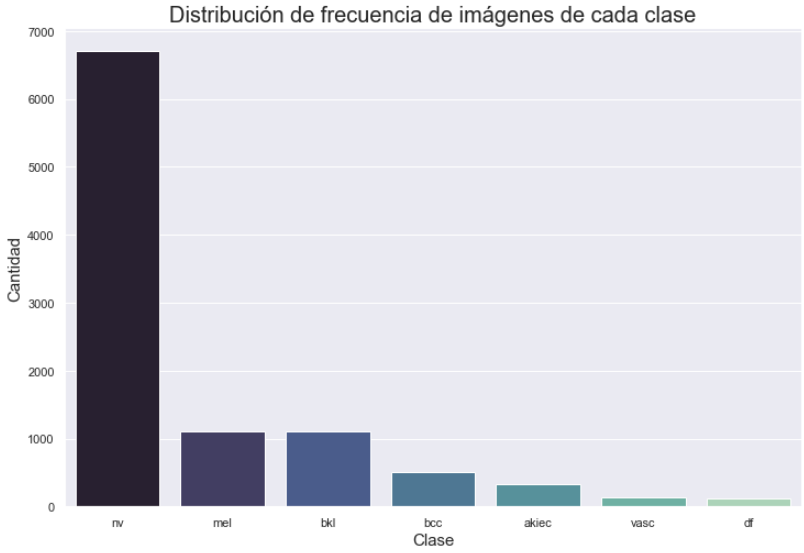


Figura 18. Diagrama de barras con la distribución de imágenes de cada clase

En el siguiente diagrama de barras observamos que el método de confirmación del diagnóstico más frecuente es por histopatología (53,32%), seguido por hacer seguimiento (36,98%), consenso (9%) y microscopio confocal (0,69%). Esto significa que para casi el 50% de los casos son necesarios costes adicionales e inversión de más tiempo para confirmar el diagnóstico, debido a la falta de precisión visual del especialista.

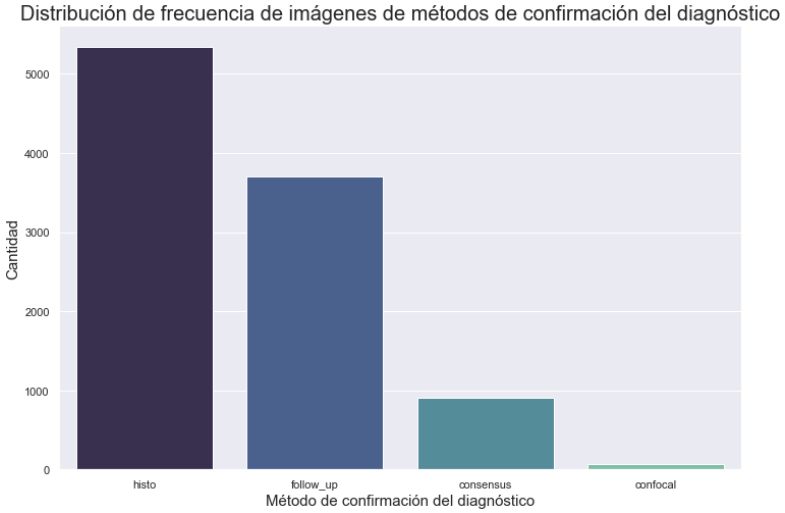


Figura 19. Diagrama de barras con la distribución de imágenes de los métodos de confirmación del diagnóstico

En el histograma con las edades de los pacientes, se observa una distribución estándar con el máximo en el intervalo comprendido entre los 40 y 50 años.

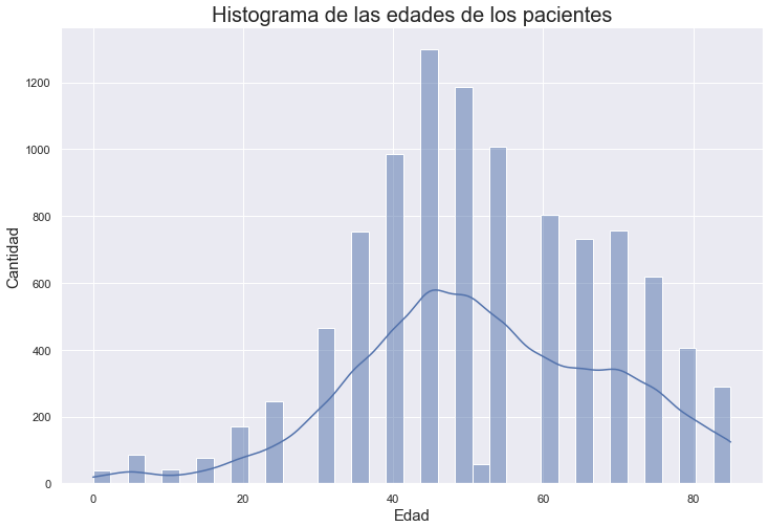


Figura 20. Histograma de las edades de los pacientes

El siguiente diagrama de barras muestra las partes del cuerpo en las que se localizan las lesiones de las imágenes. Las ubicaciones con más frecuencia de lesiones son la espalda, extremidades inferiores y tronco.

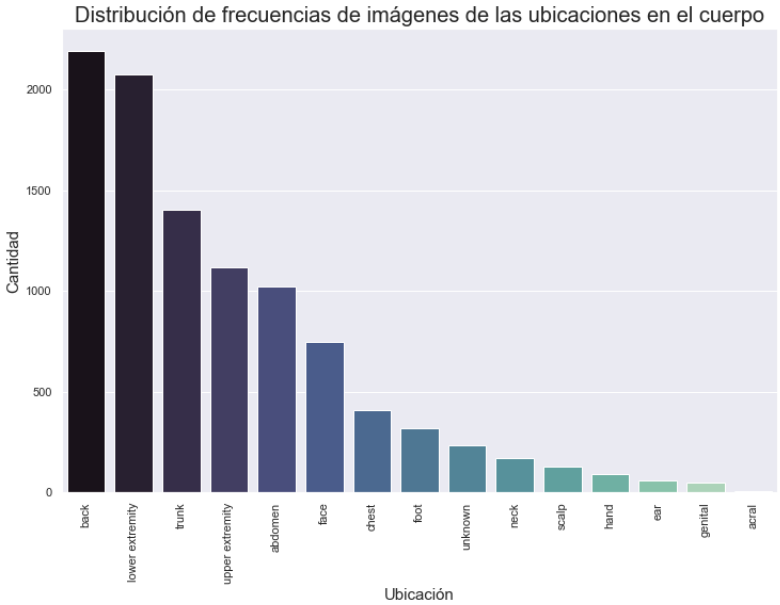


Figura 21. Diagrama de barras de distribución de frecuencias de las imágenes en el cuerpo

Las imágenes están bien balanceadas en cuanto al género, tal y como se observa en el gráfico de tarta siguiente:

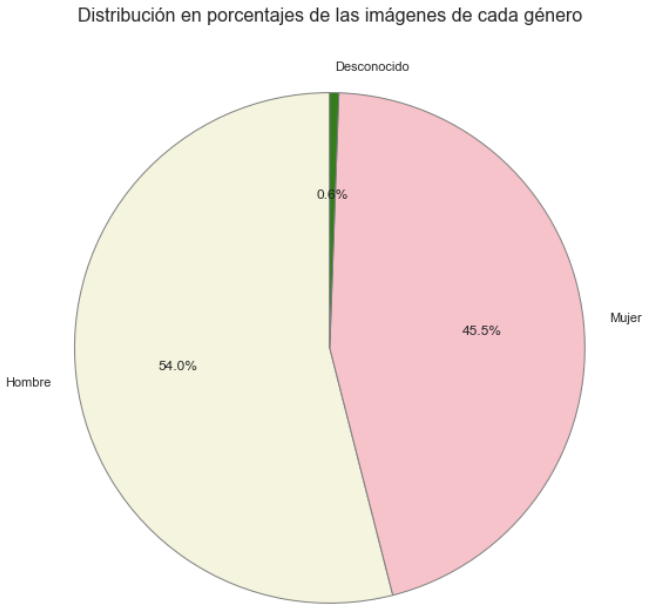


Figura 22. Gráfico de tarta con la distribución de imágenes en géneros

En el histograma por edades y géneros, se observa cómo hasta los 45 años aproximadamente, predominan las lesiones en mujeres, pero a partir de esa edad hay muchas más lesiones en hombres.

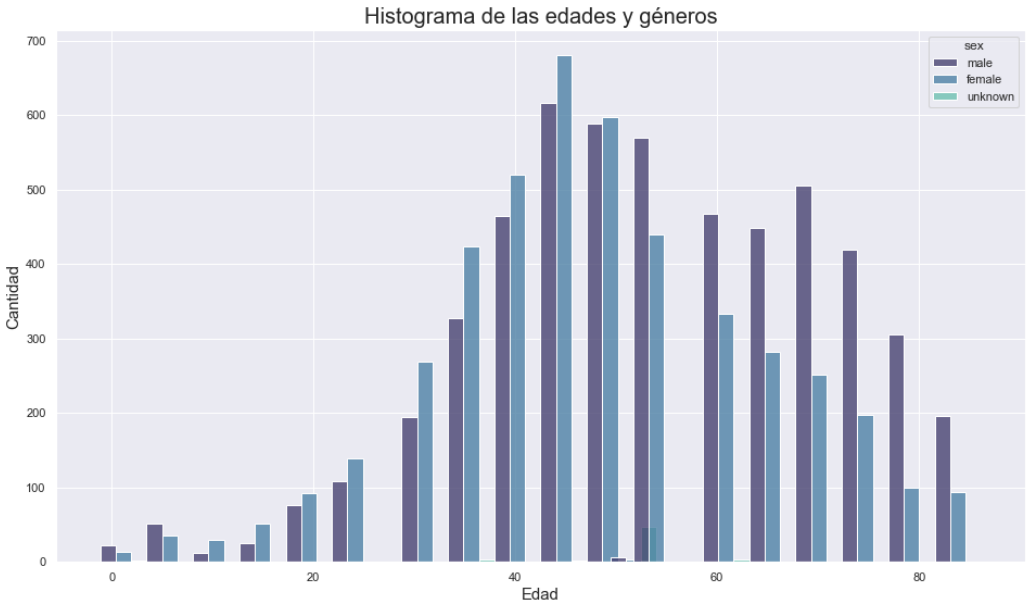


Figura 23. Histograma de edades y géneros

En el diagrama de barras con la distribución de lesiones por género y según la ubicación en el cuerpo, se observa que en la espalda y el tronco son más frecuentes en los hombres, mientras que en las extremidades inferiores es en las mujeres donde más hay.

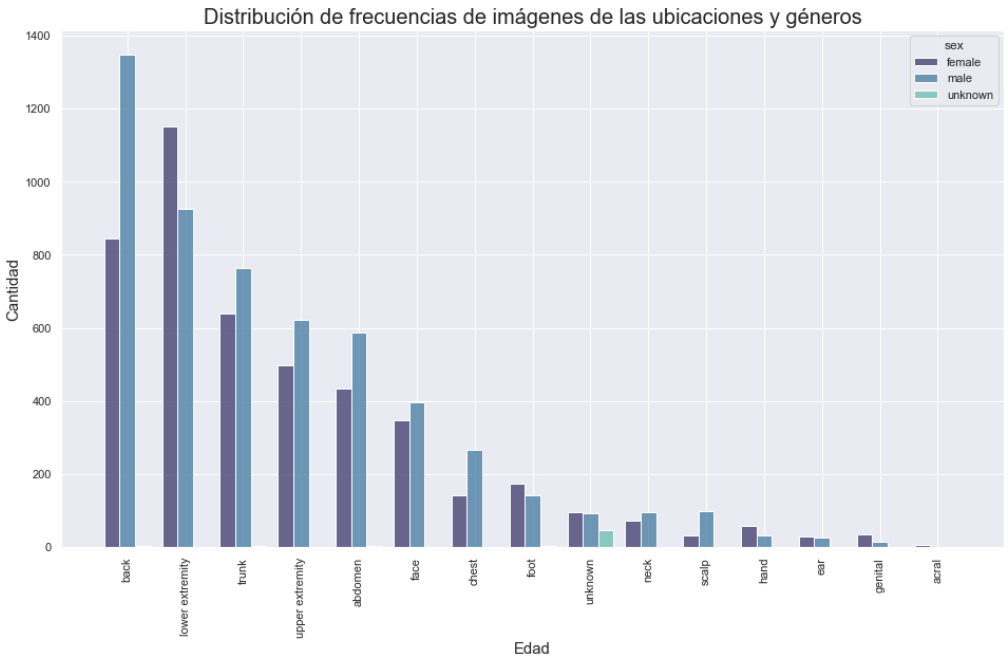


Figura 24. Diagrama de barras con la distribución de imágenes en géneros y localización

3.5. Entrenamientos, evaluaciones y resultados

En este apartado se detallan los entrenamientos, evaluaciones y resultados obtenidos con cada una de las redes que forman parte de la inteligencia colectiva. Para cada red, se indican los datos utilizados, hiperparámetros evaluados, resultados obtenidos durante las fases de entrenamiento y validación, mediante las métricas de exactitud (*accuracy*) y pérdida (*loss*) y la curva ROC, y con los conjuntos de pruebas, haciendo uso de la matriz de confusión y sus métricas.

El número de épocas se fija en 50 para todos los modelos de red y se configura la parada temprana cuando la pérdida en la validación (*var_loss*) no mejora durante cinco épocas consecutivas. Esto se hace mediante la función *EarlyStopping* de TensorFlow (tf.keras.callbacks.EarlyStopping).

Se parametriza la caída de la tasa de aprendizaje con un valor mínimo de delta de 0.0001, un valor mínimo para la tasa de aprendizaje de 0.000001 y factor de 0.2, cuando la exactitud en la validación no mejora durante dos épocas consecutivas. La configuración se realiza con la función ReduceLROnPlateau de TensorFlow (tf.keras.callbacks. ReduceLROnPlateau).

Los puntos de control se configuran para que solo guarden los pesos del mejor modelo, entendido como el que obtiene mayor exactitud en la validación (val_*accuracy*), mediante la función de ModelCheckpoint de TensorFlow (tf.keras.callbacks.ModelCheckpoint).

Los resultados de los entrenamientos se guardan en una tabla, objeto *DataFrame* de la librería Pandas, y contiene los valores de exactitud (*accuracy*) y pérdida (*loss*) de las fases de entrenamiento y validación del mejor modelo por cada combinación de hiperparámetros probada. La tabla también contiene los valores de los hiperparámetros, con el objeto de conocer que combinación funcionó mejor. Una vez finalizados los entrenamientos, se imprime la tabla ordenada descendentemente por el valor de la exactitud en la validación, que es la métrica que mejor indica el desempeño obtenido por el modelo durante el entrenamiento.

3.5.1. CNN *adhoc*

La CNN *adhoc* se entrena con todas las imágenes del conjunto de datos con dimensiones 28x28x3. Se aplica la técnica de sobremuestreo para corregir el desbalanceo de imágenes de las clases y aumento de datos durante el entrenamiento para reducir el sobreentrenamiento y mejorar la generalización. Para el sobremuestreo se utiliza la función RandomOverSampler de la librería imblearn.over_sampling y para el aumento de datos se utiliza la función ImageDataGenerator de la librería tensorflow.keras.preprocessing.image.

```
TABLA DE FRECUENCIAS

label
0    6705
1    6705
2    6705
3    6705
4    6705
5    6705
6    6705
Name: label, dtype: int64
```

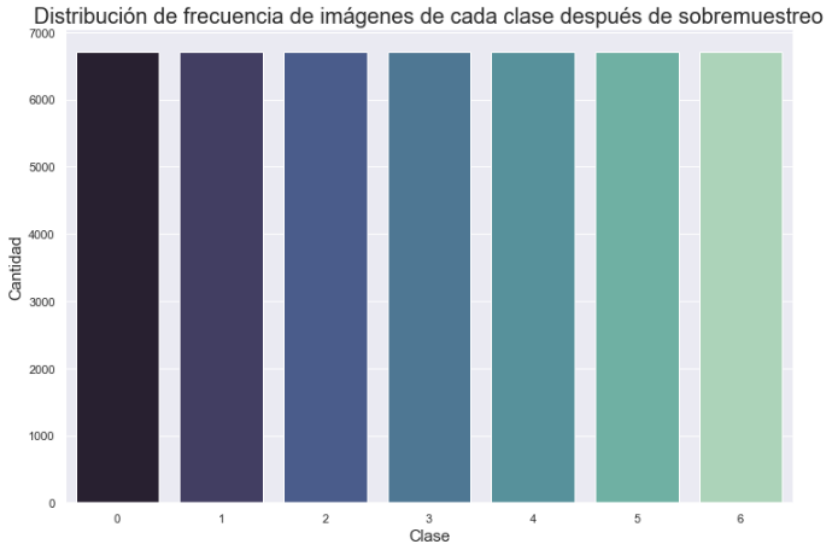


Figura 25. Distribución de imágenes en clases después de aplicar el sobremuestreo

Los tamaños del lote utilizados para entrenar esta red son 16, 32, 64 y 128. Las tasas de aprendizaje utilizadas son: 0.001, 0.0005 y 0.0001.

3.5.1.1.Historial de entrenamientos

En la tabla de entrenamientos de esta red, se observa como los mejores resultados se han obtenido con un tamaño del lote de 16 y una tasa de aprendizaje de 0.001. A esta red le va mejor un tamaño del lote más pequeño y una tasa de aprendizaje más grande.

	epochs	batch_size	learning_rate	loss	accuracy	val_loss	val_accuracy	elapsed_time
0	50	16	0.0010	0.0067	0.9982	0.0982	0.9812	147.650650
4	50	32	0.0005	0.0025	1.0000	0.1224	0.9810	105.646097
6	50	64	0.0010	0.0021	0.9999	0.1174	0.9806	53.518609
9	50	128	0.0010	0.0047	0.9997	0.1049	0.9796	48.851923
3	50	32	0.0010	0.0064	0.9986	0.1128	0.9794	93.632691
7	50	64	0.0005	0.0037	0.9996	0.1340	0.9767	81.852411
2	50	16	0.0001	0.0128	0.9982	0.1143	0.9743	275.451342
5	50	32	0.0001	0.0287	0.9942	0.1195	0.9734	211.669783
10	50	128	0.0005	0.0184	0.9968	0.1173	0.9714	70.509134
1	50	16	0.0005	0.0514	0.9828	0.1621	0.9647	136.559159

Tabla 8. Tabla con el historial de entrenamientos de la red CNN *adhoc*

A continuación, se visualizan las gráficas con la exactitud y pérdida obtenidas durante las fases de entrenamiento y validación. Una evidencia de que el modelo aprende con el transcurso de las épocas es que las líneas que representan la exactitud suben en el eje vertical y las líneas que representan la pérdida bajan en paralelo sin que la separación entre ambas aumente ya que, de ser así, sería signo de que el modelo está sobreentrenado.

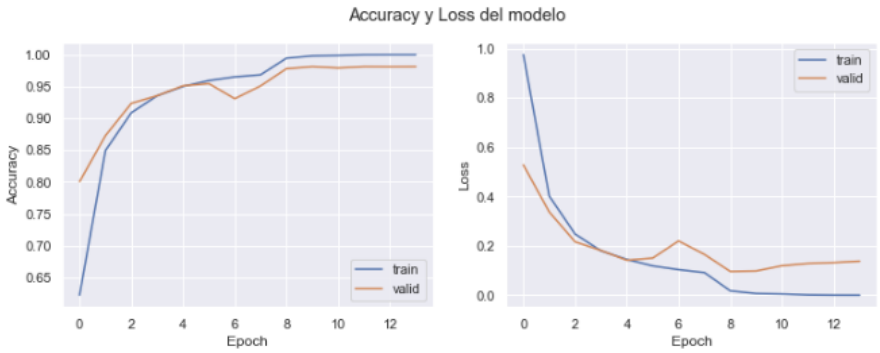


Figura 26. Gráficas de *accuracy* y *loss* de la red CNN *adhoc*

3.5.1.2.Evaluación del modelo

La evaluación del modelo se realiza con el conjunto de prueba (test) y es un buen indicador de la bondad del modelo. Se utiliza la función evaluate de Tensorflow (tf.keras.Model.evaluate).

```
# Evaluación del modelo
model = build_model_cnn_adhoc()
model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.load_weights(input_path + 'best_model1.h5')

loss, acc = model.evaluate(X_test, y_test, verbose = 2)   # Evaluación del modelo con el conjunto de test
```

```
294/294 - 3s - loss: 0.0968 - accuracy: 0.9814 - 3s/epoch - 11ms/step
```

Figura 27. Evaluación de la red CNN *adhoc*

3.5.1.3.Curva ROC

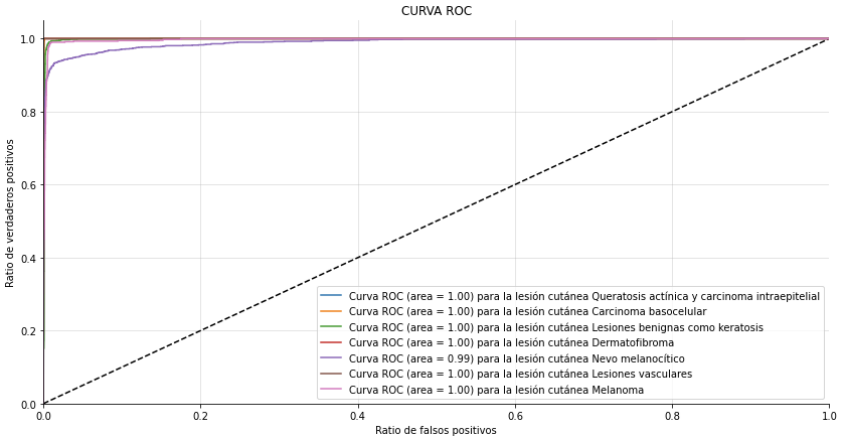


Figura 28. Curva ROC de la red CNN *adhoc*

3.5.1.4.Matriz de confusión

La matriz de confusión se genera en valores absolutos y en porcentajes, siendo esta última forma la más recomendada para interpretarla.

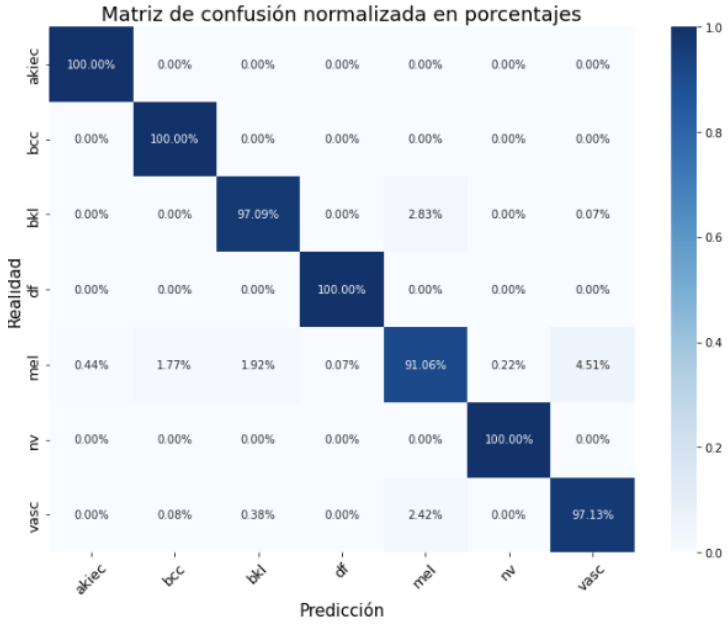


Figura 29. Matriz de confusión de la red CNN *adhoc*

3.5.1.5.Métricas

Accuracy	0.98061
Precision	0.98061
Recall	0.98069
F1 Score	0.98053

Tabla 9. Métricas de la red CNN *Adhoc*

3.5.2. Vision Transformer ViT-B16

La red Vision Transformer ViT-B16 requiere que las imágenes tengan dimensiones 224x224, con lo que es necesario extraer sus características en ese tamaño. Con el objetivo de no sobrepasar la memoria del equipo, se recorre el fichero de metadados entero y se redimensionan las imágenes en trozos (*chunks*) de 100 unidades cada uno para, posteriormente, unirlos en un nuevo *dataframe*. Una vez obtenida la nueva tabla con las características de las 10.015 imágenes con dimensiones 224x224x3, se realiza una selección de las imágenes, fijando un máximo de 750 por categoría. Después se aplica la misma técnica de sobremuestreo utilizada con el conjunto de datos que se entrenó la CNN *adhoc*, obteniendo un total de 5.250 imágenes, 750 de cada categoría. Este conjunto se utiliza con el resto de las redes.

Para entrenar esta red se utilizan los tamaños del lote de 8, 12 y 16. Las tasas de aprendizaje utilizadas son: 0.001, 0.0005 y 0.0001.

3.5.2.1. Historial de entrenamientos

El mejor modelo se obtiene con tamaño del lote de 16 y tasa de aprendizaje 0.0001.

	epochs	batch_size	learning_rate	loss	accuracy	val_loss	val_accuracy	elapsed_time
8	50	16	0.0001	0.1425	0.9886	0.3491	0.9086	6810.953967
5	50	12	0.0001	0.0993	0.9917	0.3950	0.9048	5086.418187
2	50	8	0.0001	0.1883	0.9717	0.3921	0.8867	5048.729994
7	50	16	0.0005	0.1725	0.9569	0.4304	0.8686	7043.452579
4	50	12	0.0050	0.3110	0.9110	0.4756	0.8505	8448.269984
1	50	8	0.0050	0.4133	0.8671	0.5469	0.8200	8611.470835
6	50	16	0.0010	0.5445	0.9917	0.6262	0.7895	7904.796971
0	50	8	0.0010	0.5746	0.8207	0.6555	0.7791	10335.149066
3	50	12	0.0010	0.5673	0.8076	0.6714	0.7648	7051.271807

Tabla 10. Tabla con el historial de entrenamientos de la red ViT-B16

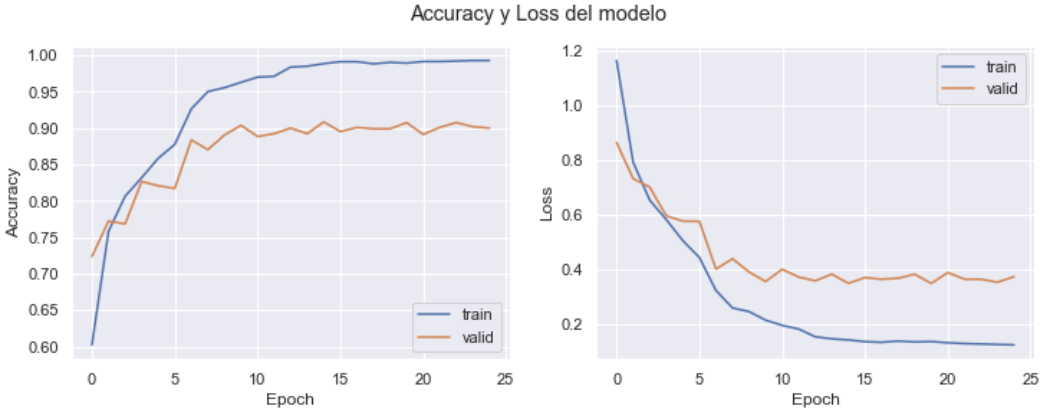


Figura 30. Gráficas de *accuracy* y *loss* de la red ViT-B16

3.5.2.2.Evaluación del modelo

```
# Evaluación del modelo
model = build_model_vit()
model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.load_weights(input_path + 'best_model2.h5')

loss, acc = model.evaluate(X_test, y_test, verbose = 2) # Evaluación del modelo con el conjunto de test
```

```
33/33 - 26s - loss: 0.3493 - accuracy: 0.9019 - 26s/epoch - 800ms/step
```

Figura 31. Evaluación de la red ViT-B16

3.5.2.3.Curva ROC

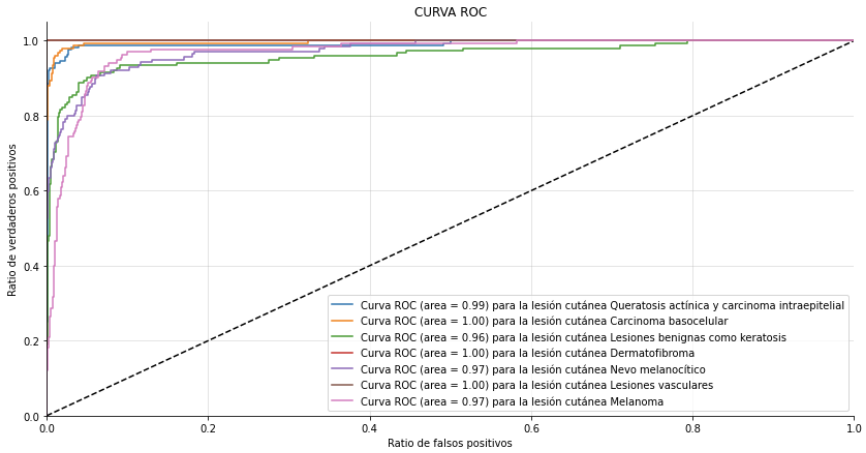


Figura 32. Curva Roc de la red ViT-B16

3.5.2.4.Matriz de confusión

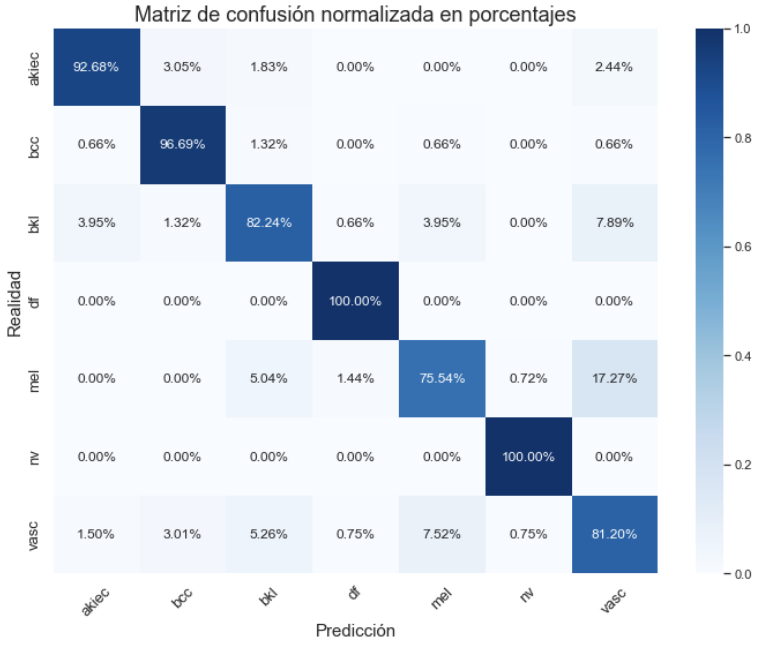


Figura 33. Matriz de confusión de la red ViT-B16

3.5.2.5.Métricas

Accuracy	0.90190
Precision	0.89832
Recall	0.89764
F1 Score	0.89696

Tabla 11. Métricas de la red ViT-B16

3.5.3. Swin Transformer

Los entrenamientos se realizan con valor un tamaño de lote 2 y tasas de aprendizaje 0.00001 y 0.000005.

3.5.3.1.Historial de entrenamientos

El mejor modelo se obtiene con tamaño del lote 2 y tasa de aprendizaje de 0.000005.

	epochs	batch_size	learning_rate	loss	accuracy	val_loss	val_accuracy	elapsed_time
1	50	2	0.000005	0.036860	0.987857	0.422922	0.885714	19038.190399
0	50	2	0.000010	0.046207	0.987381	0.442249	0.884762	14194.940979

Tabla 12. Tabla con el historial de entrenamientos de la red Swin Transformer

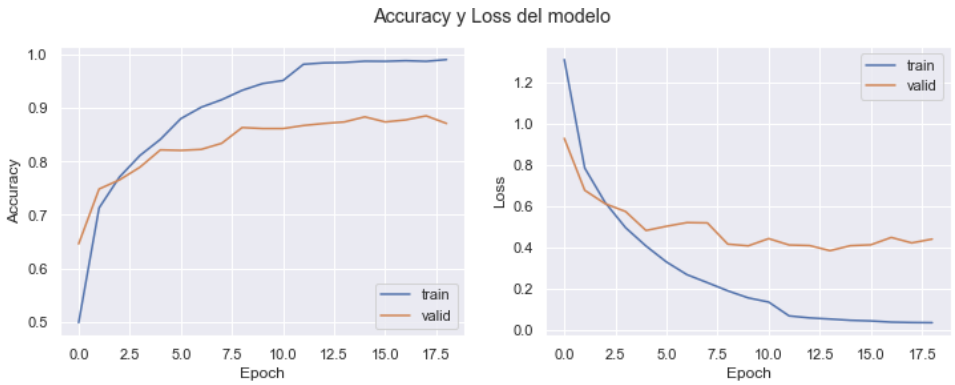


Figura 34. Gráficas de *accuracy* y *loss* de la red Swin Transformer

3.5.3.2.Evaluación del modelo

```
# Evaluación del modelo
model = build_model_SwinT()
model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.load_weights(input_path + 'best_model3.h5')

loss, acc = model.evaluate(X_test, y_test, verbose = 2) # Evaluación con el conjunto de test
```

```
33/33 - 62s - loss: 0.3900 - accuracy: 0.8781 - 62s/epoch - 2s/step
```

Figura 35. Evaluación del modelo de la red Swin Transformer

3.5.3.3.Curva ROC

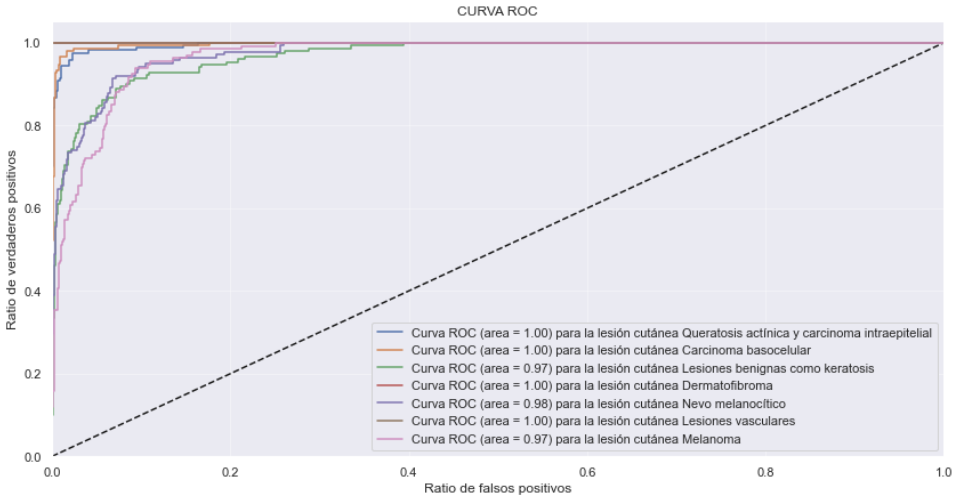


Figura 36. Curva ROC de la red Swin Transformer

3.5.3.4.Matriz de confusión

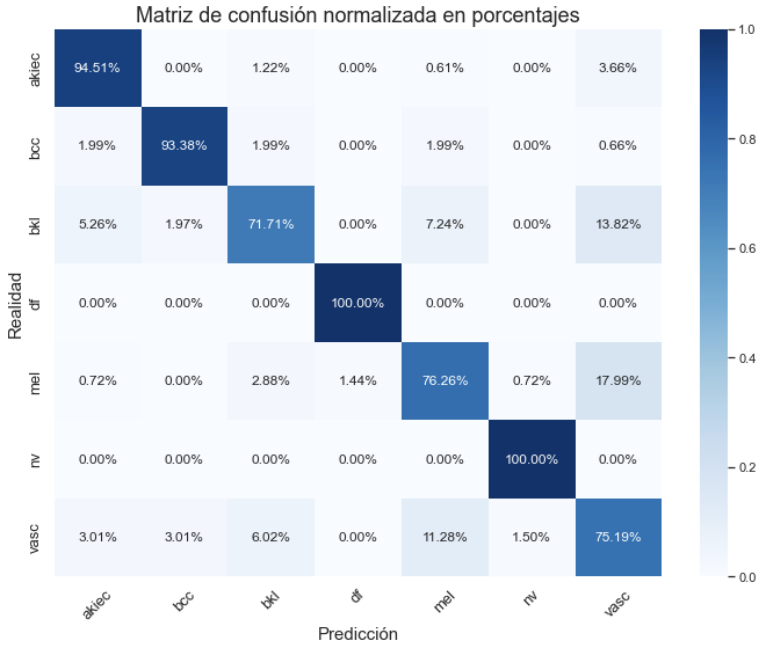


Figura 37. Matriz de confusión de la red Swin Transformer

3.5.3.5.Métricas

Accuracy	0.87810
Precision	0.87513
Recall	0.87292
F1 Score	0.87244

Tabla 13. Métricas de la red Swin Transformer

3.5.4. Red EfficientNet B0

Los entrenamientos se realizan con valores 8, 16 y 32 para el tamaño de lote y tasas de aprendizaje 0.001, 0.0005 0.0001.

3.5.4.1.Historial de entrenamientos

El mejor modelo se obtiene con tamaño del lote 32 y tasa de aprendizaje de 0.0005.

	epochs	batch_size	learning_rate	loss	accuracy	val_loss	val_accuracy	elapsed_time
7	50	32	0.0005	0.127204	0.955952	0.335595	0.910476	2055.089268
8	50	32	0.0001	0.158568	0.944762	0.349902	0.901905	1980.286054
1	50	8	0.0005	0.315411	0.880000	0.354821	0.887619	1794.649295
4	50	16	0.0005	0.259846	0.902857	0.394074	0.887619	1469.195320
6	50	32	0.0010	0.267091	0.901905	0.386346	0.879048	1578.670110
2	50	8	0.0001	0.442779	0.839524	0.391756	0.867619	1700.920623
3	50	16	0.0010	0.342122	0.877381	0.385865	0.867619	1753.032865
5	50	16	0.0001	0.381007	0.860238	0.413944	0.861905	1608.971167
0	50	8	0.0010	0.488366	0.803333	0.506268	0.826667	2593.265908

Tabla 14. Tabla con el historial de entrenamientos de la red EfficientNet B0

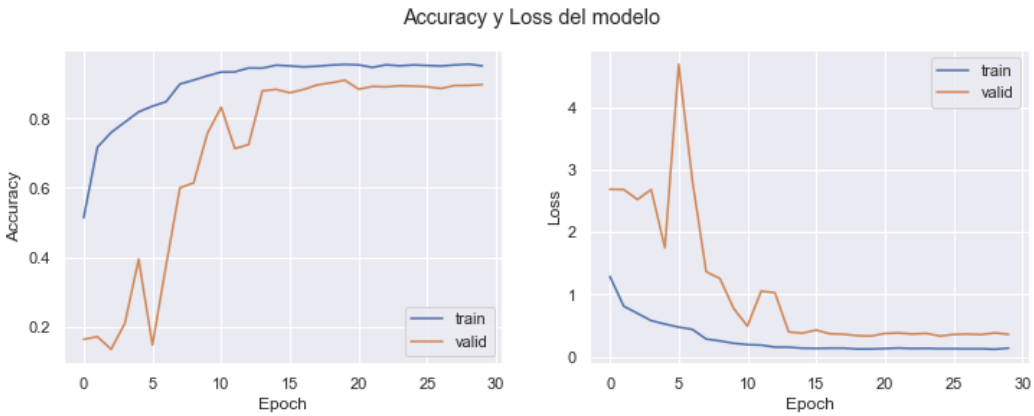


Figura 38. Gráficas de *accuracy* y *loss* de la red EficcientNet B0

3.5.4.2.Evaluación del modelo

```
# Evaluación del modelo
model = build_model_EfficientNet()
model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.load_weights(input_path + 'best_model4.h5')
model.save() # Guardado completo del modelo

loss, acc = model.evaluate(X_test, y_test, verbose = 2) # Evaluación con el conjunto de test
```

```
33/33 - 5s - loss: 0.4240 - accuracy: 0.8819 - 5s/epoch - 141ms/step
```

Figura 39. Evaluación del modelo de la red EfficientNet B0

3.5.4.3.Curva ROC

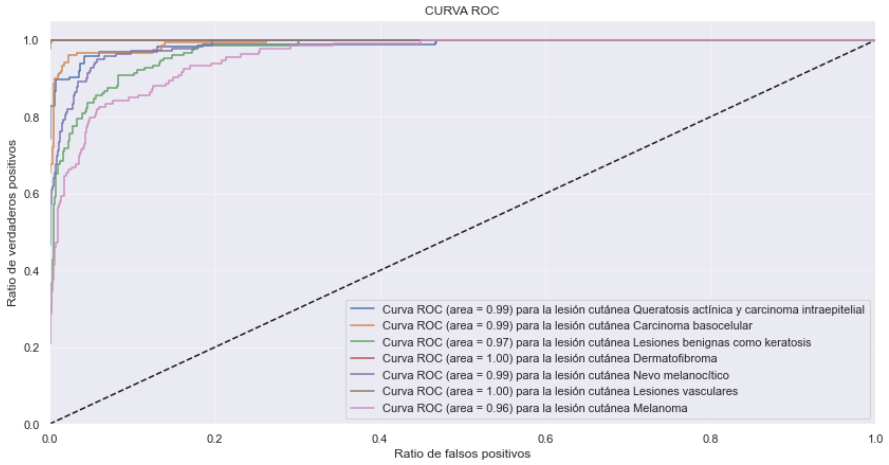


Figura 40. Curva ROC de la red EfficientNet B0

3.5.4.4.Matriz de confusión

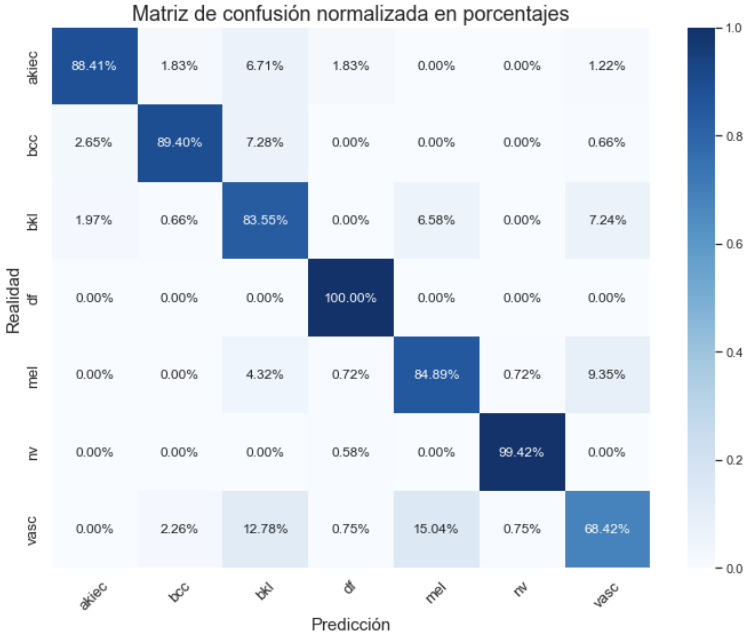


Figura 41. Matriz de confusión de la red EfficientNet B0

3.5.4.5.Métricas

Accuracy	0.88190
Precision	0.87893
Recall	0.87729
F1 Score	0.87726

Tabla 15. Métricas de la red EfficientNet B0

3.5.5. Red Xception

Los entrenamientos se realizan con valores 8, 16 y 32, para el tamaño de lote y tasas de aprendizaje 0.001, 0.0005 0.0001.

3.5.5.1. Historial de entrenamientos

El mejor modelo se obtiene con tamaño del lote 32 y tasa de aprendizaje 0.0005.

	epochs	batch_size	learning_rate	loss	accuracy	val_loss	val_accuracy	elapsed_time
7	50	32	0.0005	0.079068	0.971429	0.364875	0.902857	2293.880051
4	50	16	0.0005	0.143341	0.951667	0.341740	0.897143	3168.014806
2	50	8	0.0001	0.093192	0.971429	0.411656	0.896190	2924.242836
6	50	32	0.0010	0.233717	0.914524	0.331691	0.888571	2998.727470
8	50	32	0.0001	0.132439	0.957143	0.372989	0.888571	1920.991433
3	50	16	0.0010	0.298789	0.890476	0.372342	0.880000	3149.690727
5	50	16	0.0001	0.198352	0.939286	0.446824	0.879048	1681.203207
1	50	8	0.0005	0.288660	0.886667	0.383360	0.871429	5152.164802
0	50	8	0.0010	0.402021	0.853571	0.481697	0.827619	6313.412219

Tabla 16. Tabla con el historial de entrenamientos de la red Xception

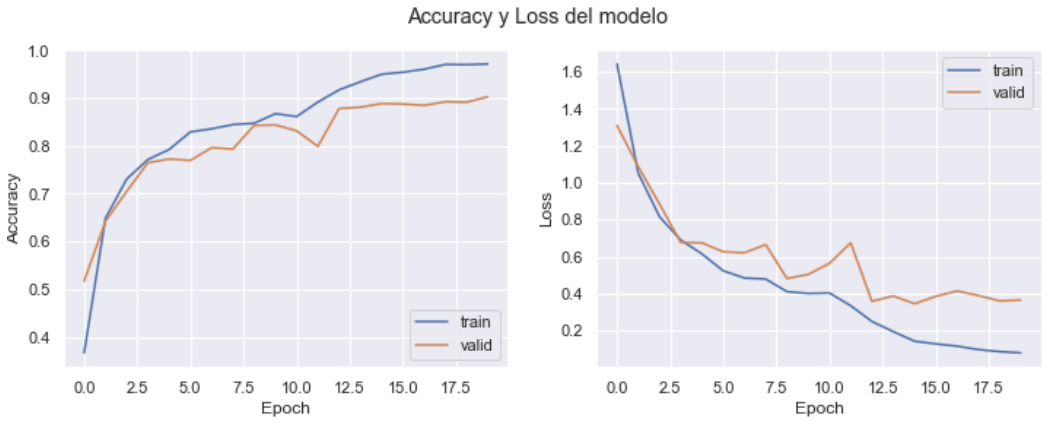


Figura 42. Gráficas de *accuracy* y *loss* de la red Xception

3.5.5.2. Evaluación del modelo

```
# Evaluación del modelo
model = build_model_Xception()
model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.load_weights(input_path + 'best_model5.h5')

loss, acc = model.evaluate(X_test, y_test, verbose = 2) # Evaluación con el conjunto de test
```

```
33/33 - 7s - loss: 0.5018 - accuracy: 0.8848 - 7s/epoch - 200ms/step
```

Figura 43. Evaluación del modelo de la red Xception

3.5.5.3.Curva ROC

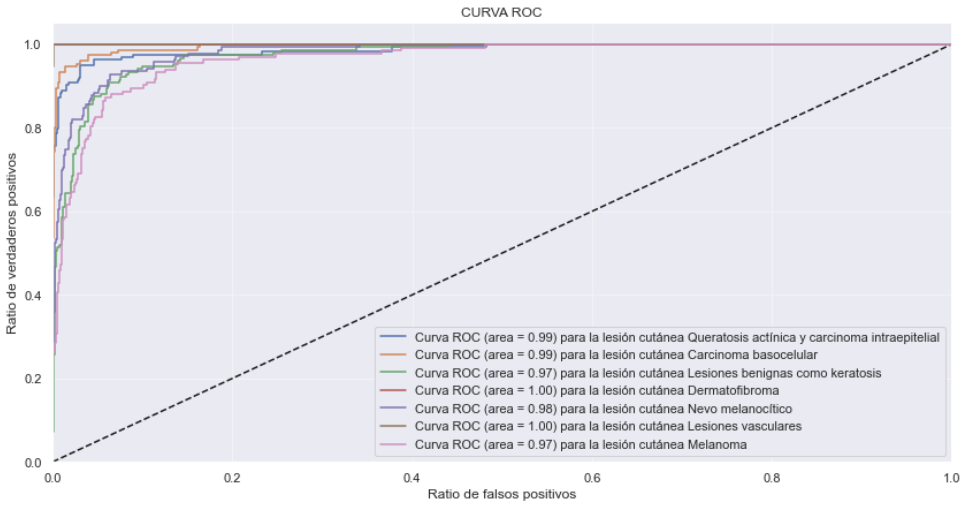


Figura 44. Curva ROC de la red Xception

3.5.5.4.Matriz de confusión

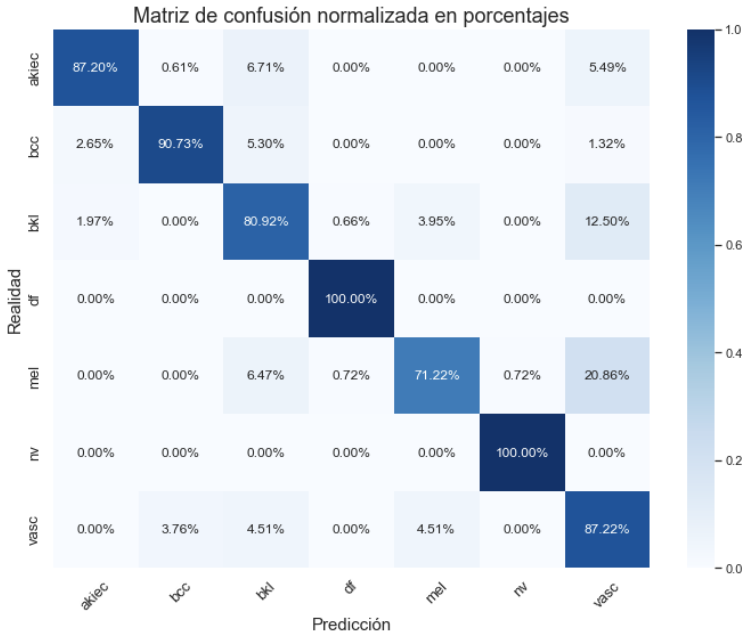


Figura 45. Matriz de confusión de la red Xception

3.5.5.5.Métricas

Accuracy	0.88476
Precision	0.88995
Recall	0.88184
F1 Score	0.88202

Tabla 17. Métricas de la red Xception

3.5.6. Red ResNet152

Los entrenamientos se realizan con valores 8 y 16 para el tamaño de lote y valores para la tasa de aprendizaje de 0.0005, 0.0001 y 0.00005.

3.5.6.1.Historial de entrenamientos

El mejor modelo se obtiene con tamaño del lote 16 y tasa de aprendizaje 0.0001.

	epochs	batch_size	learning_rate	loss	accuracy	val_loss	val_accuracy	elapsed_time
4	50	16	0.00010	0.072135	0.975714	0.357049	0.907619	4923.673581
1	50	8	0.00010	0.175007	0.937619	0.387488	0.892381	4075.349552
5	50	16	0.00005	0.117711	0.959762	0.401971	0.890476	3035.855439
2	50	8	0.00005	0.163128	0.939048	0.381899	0.889524	3639.214969
3	50	16	0.00050	0.617679	0.776667	0.617344	0.788571	4578.164848
0	50	8	0.00050	0.662315	0.757143	0.664039	0.756190	6214.434318

Tabla 18. Tabla con el historial de entrenamientos de la red ResNet152

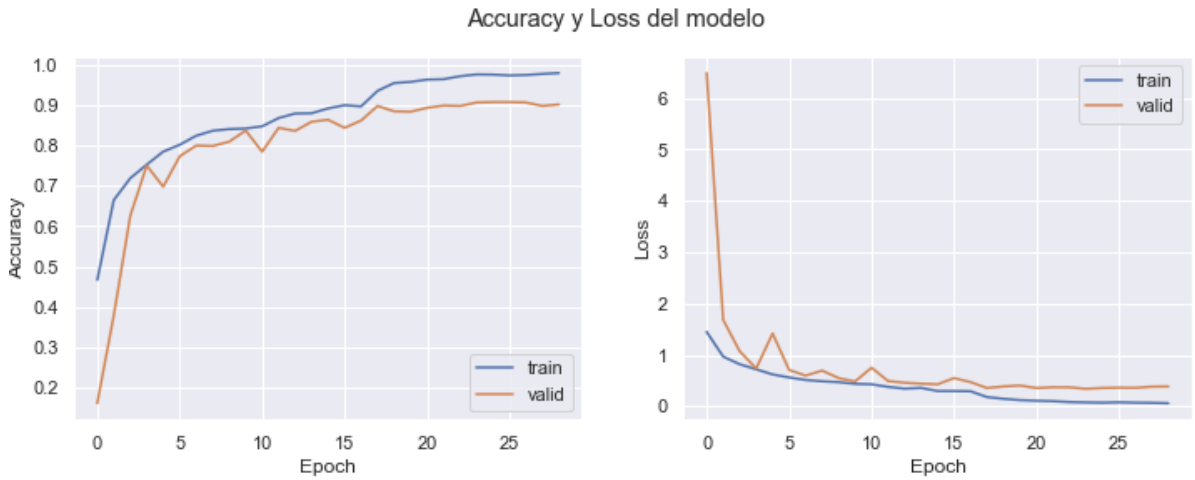


Figura 46. Gráficas de *accuracy* y *loss* de la red ResNet152

3.5.6.2.Evaluación del modelo

```
# Evaluación del modelo
model = build_model_ResNet152()
model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.load_weights(input_path + 'best_model6.h5')
model.save(input_path + 'model6.h5') # guardado del modelo completo

loss, acc = model.evaluate(X_test, y_test, verbose = 2) # Evaluación con el conjunto de test
```

```
33/33 - 15s - loss: 0.5168 - accuracy: 0.8829 - 15s/epoch - 466ms/step
```

Figura 47. Evaluación del modelo de la red ResNet152

3.5.6.3.Curva ROC

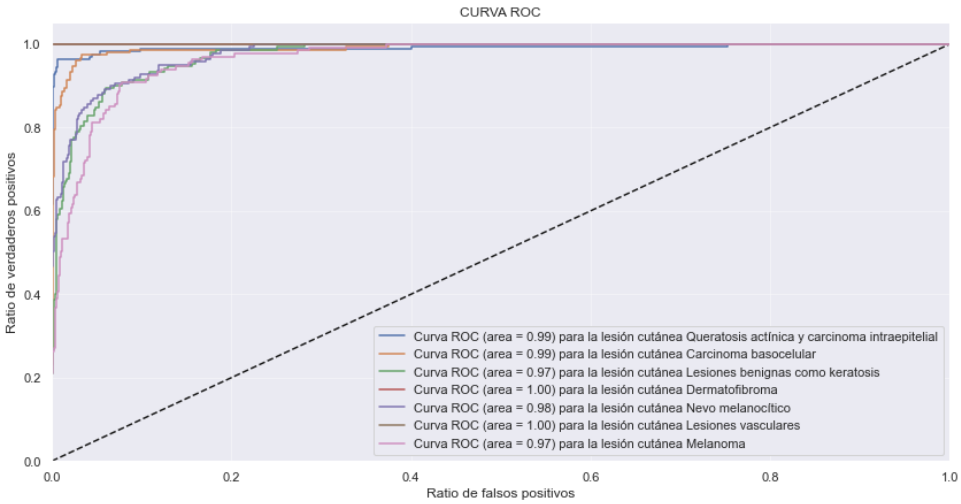


Figura 48. Curva ROC de la red ResNet152

3.5.6.4.Matriz de confusión

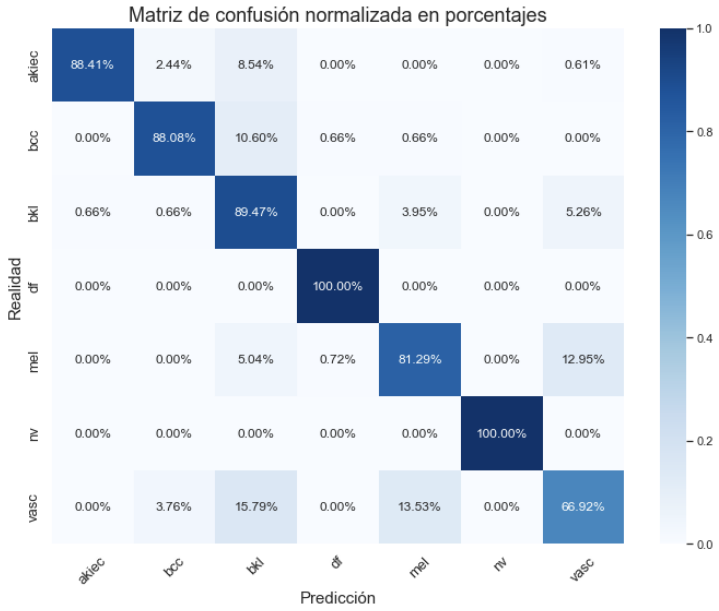


Figura 49. Matriz de confusión de la red ResNet152

3.5.6.5.Métricas

Accuracy	0.88286
Precision	0.88518
Recall	0.87740
F1 Score	0.87857

Tabla 19. Métricas de la red ResNet152

3.5.7. Red VGG16

Los entrenamientos se realizan con valores: 8, 16 y 32 para el tamaño de lote y valores para la tasa de aprendizaje: 0.001, 0.0005 y 0.0001.

3.5.7.1.Historial de entrenamientos

El mejor modelo se obtiene con un tamaño del lote de 16 y tasa de aprendizaje 0.0001.

	epochs	batch_size	learning_rate	loss	accuracy	val_loss	val_accuracy	elapsed_time
5	50	16	0.0001	0.342395	0.878571	0.522015	0.828571	2466.009813
8	50	32	0.0001	0.440644	0.836429	0.576473	0.812381	2785.747329
2	50	8	0.0001	0.404772	0.851905	0.570240	0.796191	3422.108115
6	50	32	0.0010	1.984820	0.152381	1.950592	0.162857	557.877440
0	50	8	0.0010	1.981782	0.138571	1.946490	0.133333	623.996815
1	50	8	0.0005	1.963319	0.140476	1.946393	0.133333	620.249218
4	50	16	0.0005	1.955907	0.138333	1.946153	0.133333	509.341248
3	50	16	0.0010	2.330448	0.144048	1.946034	0.132381	510.046019
7	50	32	0.0005	1.948364	0.145238	1.946766	0.126667	477.214018

Tabla 20. Tabla con el historial de entrenamientos de la red VGG16

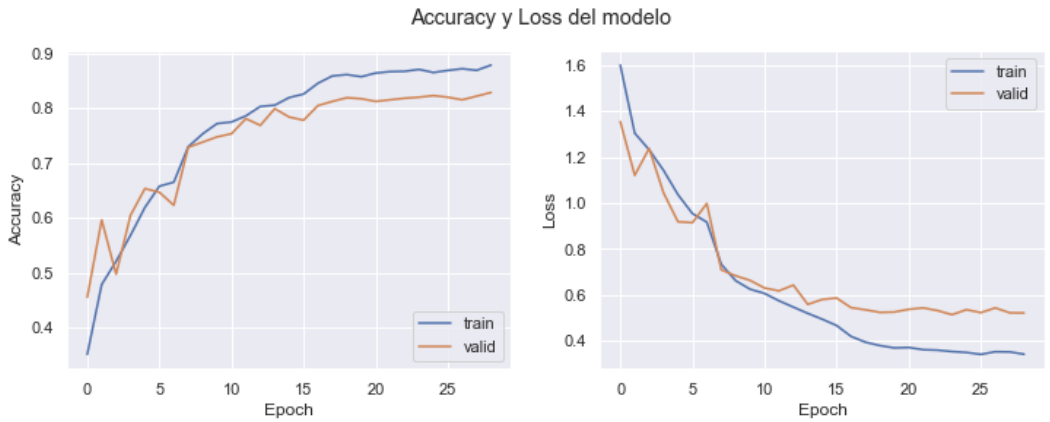


Figura 50. Gráficas de *accuracy* y *loss* de la red VGG16

3.5.7.2.Evaluación del modelo

```
# Evaluación del modelo
model = build_model_VGG16()
model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.load_weights(input_path + 'best_model7.h5')
model.save(input_path + 'model7.h5') # guardado del modelo completo

loss, acc = model.evaluate(X_test, y_test, verbose = 2) # Evaluación con el conjunto de test
```

```
33/33 - 7s - loss: 0.5479 - accuracy: 0.8105 - 7s/epoch - 197ms/step
```

Figura 51. Evaluación del modelo de la red VGG16

3.5.7.3.Curva ROC

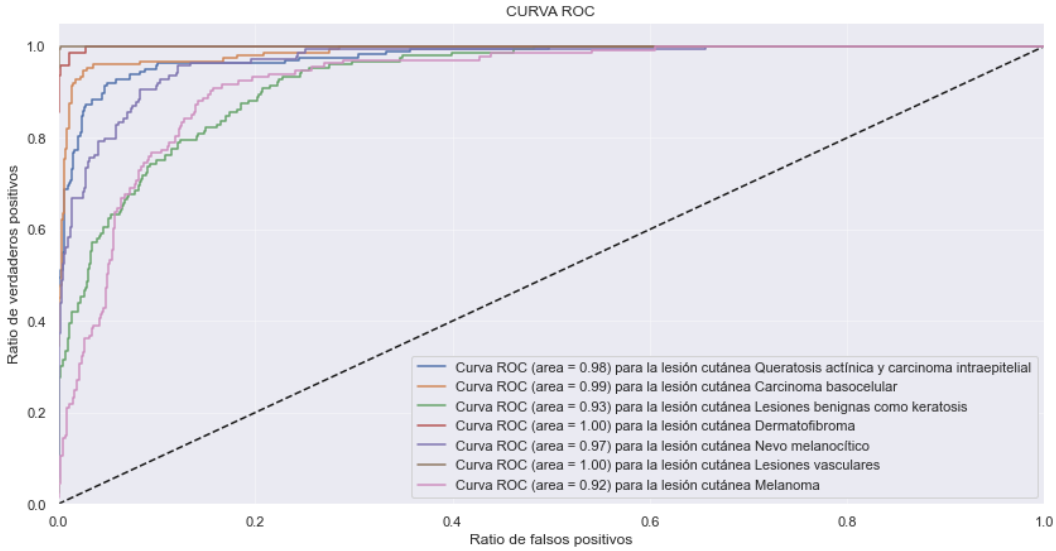


Figura 52. Curva ROC de la red VGG16

3.5.7.4.Matriz de confusión

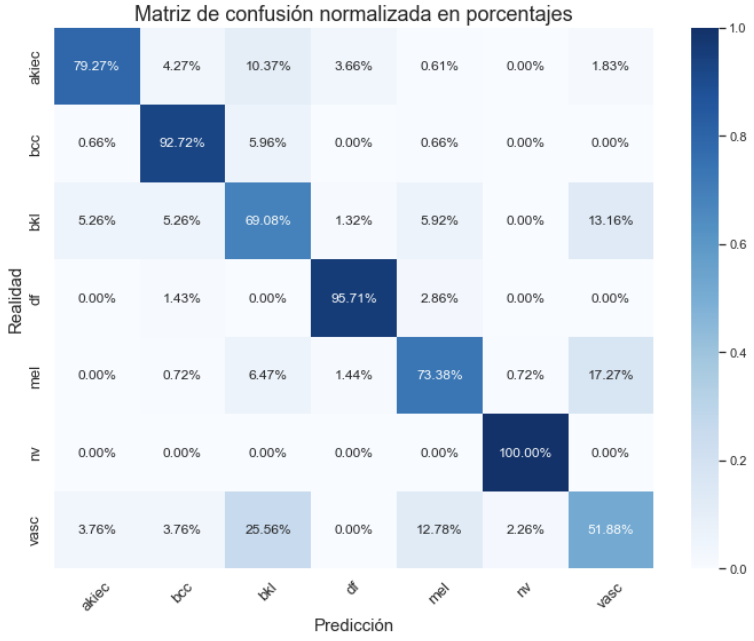


Figura 53. Matriz de confusión de la red VGG16

3.5.7.5.Métricas

Accuracy	0.81048
Precision	0.80412
Recall	0.80291
F1 Score	0.80195

Tabla 21. Métricas de la red VGG16

3.5.8. Red DenseNet201

Los entrenamientos se realizan con valores: 8, 12 y 16 para el tamaño de lote y valores para la tasa de aprendizaje: 0.001, 0.0005 y 0.0001.

3.5.8.1.Historial de entrenamientos

El mejor modelo se obtiene con tamaño del lote 12 y tasa de aprendizaje 0.0001.

	epochs	batch_size	learning_rate	loss	accuracy	val_loss	val_accuracy	elapsed_time
5	50	12	0.0001	0.076687	0.973810	0.329308	0.915238	2624.677022
2	50	8	0.0001	0.147016	0.950714	0.378072	0.901905	2855.050168
8	50	16	0.0001	0.271703	0.908095	0.472238	0.870476	1653.618476
4	50	12	0.0005	0.436481	0.828333	0.445523	0.857143	4429.958042
1	50	8	0.0005	0.482882	0.812381	0.505829	0.831429	6662.898491
7	50	16	0.0005	0.545795	0.786190	0.497713	0.823810	3055.969200
3	50	12	0.0010	0.882077	0.649048	0.733701	0.744762	5374.358230
0	50	8	0.0010	1.099187	0.557381	0.987330	0.636190	5236.376637
6	50	16	0.0010	1.961942	0.147143	1.946099	0.133333	725.986115

Tabla 22. Tabla con el historial de entrenamientos de la red DenseNet201

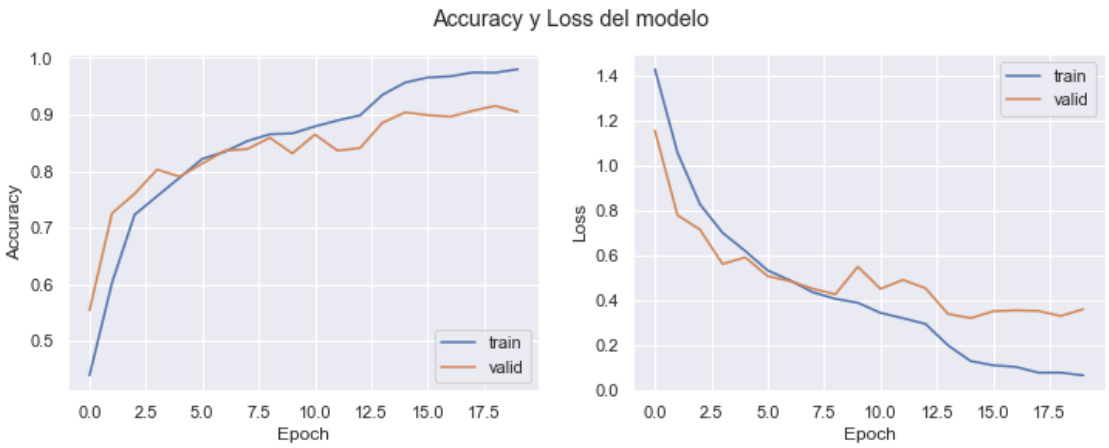


Figura 54. Gráficas de *accuracy* y *loss* de la red DenseNet201

3.5.8.2.Evaluación del modelo

```
# Evaluación del modelo
model = build_model_DenseNet201()
model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.load_weights(input_path + 'best_model8.h5')
model.save(input_path + 'model8.h5') # guardado del modelo completo

loss, acc = model.evaluate(X_test, y_test, verbose = 2) # Evaluación con el conjunto de test
```

```
33/33 - 11s - loss: 0.3711 - accuracy: 0.9019 - 11s/epoch - 337ms/step
```

Figura 55. Evaluación del modelo de la red DenseNet201

3.5.8.3.Curva ROC

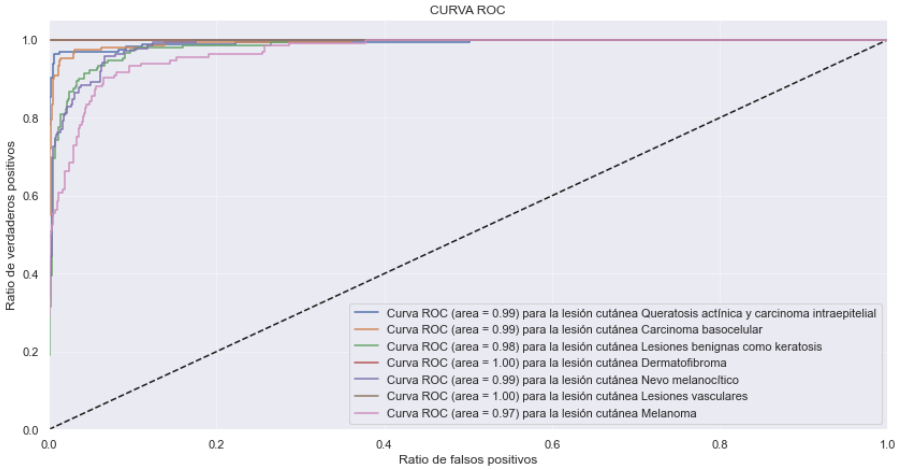


Figura 56. Curva ROC de la red DenseNet201

3.5.8.4.Matriz de confusión

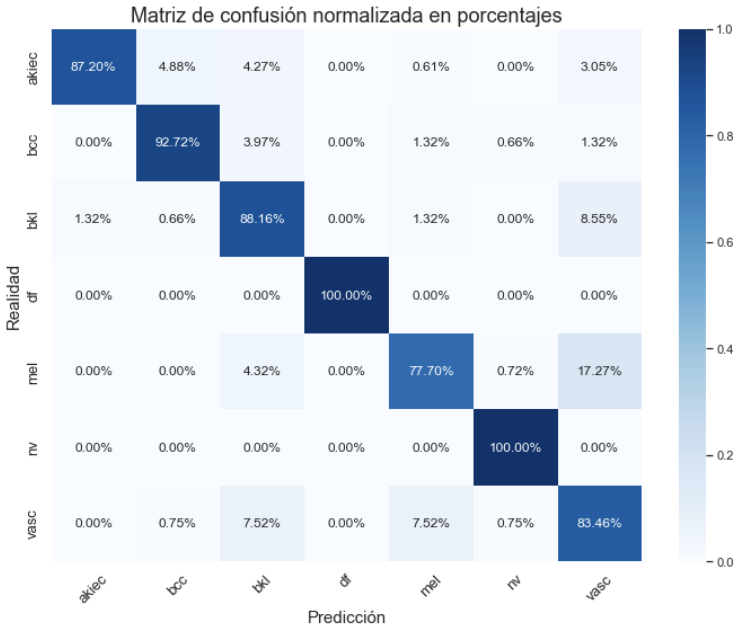


Figura 57. Matriz de confusión de la red DenseNet201

3.5.8.5.Métricas

Accuracy	0.90190
Precision	0.90265
Recall	0.89889
F1 Score	0.89902

Tabla 23. Métricas de la red DenseNet201

3.5.9. Red MobileNetV2

Los entrenamientos se realizan con valores: 8, 16 y 32 para el tamaño de lote y valores para la tasa de aprendizaje: 0.001, 0.0005 y 0.0001.

3.5.9.1.Historial de entrenamientos

El mejor modelo se obtiene con tamaño del lote 8 y tasa de aprendizaje 0.0001.

	epochs	batch_size	learning_rate	loss	accuracy	val_loss	val_accuracy	elapsed_time
2	50	8	0.0001	0.221844	0.921429	0.372402	0.880952	1972.402670
5	50	16	0.0001	0.220308	0.925714	0.422564	0.880000	1535.868850
4	50	16	0.0005	0.251036	0.911429	0.452651	0.857143	1050.722956
1	50	8	0.0005	0.450207	0.827619	0.516795	0.822857	1358.170453
8	50	32	0.0001	0.543555	0.800952	1.596956	0.572381	316.735355
7	50	32	0.0005	0.670966	0.775000	3.967993	0.453333	318.182488
6	50	32	0.0010	1.469831	0.460238	5.620605	0.320952	315.825088
0	50	8	0.0010	1.306824	0.498333	2.660771	0.293333	407.726113
3	50	16	0.0010	1.462675	0.463571	4.389105	0.205714	373.559875

Tabla 24. Tabla con el historial de entrenamientos de la red MobileNetV2

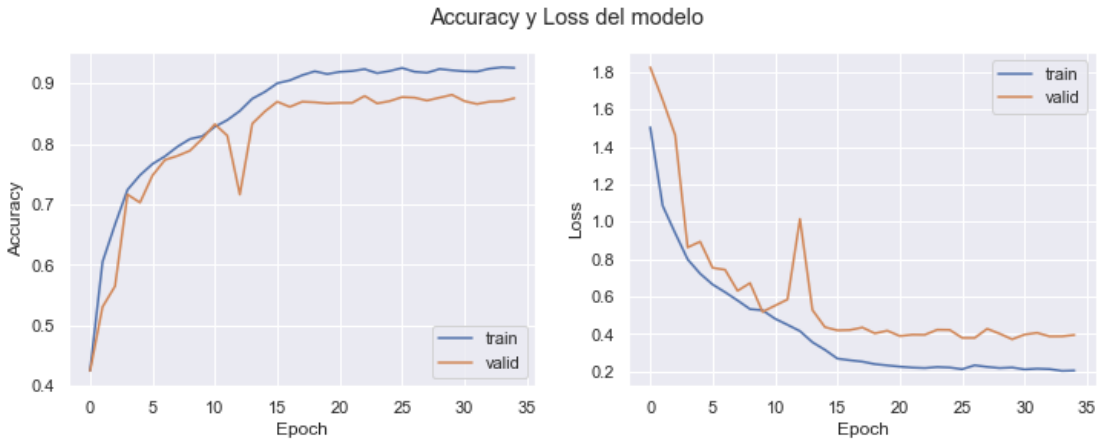


Figura 58. Gráficas de *accuracy* y *loss* de la red MobileNetV2

3.5.9.2.Evaluación del modelo

```
# Evaluación del modelo
model = build_model_MobileNetV2()
model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.load_weights(input_path + 'best_model9.h5')
model.save(input_path + 'model9.h5') # guardado del modelo completo

loss, acc = model.evaluate(X_test, y_test, verbose = 2) # Evaluación con el conjunto de test
```

```
33/33 - 11s - loss: 0.8296 - accuracy: 0.7838 - 11s/epoch - 335ms/step
```

Figura 59. Evaluación del modelo de la red MobileNetV2

3.5.9.3.Curva ROC

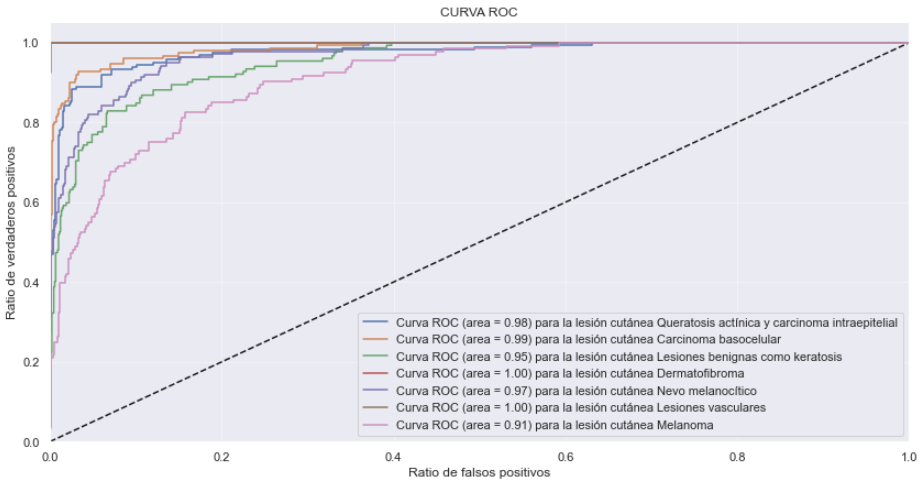


Figura 60. Curva ROC de la red MobileNetV2

3.5.9.4.Matriz de confusión

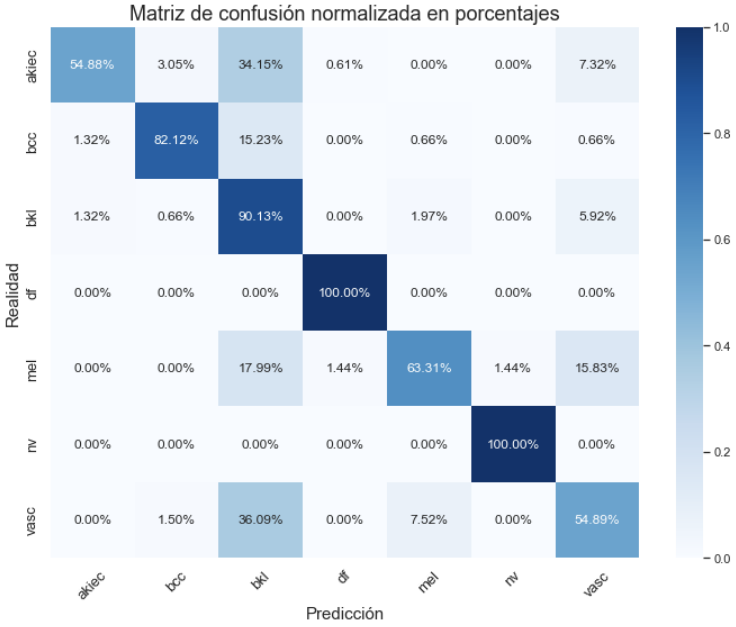


Figura 61. Matriz de confusión de la red MobileNetV2

3.5.9.5.Métricas

Accuracy	0.78381
Precision	0.83215
Recall	0.77904
F1 Score	0.78474

Tabla 25. Métricas de la red MobileNetV2

3.5.10. Red InceptionResNetV2

Los entrenamientos se realizan con valores: 8, 16 y 32 para el tamaño del lote y valores para la tasa de aprendizaje: 0.001, 0.0005 y 0.0001.

3.5.10.1.Historial de entrenamientos

El mejor modelo se obtiene con tamaño del lote 16 y tasa de aprendizaje 0.0005.

	epochs	batch_size	learning_rate	loss	accuracy	val_loss	val_accuracy	elapsed_time
4	50	16	0.0005	0.183448	0.935714	0.362152	0.894286	3781.938641
5	50	16	0.0001	0.173040	0.941429	0.551845	0.882857	1577.189186
7	50	32	0.0005	0.239610	0.918571	0.381099	0.882857	1873.608352
8	50	32	0.0001	0.064099	0.981429	0.555833	0.880000	1873.814818
6	50	32	0.0010	0.268286	0.904524	0.384652	0.879048	4721.218711
3	50	16	0.0010	0.232047	0.915952	0.424880	0.876190	4164.725408
1	50	8	0.0005	0.257744	0.905000	0.420164	0.868571	4360.199608
2	50	8	0.0001	0.423480	0.860714	0.737122	0.819048	1518.359570
0	50	8	0.0010	0.561571	0.780476	0.556510	0.805714	5079.191411

Tabla 26. Tabla con el historial de entrenamientos de la red InceptionResNetV2

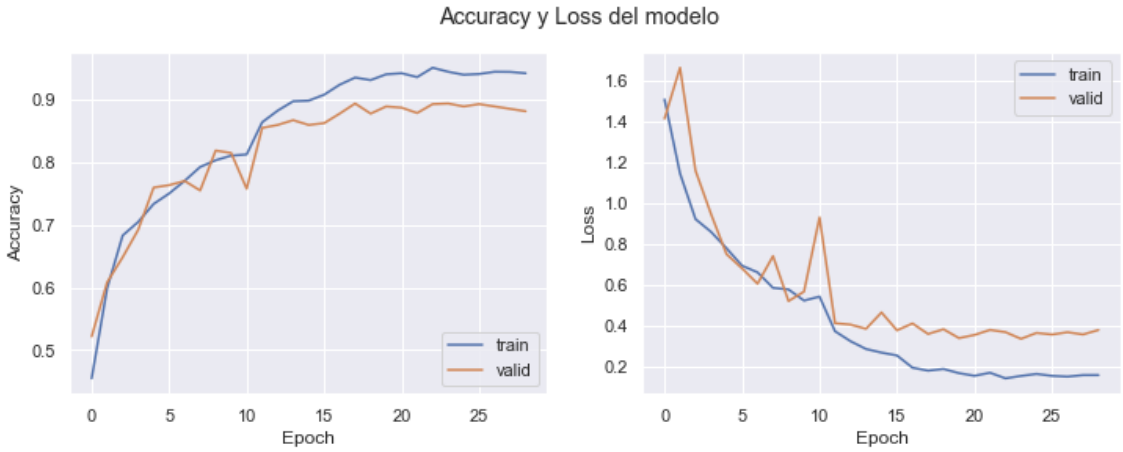


Figura 62. Gráficas de *accuracy* y *loss* de la red InceptionResNetV2

3.5.10.2.Evaluación del modelo

```
# Evaluación del modelo
model = build_model_InceptionResNetV2()
model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.load_weights(input_path + 'best_model10.h5')
model.save(input_path + 'model10.h5') # guardado del modelo completo

loss, acc = model.evaluate(X_test, y_test, verbose = 2) # Evaluación con el conjunto de test
```

```
33/33 - 13s - loss: 0.4437 - accuracy: 0.8771 - 13s/epoch - 396ms/step
```

Figura 63. Evaluación del modelo de la red InceptionResNetV2

3.5.10.3.Curva ROC

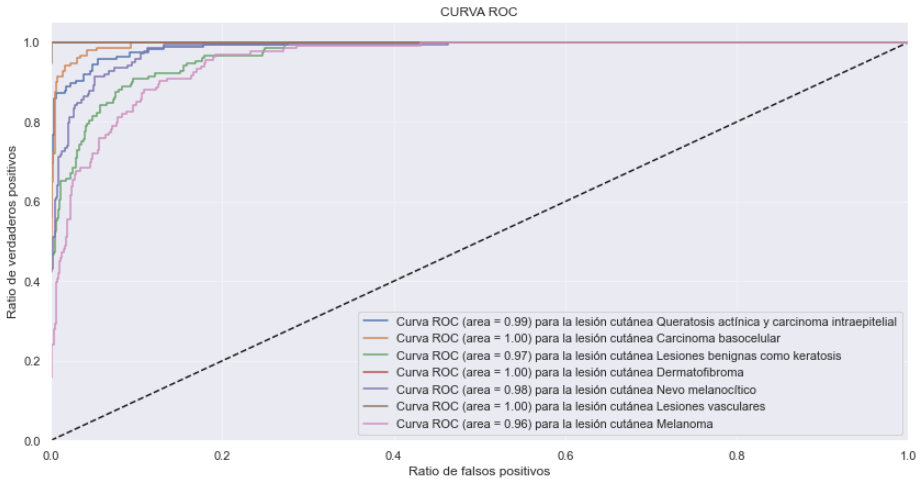


Figura 64. Curva ROC de la red InceptionResNetV2

3.5.10.4.Matriz de confusión

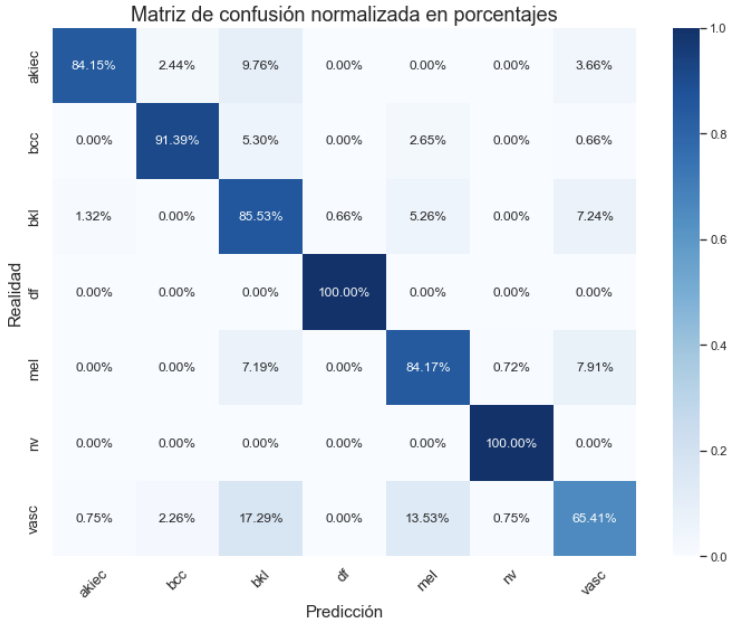


Figura 65. Matriz de confusión de la red InceptionResNetV2

3.5.10.5.Métricas

Accuracy	0.87714
Precision	0.87899
Recall	0.87236
F1 Score	0.87313

Tabla 27. Métricas de la red InceptionResNetV2

3.5.11. Red *dummy*

Esta red *dummy* ha sido creada con el único objetivo de comparar el desempeño de las redes bien entrenadas y ajustadas con una red neuronal mal entrenada. La arquitectura de este modelo de red es el mismo que el de la CNN *adhoc*, pero en esta ocasión la red ha sido entrenada durante solo 2 épocas, con tamaño del lote de 128 y tasa de aprendizaje de 0.0001.

Tal y como se puede observar en el historial de entrenamiento, la exactitud durante la validación es tan solo del 32,44% y la evaluación del modelo devuelve solo el 27,93% de *accuracy*. La curva ROC, la matriz de confusión y las métricas dejan constancia del mal desempeño de esta red, que no podría ser utilizada para clasificar lesiones cutáneas utilizando imágenes.

3.5.11.1.Historial de entrenamientos

	epochs	batch_size	learning_rate	loss	accuracy	val_loss	val_accuracy	elapsed_time
0	2	128	0.00001	1.810083	0.300353	1.740995	0.324368	8.004126

Tabla 28. Tabla con el historial de los entrenamientos de la red *dummy*

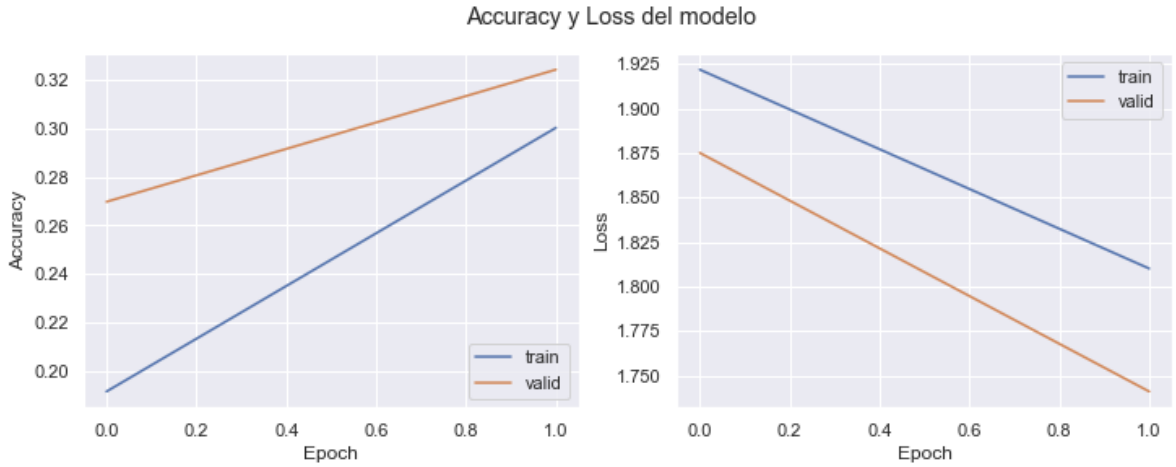


Figura 66. Gráficas de *accuracy* y *loss* de la red *dummy*

3.5.11.2.Evaluación del modelo

```
# Evaluación del modelo
model = build_model_cnn_adhoc()
model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.load_weights(input_path + 'cnn_adhoc_dummy_best_model.h5')
model.save(input_path + 'model_dummy.h5') # Guardado completo del modelo

loss, acc = model.evaluate(X_test, y_test, verbose = 2)   # Evaluación del modelo con el conjunto de test
```

```
294/294 - 2s - loss: 1.7430 - accuracy: 0.2793 - 2s/epoch - 6ms/step
```

Figura 67. Evaluación del modelo de la red *dummy*

3.5.11.3.Curva ROC

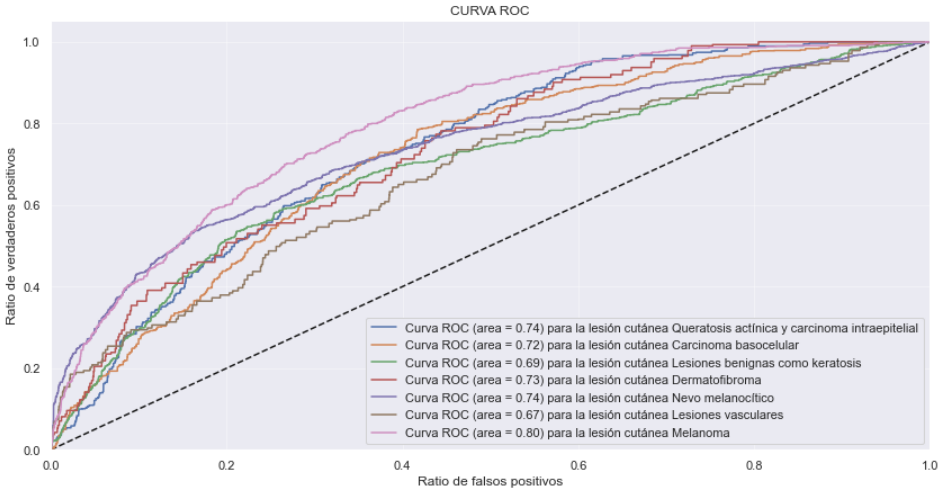


Figura 68. Cruva ROC de la red *dummy*

3.5.11.4.Matriz de confusión

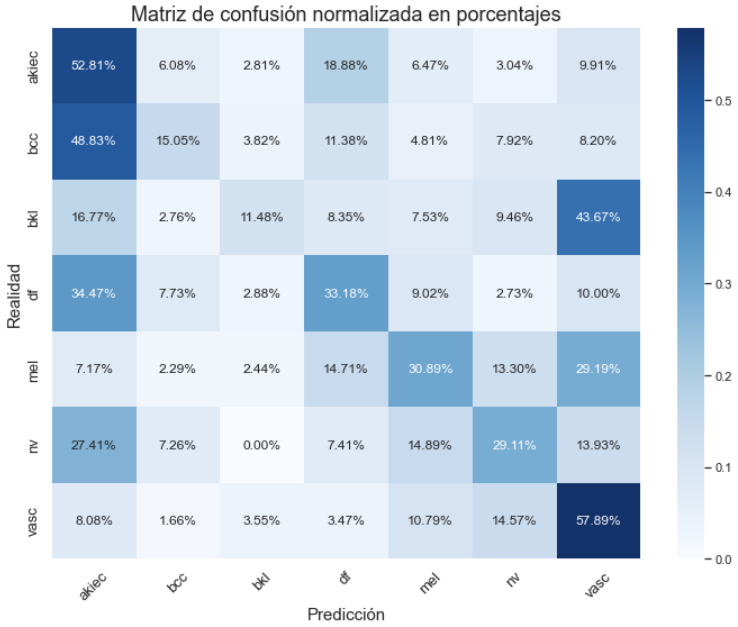


Figura 69. Matriz de confusión de la red *dummy*

3.5.11.5.Métricas

Accuracy	0.32598
Precision	0.35034
Recall	0.32916
F1 Score	0.30818

Figura 70. Tabla con las métricas de la red *dummy*

3.6. Inteligencia colectiva

3.6.1. Función de clasificación

Una vez que se han entrenado y ajustado las 10 redes neuronales, se procede a construir la inteligencia colectiva. Para ello, se insertan en la matriz de pesos detallada en el punto 2.6 de este documento, los valores de las exactitudes obtenidas en las matrices de confusión de cada una de las redes con el conjunto de datos de prueba.

	C0	C1	C2	C3	C4	C5	C6
0	1.00000	1.00000	0.97466	1.00000	0.91057	1.00000	0.97962
1	0.92683	0.96689	0.82237	1.00000	0.75540	1.00000	0.81203
2	0.94512	0.93377	0.71711	1.00000	0.76259	1.00000	0.75188
3	0.88415	0.89404	0.83553	1.00000	0.84892	0.99415	0.68421
4	0.87195	0.90728	0.80921	1.00000	0.71223	1.00000	0.87218
5	0.88415	0.88079	0.89474	1.00000	0.81295	1.00000	0.66917
6	0.79268	0.92715	0.69079	0.95714	0.73381	1.00000	0.51880
7	0.87195	0.92715	0.88158	1.00000	0.77698	1.00000	0.83459
8	0.54878	0.82119	0.90132	1.00000	0.63309	1.00000	0.54887
9	0.84146	0.91391	0.85526	1.00000	0.84173	1.00000	0.65414

Tabla 29. Matriz de pesos de las redes para todas las clases

A continuación, se crea la función de nombre ic_class_image, que recibe como parámetro la ruta completa de una imagen. Dicha imagen es clasificada por cada una de las 10 redes que forman la inteligencia colectiva. Se insertan en la matriz de probabilidades los valores obtenidos para cada una de las 7 clases por cada una de las 10 redes.

	C0	C1	C2	C3	C4	C5	C6
0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000
1	0.01015	0.00586	0.05173	0.02184	0.01631	0.02050	0.87361
2	0.00000	0.00000	0.00003	0.00000	0.00000	0.00000	0.99996
3	0.00005	0.00003	0.26630	0.00000	0.04381	0.00000	0.68981
4	0.00000	0.00000	0.00024	0.00000	0.00882	0.00000	0.99093
5	0.00000	0.00000	0.03351	0.00000	0.00915	0.00000	0.95734
6	0.00145	0.00023	0.89042	0.00012	0.02161	0.00000	0.08616
7	0.00000	0.00000	0.00000	0.00000	0.00007	0.00000	0.99992
8	0.00053	0.00007	0.04191	0.00000	0.00767	0.00001	0.94980
9	0.00010	0.00102	0.72712	0.00001	0.16412	0.00003	0.10760

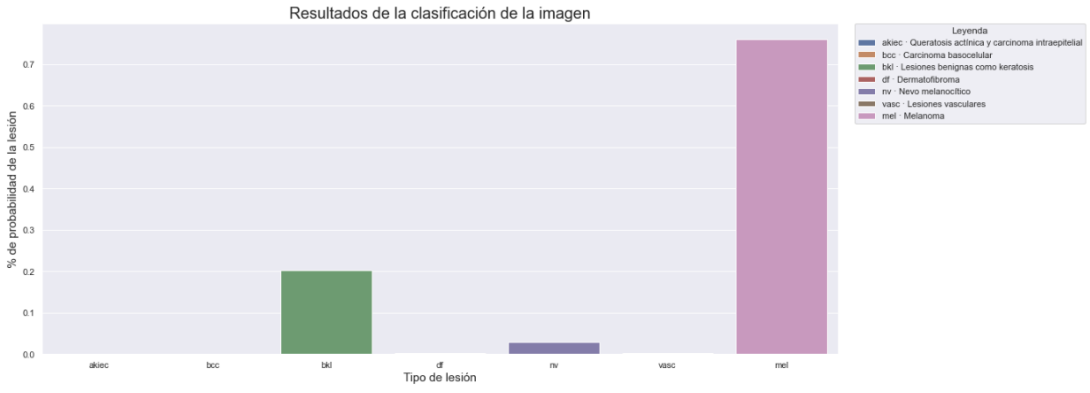
Tabla 30. Matriz de probabilidades de la clasificación de una imagen con la IC

El siguiente paso que realiza la función, es la multiplicación de la matriz de probabilidades por la matriz de pesos, obteniendo clasificaciones ponderadas en base al desempeño de cada red para clasificar cada uno de los tipos de lesiones cutáneas.

	C0	C1	C2	C3	C4	C5	C6
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.979620
1	0.009407	0.005666	0.042541	0.021840	0.012321	0.02050	0.709398
2	0.000000	0.000000	0.000022	0.000000	0.000000	0.00000	0.751850
3	0.000044	0.000027	0.222502	0.000000	0.037191	0.00000	0.471975
4	0.000000	0.000000	0.000194	0.000000	0.006282	0.00000	0.864269
5	0.000000	0.000000	0.029983	0.000000	0.007438	0.00000	0.640623
6	0.001149	0.000213	0.615093	0.000115	0.015858	0.00000	0.044700
7	0.000000	0.000000	0.000000	0.000000	0.000054	0.00000	0.834523
8	0.000291	0.000057	0.037774	0.000000	0.004856	0.00001	0.521317
9	0.000084	0.000932	0.621877	0.000010	0.138145	0.00003	0.070385

Tabla 31. Matriz con las clasificaciones ponderadas

Por último, se calcula la clase predicha por la IC, como aquella que la suma de los valores asignados por las redes es el más grande. También se obtiene la certeza del resultado, calculada como la media de los porcentajes de probabilidad de las redes que predijeron la misma clase. La función devuelve un vector con las probabilidades obtenidas por la IC para cada lesión cutánea. Este vector tiene por objetivo la representación visual de los resultados en un gráfico de barras.




```
Tipo de lesión: mel · Melanoma. Acierto: 74.00%
```

Figura 71. Resultados de la clasificación de una imagen por la IC con el % de probabilidad de cada clase

3.6.2. Prueba de la IC

Para probar el desempeño de la inteligencia colectiva, se seleccionan 100 imágenes aleatorias de cada clase y se clasifican con la función de clasificación de la IC. Con el

objetivo de saber si la IC mejora los resultados obtenidos por las redes clasificando individualmente, se guardan en una tabla la exactitud de cada red y el obtenido por la IC.

Se verifica que la exactitud de la IC supera en más de 3 puntos porcentuales a la de la red que mejor desempeño tiene, el modelo 3 · Swin Transformer, y en más de 16 puntos a la que peor exactitud tiene, el modelo 9 · MobileNetV2.

	Red	Accuracy
0	Model 1 · CCN Adhoc	0.777143
1	Model 2 · ViT-B16	0.824286
2	Model 3 · Swin Transformer	0.887143
3	Model 4 · EfficientNet B0	0.825714
4	Model 5 · Xception	0.820000
5	Model 6 · ResNet152	0.861429
6	Model 7 · VGG16	0.785714
7	Model 8 · DenseNet201	0.872857
8	Model 9 · MobileNetV2	0.757143
9	Model 10 · InceptionResNetV2	0.835714
10	IC · Inteligencia Colectiva	0.918571

Tabla 32. *Accuracy* de la IC y de las 10 redes con un conjunto de 100 imágenes aleatorias de cada clase

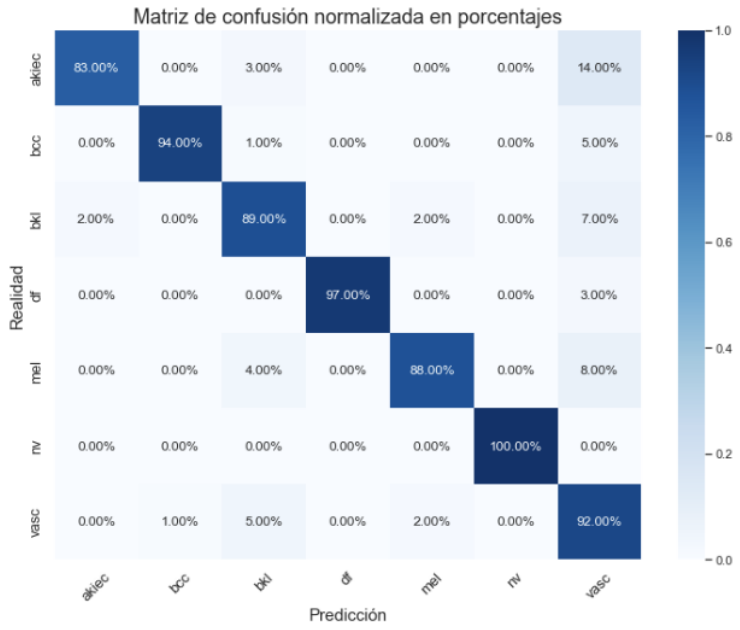


Figura 72. Matriz de confusión de la IC con un conjunto de 100 imágenes aleatorias de cada clase

Accuracy	0.91857
Precision	0.92974
Recall	0.91857
F1 Score	0.92107

Tabla 33. Métricas de la IC con un conjunto de 100 imágenes de cada clase

3.7. Aplicación

El último paso de este trabajo es poner el clasificador IC como servicio, con el objetivo de que otros usuarios puedan utilizarlo para clasificar otras imágenes. Para ello, se consideran dos enfoques: servicio web y servicio API. El primero permite realizar clasificaciones manuales de una en una, mientras que el segundo, otorga la funcionalidad de realizar clasificaciones masivas que se pueden automatizar mediante scripts.

Tanto la aplicación web como el servicio API, se implementan con Flask, que es un *framework* que permite desarrollar aplicaciones en servidor en lenguaje Python.

3.7.1. Web

Se diseña una web sencilla con el objetivo de facilitar su uso. La página principal únicamente dispone de un título y un botón que permite seleccionar una imagen nueva. Al pulsarlo, abre el explorador de archivos y el usuario busca y selecciona la imagen.

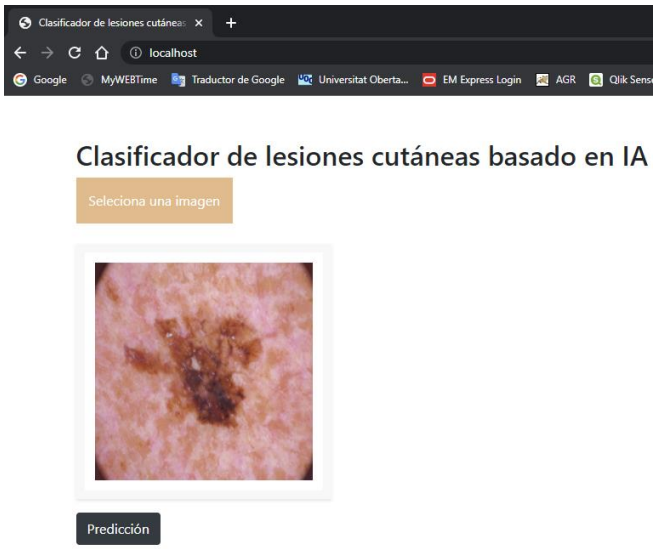


Figura 73. Aplicación web con la previsualización de la imagen seleccionada por el usuario

A continuación, se hace visible el botón “Predicción”, que es el que hace la llamada a la función de clasificación de imagen de la IC. En aproximadamente 3 segundos, se obtiene la respuesta de la clasificación, que consta de un gráfico de barras con la probabilidad asignada a cada clase y un texto con la descripción del tipo de lesión cutánea predicho y el porcentaje de acierto del resultado.

A la aplicación se puede acceder localmente en la url: http://localhost o remotamente mediante la url: http://{IP_o_hostname}. En el directorio raíz de la APP hay un fichero README con las instrucciones para la instalación del software, configuración del entorno y uso de la aplicación web.

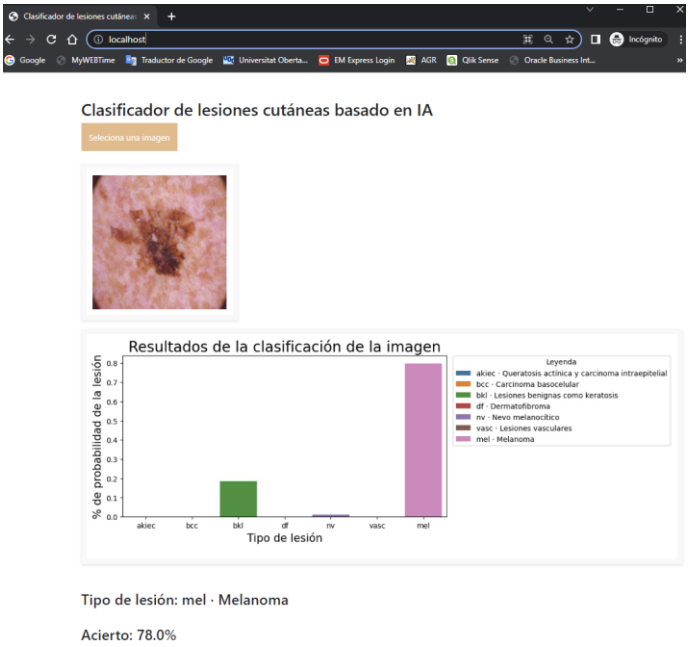


Figura 74. Aplicación web con el resultado de la clasificación de una imagen

3.7.2. API

La API está implementada para atender solicitudes GET y POST, en las que recibe una imagen como entrada y devuelve, en formato JSON, el nombre del fichero con la imagen (image_name), el porcentaje de acierto de la IC (ic_accuracy), el identificador corto del tipo de lesión (lesion_skin_cod), la descripción del tipo de lesión (lesion_skin_desc) y un vector con las probabilidades asignadas a cada una de las 7 clases (ic_prob).

Figura 75. Clasificaciones de imágenes a través de la API en SO Windows, Linux y MAC con Curl

3.7.3. Contenido de la aplicación

El Anexo 2 muestra el árbol de directorios y archivos de la aplicación y las descripciones de cada uno.

3.7.4. Clasificación de melanoma con API en imágenes nuevas

Con el objetivo de verificar la fiabilidad de la herramienta con datos totalmente nuevos y poniendo el foco en el melanoma, por ser la más peligrosa de las lesiones cutáneas tratadas por la IC, se descarga un nuevo conjunto de datos de ISIC, disponible en Kaggle (https://www.kaggle.com/datasets/nodoubttome/skin-cancer9-classesisic). El conjunto cuenta con un total de 2.357 imágenes que están agrupadas en los directorios Train y Test, y dentro de cada directorio, nueve directorios más, uno por cada tipo de lesión cutánea. De este conjunto, se han seleccionado 100 imágenes del directorio melanoma ubicado dentro del directorio Train y se han clasificado utilizando el siguiente comando desde una consola CMD de Windows en el directorio que contiene las imágenes:

for %a in (*) do curl localhost:80/api -F file=@%a >> melanoma_results.txt

El comando ejecuta la petición a la API por cada una de las imágenes del directorio y guarda el resultado en el fichero melanoma_results.txt. Una vez obtenidos los resultados de la API a las 100 peticiones, se cuentan los resultados clasificados como “melanoma” con el comando:

find /i "Melanoma" melanoma_results.txt | find "" /v /c

La IC ha clasificado correctamente 96 de las 100 (96% de acierto) imágenes de melanoma.

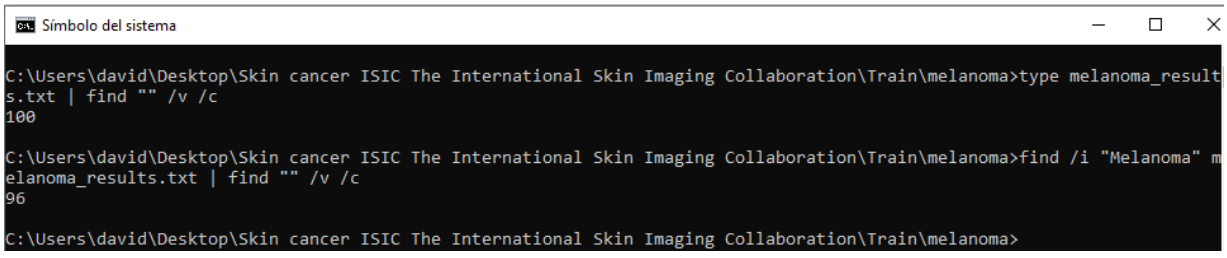


Figura 76. Recuento de imágenes clasificadas y clasificadas como melanoma

4. Conclusiones y trabajos futuros

En este trabajo se ha tratado un problema del ámbito de la medicina, concretamente, el de cáncer de piel. Al igual que con el resto de los cánceres, la probabilidad de sobrevivir a la enfermedad aumenta considerablemente cuando su detección es temprana.

Se han citado las formas que hay en la actualidad para diagnosticar el cáncer de piel. Dos de ellas son tempranas, rápidas de realizar y relativamente baratas, el auto chequeo y la inspección visual llevada a cabo por un especialista, pero ambas ofrecen porcentajes de acierto bajos, incluso cuando la inspección es realizada por especialistas con más de 15 años de experiencia el porcentaje de acierto es inferior al 70%, tal y como se ha visto en varios de los artículos revisados. El resto de las pruebas, como las biopsias, microscopio confocal o consenso de expertos, conllevan costes hospitalarios elevados, además de dilatar el tiempo hasta la obtención del diagnóstico final, algo que resta del porcentaje de probabilidad de supervivencia de los pacientes.

Se ha realizado una revisión sistemática de artículos recientes que han utilizado con éxito técnicas de inteligencia artificial, concretamente, de aprendizaje profundo (*deep learning*), para diagnosticar enfermedades mediante el uso de imágenes. Se ha visto que en los trabajos revisados los porcentajes de acierto de los modelos utilizados han superado el 85% de acierto en todos los casos, además de facilitar los resultados en apenas unos segundos, una vez que los modelos han sido entrenados y ajustados correctamente.

Se ha diseñado una red CNN desde cero y seleccionado nueve de las mejores redes neuronales preentrenadas con el conjunto de datos ImageNet que aparecen en los artículos revisados, dos de ellas del tipo transformer. Se ha seleccionado uno de los mejores conjuntos de datos de imágenes de lesiones cutáneas disponible públicamente en la actualidad, HAM10000. El conjunto de imágenes ha sido tratado debidamente para entrenar las redes neuronales. Para todas las redes neuronales se ha realizado la búsqueda de los mejores hiperparámetros, se han entrenado y evaluado su desempeño mediante el uso de las métricas adecuadas. Una vez que las diez redes neuronales han sido evaluadas y, dado los excelentes resultados obtenidos por todas ellas, se ha decidido crear una inteligencia colectiva, simulando un consenso de expertos. Se ha comparado el desempeño de la IC con el obtenido por cada una de las redes individualmente y se ha observado que es superior, con lo que se ha optado por usar la IC en lugar de una única red neuronal.

Como prueba adicional para confirmar el buen funcionamiento de la IC, se han seleccionado 100 imágenes aleatorias clasificadas como melanoma de un conjunto de imágenes de ISIC no visto antes por la IC, y ésta, ha sido capaz de clasificar correctamente 96 imágenes. Para las 4 imágenes clasificadas incorrectamente la IC ha aportado en los gráficos de resultados un porcentaje de probabilidad de melanoma significativo, que a ojos de un dermatólogo serían susceptibles de realizar pruebas adicionales para confirmar su diagnóstico, con lo que no quedarían como falsos negativos. Ese es uno de los motivos por los que la herramienta requiere que siempre sea utilizada por un médico especialista capaz no solo de interpretar los resultados sino de tomar la decisión correcta en base a ellos.

Con el objetivo de que la IC pueda ser utilizada por terceras personas para realizar los diagnósticos de nuevas imágenes, se ha diseñado una aplicación web y un servicio API. La aplicación web puede ser utilizada en un equipo local o en un servidor remoto y su despliegue es muy rápido y sencillo. La curva de aprendizaje es casi nula, con lo que puede

ser utilizada por casi cualquier persona. La aplicación no requiere la instalación de software adicional en los equipos clientes y muy escaso en el equipo servidor. Devuelve los diagnósticos en apenas 3 segundos con un porcentaje de acierto superior al 93%. La API se ha desarrollado para realizar diagnósticos de paquetes de imágenes mediante procesos automatizados. El tiempo de respuesta de la API en los diagnósticos para cada imagen es idéntico al ofrecido por la aplicación web en las mismas condiciones de red.

Es imperativo resaltar la importancia que tiene que la herramienta sea utilizada por un dermatólogo, debido a sus limitaciones, principalmente la del sesgo de los colores de piel amarilla y negra, y para aquellos diagnósticos sospechosos para los que la IC sea incapaz de clasificar la lesión cutánea con un porcentaje de acierto elevado. La herramienta puede ser, por tanto, un excelente asistente para el médico especialista, reduciendo encarecidamente los tiempos para obtener el diagnóstico final, con un ahorro de gastos clínicos de pruebas adicionales considerable y dando apoyo al médico en la toma de decisiones.

Es muy importante destacar que en la actualidad los conjuntos de imágenes de lesiones cutáneas disponibles públicamente no contienen volúmenes de imágenes suficientes de piel amarilla y negra, con lo que no es posible garantizar los mismos porcentajes de acierto detallados en este trabajo cuando se realizan diagnósticos de imágenes para estos colores de piel. Realizando búsquedas de conjuntos de datos con volúmenes suficientes de imágenes de lesiones cutáneas de piel negra, se encontró este artículo [62] en el que se mencionaba el problema de sesgo actual para entrenar IA por la falta de conjuntos de datos con imágenes de piel negra. A pesar de que la probabilidad de padecer melanoma de las personas de piel negra en comparación con las de piel blanca es muy inferior, 1 de cada 1.000 vs 1 de cada 38, la probabilidad existe y, por tanto, la problemática debe ser tratada en trabajos futuros.

En cuanto a las bases de datos de piel amarilla, realizando la búsqueda de conjuntos de datos de imágenes, se encontró este artículo [63], en el que se mencionaba una base de datos con 107.565 imágenes de asiáticos, pero no está disponible en la url indicada en el artículo.

Las redes neuronales necesitan datos de calidad y con la cantidad suficiente para ser entrenadas adecuadamente. Dado que en el momento de realizar este trabajo no había disponibles bases de datos de personas de piel amarilla y negra, quedaría pendiente este aspecto para trabajos futuros. Sería ideal entrenar las redes con los conjuntos de datos de cada color de piel obteniendo 3 ICs, cada una especializada en un color de piel. En la aplicación web habría que añadir un selector que permitiera seleccionar el color de piel de la lesión cutánea que se pretende analizar para que utilizara la IC adecuada en cada caso. Esta sería la limitación actual más importante de la herramienta, algo a tener en cuenta cuando los dermatólogos analizaran lesiones cutáneas de pacientes de piel amarilla o negra.

También ha quedado pendiente para trabajos futuros la implementación de una aplicación para dispositivos móviles. Con la aplicación actual es posible acceder desde el navegador web de un dispositivo móvil, siempre que el servidor esté accesible, pero dadas las características de estos dispositivos, su usabilidad y rendimiento mejoraría significativamente si se utilizara la IC desde una APP móvil. Esta sería la hoja de ruta de trabajos futuros para la mejora de la herramienta:

- Obtener bases de datos de calidad y con la cantidad suficiente de imágenes de lesiones cutáneas etiquetadas adecuadamente de piel amarilla y negra.

- Entrenar las 10 redes que forman la IC con las dos bases de datos con los dos conjuntos de datos independientemente, y obteniendo sus matrices de pesos. Con esto, se dispondría de 3 ICs, cada una especializada en los distintos colores de piel.

- Añadir un selector en la aplicación web que permita escoger el color de piel de la lesión cutánea a diagnosticar.

- Desarrollar la APP para móvil que aportaría movilidad a los dermatólogos, además de poder analizar imágenes capturadas con las cámaras de los *smartphones* en tiempo real.

- Desarrollar redes neuronales que añadan explicabilidad a la IC que permitan a los humanos, en este caso a los dermatólogos, entender porque el modelo tomó la decisión aportando una explicación razonable sobre la predicción, en lugar de aportar una caja negra.

5. Glosario

***Data augmentation* (aumento de datos).** En el análisis de datos son técnicas utilizadas para aumentar la cantidad de datos agregando copias ligeramente modificadas de datos ya existentes o datos sintéticos creados a partir de datos existentes.

Deep Learning (DL). Tipo de aprendizaje automático que entrena a una computadora para que realice tareas como las que hacemos los seres humanos, como el reconocimiento del habla, la identificación de imágenes o hacer predicciones.

Google Colab. Entorno colaborativo en la nube para trabajar aprendizaje automático en Python con Jupyter Notebooks.

Jupyter Notebooks. Aplicación web de código abierto que permite crear y compartir código y documentos.

Kaggle. Entorno colaborativo en la nube para trabajar aprendizaje automático en Python con Jupyter Notebooks.

Keras. Biblioteca de redes neuronales artificiales de código abierto desarrollada en Python.

Hiperparámetros. Parámetros ajustables que permiten controlar el proceso de entrenamiento de un modelo. En la redes neuronales los más utilizados son la tasa de aprendizaje (*learning rate*) y el tamaño del lote (*batch size*). El rendimiento de un modelo depende en gran medida de los hiperparámetros

ImageNet. Base de datos de imágenes organizada según una jerarquía de WordNet, en la que cada nodo de la jerarquía está representado por cientos y miles de imágenes. Los datos están disponibles de forma gratuita para los investigadores para fines no comerciales.

ISIC (*International Skin Imaging Collaboration*). asociación internacional entre la academia y la industria diseñada para desarrollar y promover estándares para imágenes digitales de la piel para ayudar a reducir la mortalidad por melanoma.

***Oversampling* (sobremuestreo)**. En estadística, es una técnica utilizada para ajustar la distribución de clases de un conjunto de datos, augmentando el números de las clases minoritarias con copias hasta igualar las cantidad de muestras de todas las clases.

PubMed. una base de datos, de acceso libre y especializada en ciencias de la salud, con más de 19 millones de referencias bibliográficas.

Python. Lenguaje de programación de alto nivel.

PyTorch. Biblioteca de aprendizaje automático de código abierto basada en la biblioteca de Torch, utilizado para aplicaciones que implementan cosas como visión artificial y procesamiento de lenguajes naturales.

Red Neuronal Convolucional (CNN). Tipo de red neuronal artificial donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria de un cerebro biológico.

Red Neuronal Transformer. Tipo de red neuronal artificial que aprende contexto y, por lo tanto, significado mediante el seguimiento de relaciones en datos secuenciales como las palabras o píxeles de imágenes.

Tensorflow. Biblioteca de código abierto para aprendizaje automático a través de un rango de tareas, y desarrollado por Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales.

6. Bibliografía

[1] The Skin Cancer Foundation, “Líder en la Lucha Contra el Cáncer de Piel”, 2022. https://cancerdepiel.org/quienes-somos/lider-en-la-lucha-contra-el-cancer-de-piel (accedido el 3 de octubre de 2022).

[2] American Cancer Society, “Detección temprana, diagnóstico y clasificación por etapas”, 2022. https://www.cancer.org/es/cancer/cancer-de-piel-tipo-melanoma/deteccion-diagnostico-clasificacion-por-etapas.html (accedido el 3 de octubre de 2022).

[3] Naeem A., Anees T., Fiza M., Naqvi R.A.; Lee S. SCDNet: A Deep Learning-Based Framework for the Multiclassification of Skin Cancer Using Dermoscopy Images. Sensors (Basel). 2022 Jul 28;22(15):5652. doi: 10.3390/s22155652. [PubMed]

[4] Hou Lee J.R., Pavlova M., Famouri M., Wong A. Cancer-Net SCa: tailored deep neural network designs for detection of skin cancer from dermoscopy images. BMC Med Imaging. 2022 Aug 9;22(1):143. doi: 10.1186/s12880-022-00871-w. [PubMed]

[5] Gouda W., Sama N.U., Al-Waakid G., Humayun M., Jhanjhi N.Z. Detection of Skin Cancer Based on Skin Lesion Images Using Deep Learning. Healthcare (Basel). 2022 Jun 24;10(7):1183. doi: 10.3390/healthcare10071183. [PubMed]

[6] Fraiwan M., Faouri E. On the Automatic Detection and Classification of Skin Cancer Using Deep Transfer Learning. Sensors (Basel), 2022 Jun 30;22(13):4963. doi: 10.3390/s22134963. [PubMed]

[7] Popescu D., El-Khatib M., Ichim L. Skin Lesion Classification Using Collective Intelligence of Multiple Neural Networks. Sensors (Basel). 2022 Jun 10;22(12):4399. doi: 10.3390/s22124399. [PubMed]

[8] Alfi I.A., Rahman M., Shorfuzzaman Md.M., Nazir A. A Non-Invasive Interpretable Diagnosis of Melanoma Skin Cancer Using Deep Learning and Ensemble Stacking of Machine Learning Models. Diagnostics (Basel). 2022 Mar 17;12(3):726. doi: 10.3390/diagnostics12030726. [PubMed]

[9] Bechelli S., Delhommelle J. Machine Learning and Deep Learning Algorithms for Skin Cancer Classification from Dermoscopic Images. Bioengineering (Basel). 2022 Feb 27;9(3):97. doi: 10.3390/bioengineering9030097. [PubMed]

[10] Arif M., Philip F.M., Ajesh F., Izdrui D., Craciun M.D., Geman O. Automated Detection of Nonmelanoma Skin Cancer Based on Deep Convolutional Neural Network. J Healthc Eng. 2022 Feb 10;2022:6952304. doi: 10.1155/2022/6952304. eCollection 2022. [PubMed]

[11] Mazoure B., Mazoure A., Bédard J., Makarenkov V. DUNEScan: a web server for uncertainty estimation in skin cancer detection with deep neural networks. Sci Rep. 2022 Jan 7; 12(1): 179. doi: 10.1038/s41598-021-03889-2. [PubMed]

[12] Hasan M.R., Fatemi M.I., Khan M.M., Kaur M., Zaguia A. Comparative Analysis of Skin Cancer (Benign vs. Malignant) Detection Using Convolutional Neural Networks. J Healthc Eng. 2021 Dec 11;2021:5895156. doi: 10.1155/2021/5895156. eCollection 2021. [PubMed]

[13] Jaworek-Korjakowska J., Brodzicki A., Cassidy B., Kendrick C., Yap M.H.. Interpretability of a Deep Learning Based Approach for the Classification of Skin Lesions into Main Anatomic Body Sites. Cancers (Basel). 2021 Dec 1;13(23):6048. doi: 10.3390/cancers13236048. [PubMed]

[14] Jain S., Singhania U., Tripathy B., Nasr E.A., Aboudaif M.K., Kamrani A.K. Deep Learning-Based Transfer Learning for Classification of Skin Cancer. Sensors (Basel). 2021 Dec 6;21(23):8142. doi: 10.3390/s21238142. [PubMed]

[15] Attallah O., Sharkas M. Intelligent Dermatologist Tool for Classifying Multiple Skin Cancer Subtypes by Incorporating Manifold Radiomics Features Categories. Contrast Media Mol Imaging. 2021 Sep 15;2021:7192016. doi: 10.1155/2021/7192016. [PubMed]

[16] Abbas Q., Ramzan F., Ghani M.U. Acral melanoma detection using dermoscopic images and convolutional neural networks. Vis Comput Ind Biomed Art. 2021 Oct 7;4(1):25. doi: 10.1186/s42492-021-00091-z. [PubMed]

[17] Pham T., Luong C., Hoang V., Doucet A. AI outperformed every dermatologist in dermoscopic melanoma diagnosis, using an optimized deep-CNN architecture with custom mini-batch logic and loss function. Sci Rep. 2021 Sep 1;11(1):17485. doi: 10.1038/s41598-021-96707-8. [PubMed]

[18] Maron R.C., Schlager J.G., Haggenmüller S., Kalle C., Utikal J.S., Meier F., Gellrich F.F., Hobelsberger S., Hauschild A., French L., Heinzerling L., Schlaak M., Ghoreschi K., Hilke F.J., Poch G., Heppt M.V., Berking C., Haferkamp S., Sondermann W., Schadendorf D., Schilling B., Goebeler M., Krieghoff-Henning E., Hekler A., Fröhling S., Lipka D.B., Kather J.N., Brinker T.J. A benchmark for neural network robustness in skin cancer classification. Eur J Cancer. 2021 Sep;155:191-199. doi: 10.1016/j.ejca.2021.06.047. Epub 2021 Aug 11. [PubMed]

[19] Kwiatkowska D., Kluska P., Reich A. Convolutional neural networks for the detection of malignant melanoma in dermoscopy images. Postepy Dermatol Alergol. 2021 Jun;38(3):412-420. doi: 10.5114/ada.2021.107927. Epub 2021 Jul 26. [PubMed]

[20] Chaahat, Gondhi N.K., Lehana P.K. An Evolutionary Approach for the Enhancement of Dermatological Images and Their Classification Using Deep Learning Models. J Healthc Eng. 2021 Jul 15;2021:8113403. doi: 10.1155/2021/8113403. eCollection 2021. [PubMed]

[21] Gouabou A.C.F., Damoiseaux J., Monnier J., Iguernaissi R., Moudafi A., Merad D. Ensemble Method of Convolutional Neural Networks with Directed Acyclic Graph Using Dermoscopic Images: Melanoma Detection Application. Sensors (Basel). 2021 Jun 10;21(12):3999. doi: 10.3390/s21123999. [PubMed]

[22] Alsaade F.W., Aldhyani T.H.H., Al-Adhaileh M.H. Developing a Recognition System for Diagnosing Melanoma Skin Lesions Using Artificial Intelligence Algorithms. Comput Math Methods Med. 2021 May 15;2021:9998379. doi: 10.1155/2021/9998379. eCollection 2021. [PubMed]

[23] Khan M.A., Sharif M., Akram T., Damaševičius R., Maskeliūnas R. Skin Lesion Segmentation and Multiclass Classification Using Deep Learning Features and Improved Moth Flame Optimization. Diagnostics (Basel). 2021 Apr 29;11(5):811. doi: 10.3390/diagnostics11050811. [PubMed]

[24] Maron R.C., Hekler A., Krieghoff-Henning E., Schmitt M., Schlager J.G., Utikal J.S., Brinker T.J. Reducing the Impact of Confounding Factors on Skin Cancer Classification via Image Segmentation: Technical Model Study. J Med Internet Res. 2021 Mar 25;23(3):e21695. doi: 10.2196/21695. [PubMed]

[25] Acosta M.F.J., Tovar L.Y.C., Garcia-Zapirain M.B., Percybrooks W.S. Melanoma diagnosis using deep learning techniques on dermatoscopic images. BMC Med Imaging. 2021 Jan 6;21(1):6. doi: 10.1186/s12880-020-00534-8. [PubMed]

[26] Rehman M.Z.U., Ahmed F., Alsuhibany S.A., Jamal S.S., Ali M.Z., Ahmad J. Classification of Skin Cancer Lesions Using Explainable Deep Learning. Sensors (Basel). 2022 Sep 13;22(18):6915. doi: 10.3390/s22186915. [PubMed]

[27] Girdhar N., Sinha A., Gupta S. DenseNet-II: an improved deep convolutional neural network for melanoma cancer detection. Soft comput. 2022 Aug 24;1-20. doi: 10.1007/s00500-022-07406-z. Online ahead of print. [PubMed]

[28] Alam T.M., Shaukat K., Khan W.A., Hameed I.A., Almuqren L.A., Raza M.A., Aslam M., Luo S. An Efficient Deep Learning-Based Skin Cancer Classifier for an Imbalanced Dataset. Diagnostics (Basel). 2022 Aug 31;12(9):2115. doi: 10.3390/diagnostics12092115. [PubMed]

[29] Chang C., Li Y., Wu H., Tseng M. Melanoma Detection Using XGB Classifier Combined with Feature Extraction and K-Means SMOTE Techniques. Diagnostics (Basel). 2022 Jul 19;12(7):1747. doi: 10.3390/diagnostics12071747. [PubMed]

[30] Bhimavarapu U., Battineni G. Skin Lesion Analysis for Melanoma Detection Using the Novel Deep Learning Model Fuzzy GC-SCNN. Healthcare (Basel). 2022 May 23;10(5):962. doi: 10.3390/healthcare10050962. [PubMed]

[31] Dascalu A., Walker B.N., Oron Y., David E.O. Non-melanoma skin cancer diagnosis: a comparison between dermoscopic and smartphone images by unified visual and sonification deep learning algorithms. J Cancer Res Clin Oncol. 2022 Sep;148(9):2497-2505. doi: 10.1007/s00432-021-03809-x. [PubMed]

[32] Serrano C., Lazo M., Serrano A., Toledo-Pastrana T., Barros-Tornay R., Acha B. Clinically Inspired Skin Lesion Classification through the Detection of Dermoscopic Criteria for Basal Cell Carcinoma. J Imaging. 2022 Jul 12;8(7):197. doi: 10.3390/jimaging8070197. [PubMed]

[33] Afza F., Sharif M., Khan M.A., Tariq U., Yong H., Cha J. Multiclass Skin Lesion Classification Using Hybrid Deep Features Selection and Extreme Learning Machine. Sensors (Basel). 2022 Jan 21;22(3):799. doi: 10.3390/s22030799. [PubMed]

[34] Nauta M., Walsh R., Dubowski A., Seifert C. Uncovering and Correcting Shortcut Learning in Machine Learning Models for Skin Cancer Diagnosis. Diagnostics (Basel). 2021 Dec 24;12(1):40. doi: 10.3390/diagnostics12010040. [PubMed]

[35] Lu X., Zadeh Y.A.F.A. Deep Learning-Based Classification for Melanoma Detection Using XceptionNet. J Healthc Eng. 2022 Mar 22;2022:2196096. doi: 10.1155/2022/2196096. eCollection 2022. [PubMed]

[36] Sreekalak.1, Rajkumar n., Sugumar R., Sagar K.V.D., Shobarani R., Krishnamoorthy K.P., Saini A.K., Palivela H., Yeshitla A. Skin Diseases Classification Using Hybrid AI Based Localization Approach. Comput Intell Neurosci. 2022 Aug 29;2022:6138490. doi: 10.1155/2022/6138490. eCollection 2022. [PubMed]

[37] Ghosh P., Azam S., Quadir R., Karim A., Shamrat F.M.J.M., Bhowmik S.K., Jonkman M., Hasib K.Md., Ahmed K. SkinNet-16: A deep learning approach to identify benign and malignant skin lesions. Front Oncol. 2022 Aug 8;12:931141. doi: 10.3389/fonc.2022.931141. eCollection 2022. [PubMed]

[38] Kaur R., GholamHosseini H., Sinha R., Maria Lindén M. Automatic lesion segmentation using atrous convolutional deep neural networks in dermoscopic skin cancer images. BMC Med Imaging. 2022 May 29;22(1):103. doi: 10.1186/s12880-022-00829-y. [PubMed]

[39] Martin-Gonzalez M., Azcarraga C., Martin-Gil A., Torres C., Jaen P. Efficacy of a Deep Learning Convolutional Neural Network System for Melanoma Diagnosis in a Hospital Population. Int J Environ Res Public Health. 2022 Mar 24;19(7):3892. doi: 10.3390/ijerph19073892. [PubMed]

[40] Kaur R., GholamHosseini H., Sinha R., Lindén M. Melanoma Classification Using a Novel Deep Convolutional Neural Network with Dermoscopic Images. Sensors (Basel). 2022 Feb 2;22(3):1134. doi: 10.3390/s22031134. [PubMed]

[41] Ningrum D.N.A., Yuan S., Kung W., Wu C., Tzeng I:, Huang C., Yu-Chuan Li J., Wang Y. Deep Learning Classifier with Patient's Metadata of Dermoscopic Images in Malignant Melanoma Detection. J Multidiscip Healthc. 2021 Apr 21;14:877-885. doi: 10.2147/JMDH.S306284. eCollection 2021. [PubMed]

[42] Aladhadh S., Alsanea M., Aloraini M., Khan T., Habib S., Islam M. An Effective Skin Cancer Classification Mechanism via Medical Vision Transformer. Sensors (Basel). 2022 May 25;22(11):4008. doi: 10.3390/s22114008. [PubMed]

[43] The ISIC Challenge, “ISIC Challenge Datasets”, 2022. https://challenge.isic-archive.com/data/ (accedido el 14 de octubre de 2022).

[44] Tschandl P., Rosendahl C., Kittler H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci. Data. 2018;5:180161. doi: 10.1038/sdata.2018.161. [PubMed]

[45] Haggenmüller S., Maron R.C., Hekler A., Utikal J.S., Barata C., Barnhill R.L., Beltraminelli H., Berking C., Betz-Stablein B, Blum A., Braun S.A., Carr R., Combalia M., Fernandez-Figueras M.T., Ferrara G., Fraitag S., French L.E., Gellrich F.F., Ghoreschi K., Goebeler M., Guitera P., Haenssle H.A., Haferkamp S., Heinzerling L., Heppt M.V., Hilke F.J., Hobelsberger S., Krahl D., Kutzner H., Lallas A., Liopyris K., Llamas-Velasco M., Malvehy J., Meier F., Müller C.S.L, Navarini A.A., Navarrete-Dechent C., Perasole A., Poch G., Podlipnik S., Requena L., Rotemberg V.M., Saggini A., Sangueza O.P., Santonja C., Schadendorf D., Schilling B., Schlaak M., Schlager J.G., Sergon M., Sondermann W., Soyer H.P., Starz H., Stolz W., Vale E., Weyers W., Zink A., Krieghoff-Henning E., Kather J.N., Kalle C., Lipka D.B., Fröhling S., Hauschild A., Kittler H., Brinker T.J.. Skin cancer classification via convolutional neural networks: systematic review of studies involving human experts. Eur J Cancer. 2021 Oct;156:202-216. doi: 10.1016/j.ejca.2021.06.049. Epub 2021 Sep 8. [PubMed]

[46] Kaggle, “Skin Cancer MNIST: HAM10000”, 2022. https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000 (accedido el 14 de octubre de 2022).

[47] Dosovitskiy A., Beyer L., Kolesnikov A., Weissenborn D., Zhai X., Unterthiner T., Dehghani M., Minderer M., Heigold G., Gelly S., Uszkoreit J., Houlsby N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. doi: arXiv:2010.11929. arXiv 2021 Jun 3. [arXiv]

[48] Liu Z., Lin Y., Cao Y., Hu H., Wei Y., Zhang Z., Lin S., Guo B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. doi: arXiv:2103.14030. arXiv 2021 Mar 25. [arXiv]

[49] Tan M., Le Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. doi: arXiv:1905.11946. arXiv 2019 May 28. [arXiv]

[50] Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions. doi: arXiv:1610.02357. arXiv 2016 Oct 7. [arXiv]

[51] He K., Zhang X., Ren S., Sun J.. Deep Residual Learning for Image Recognition. doi: arXiv:1512.03385. arXiv 2015 Dec 10. [arXiv]

[52] Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. doi: arXiv:1409.1556. arXiv 2014 Sep 4. [arXiv]

[53] Huang G., Liu Z., van der Maaten L., Weinberger K.Q. Densely Connected Convolutional Networks. doi: arXiv:1608.06993. arXiv 2016 Aug 25. [arXiv]

[54] Howard A., Sandler M., Chu G., Chen L., Chen B., Tan M., Wang W., Zhu Y., Pang R., Vasudevan V., Le Q.V., Hartwig H. Searching for MobileNetV3. doi: arXiv:1905.02244. arXiv 2019 May 6. [arXiv]

[55] Google Research, “EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling”, 2022. https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html (accedido el 21 de octubre de 2022)

[56] DataScientest, “VGG: ¿Qué es este modelo? ¡Daniel te lo cuenta todo!”, 2022. https://datascientest.com/es/vgg-que-es-este-modelo-daniel-te-lo-cuenta-todo (accedido el 21 de octubre de 2022)

[57] Telefonica Tech, “Cómo interpretar la matriz de confusión: ejemplo práctico”, 2021. https://empresas.blogthinkbig.com/como-interpretar-la-matriz-de-confusion-ejemplo-practico/ (accedido el 16 de octubre de 2022)

[58] Towards Data Science, “A Quick Guide to AUC-ROC in Machine Learning Models”, 2021. https://towardsdatascience.com/a-quick-guide-to-auc-roc-in-machine-learning-models-f0aedb78fbad (accedido el 16 de octubre de 2022)

[59] Redalyc, “Transfer learning en la clasificación binaria de imágenes térmicas”, 2021. https://www.redalyc.org/journal/5055/505567902007/html/ (accedido el 18 de octubre de 2022)

[60] Flask, “Flask, web development, one drop at a time”, 2022. https://flask.palletsprojects.com/en/2.2.x/ (accedido el 21 de octubre de 2022)

[61] Hoya012’s research blog, “Deep Learning Image Classification Guidebook [2] PreActResNet, Inception-v2, Inception-v3, Inception-v4, Inception-ResNet, Stochastic Depth ResNet, WRN”, 2022. https://hoya012.github.io/blog/deeplearning-classification-guidebook-2/ (accedido el 24 de noviembre de 2022)

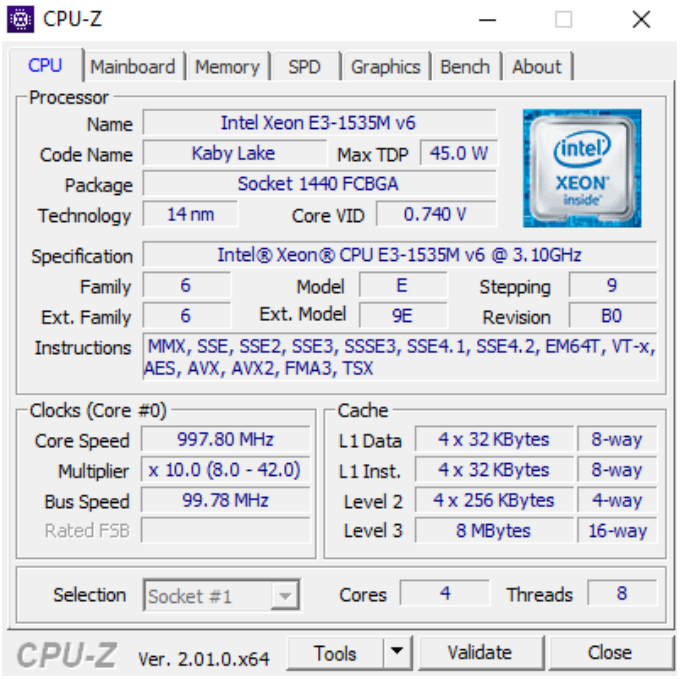
[62] NCRI, National Cancer Research Institute, “Data available for training AI to spot skin cancer are insufficient and lacking in pictures of darker skin”, 2022. https://www.ncri.org.uk/ai-to-spot-skin-cancer-lacking-pictures-of-darker-skin/ (accedido el 20 de diciembre de 2022)

[63] Xie B., He X., Zhao S., Li Y., Su J., Zhao X., Kuang Y., Wang Y. and Chen X. XiangyaDerm: A Clinical Image Dataset of Asian Race for Skin Disease Aided Diagnosis. Intravascular Imaging and Computer Assisted Stenting and Large-Scale Annotation of Biomedical Data and Expert Label Synthesis,2019. AiRI Artificial Intelligence and Robotics Lab. [AIRL]

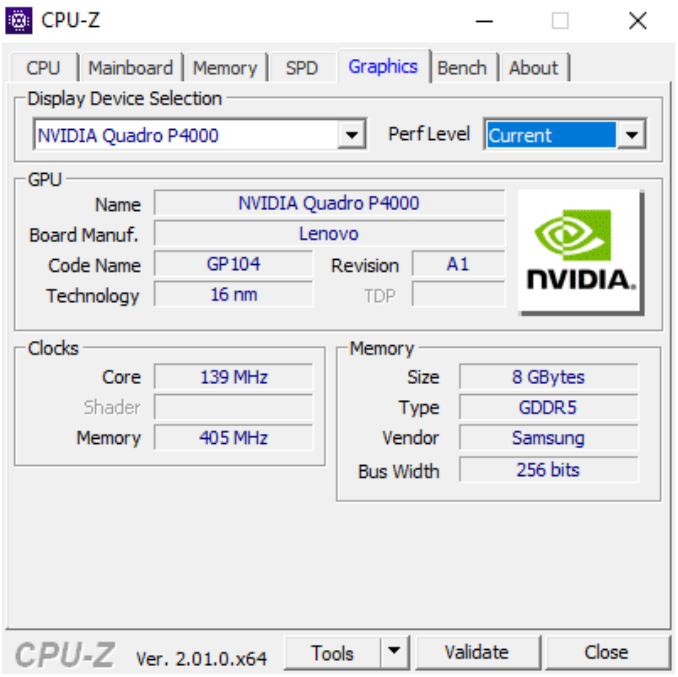
7. Anexos

Anexo 1. Características del equipo local

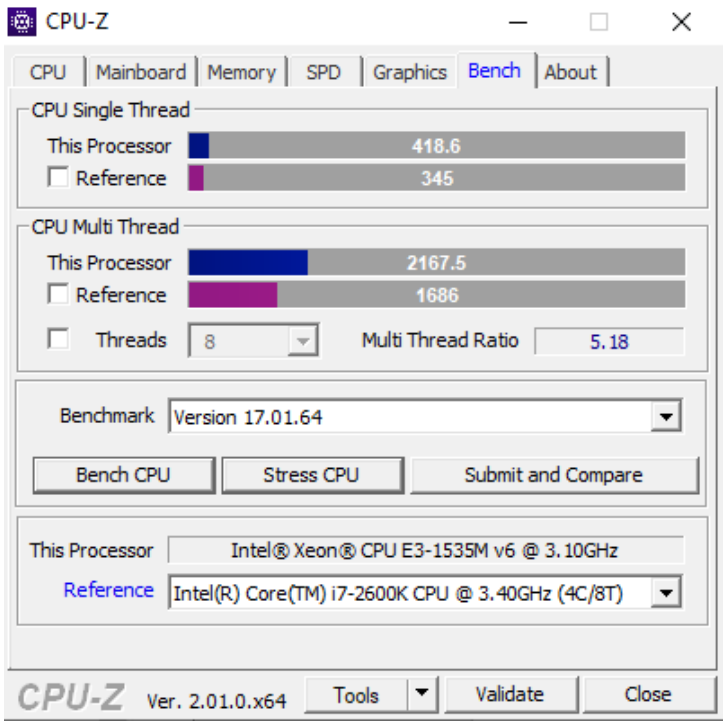
CPU



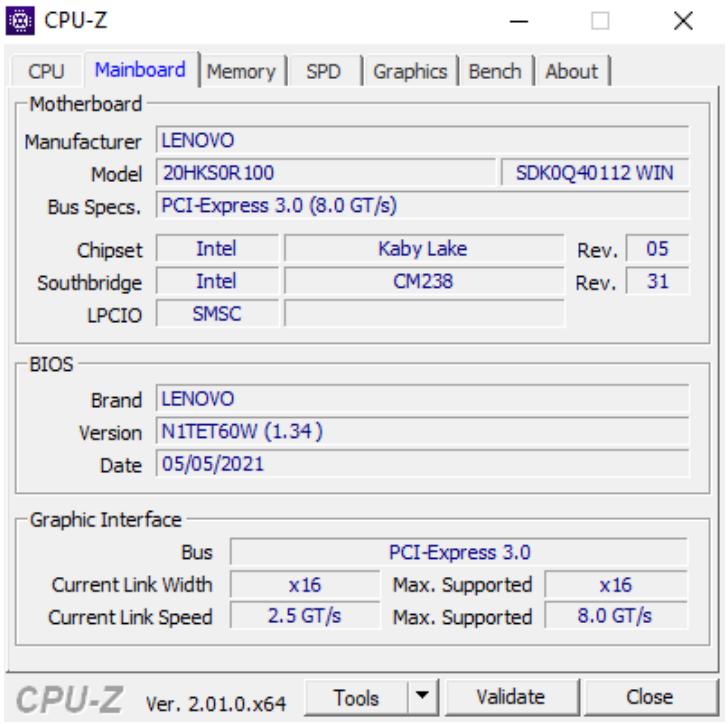
GPU



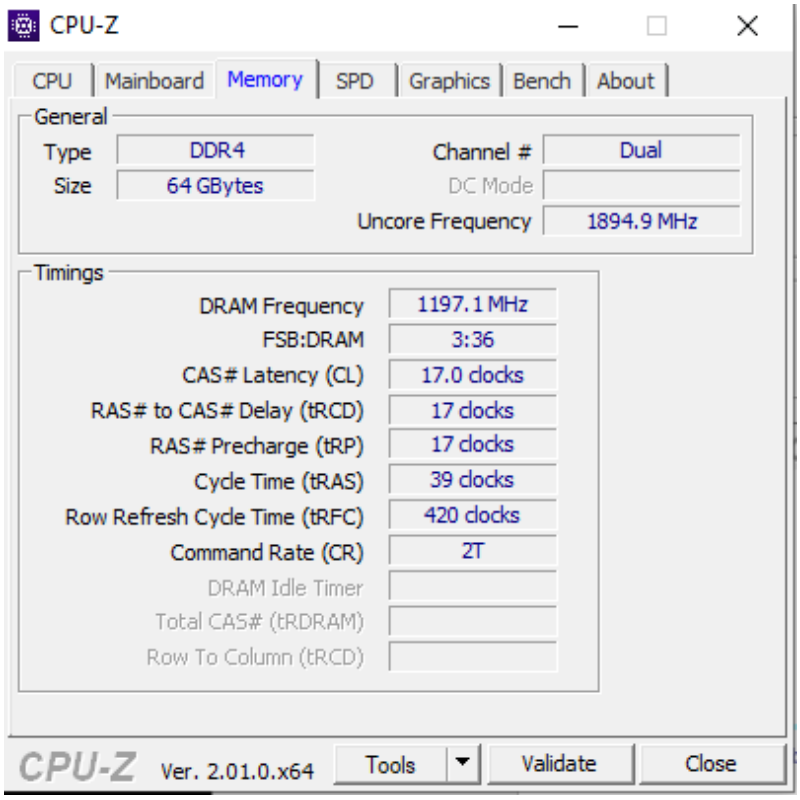
Benchmark CPU vs Intel i7-2600K 3.40 GHz

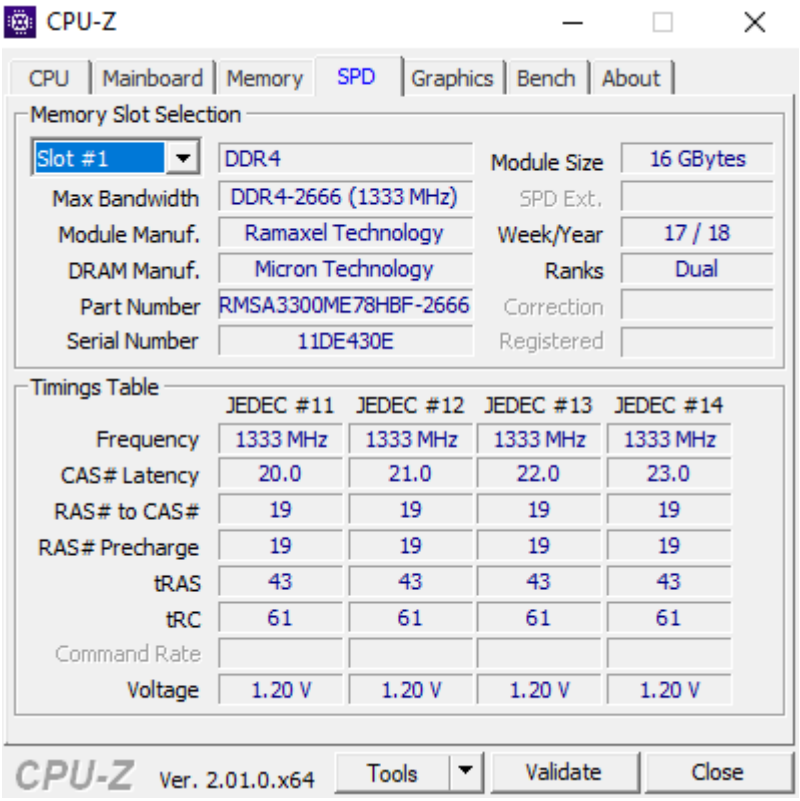


Placa base

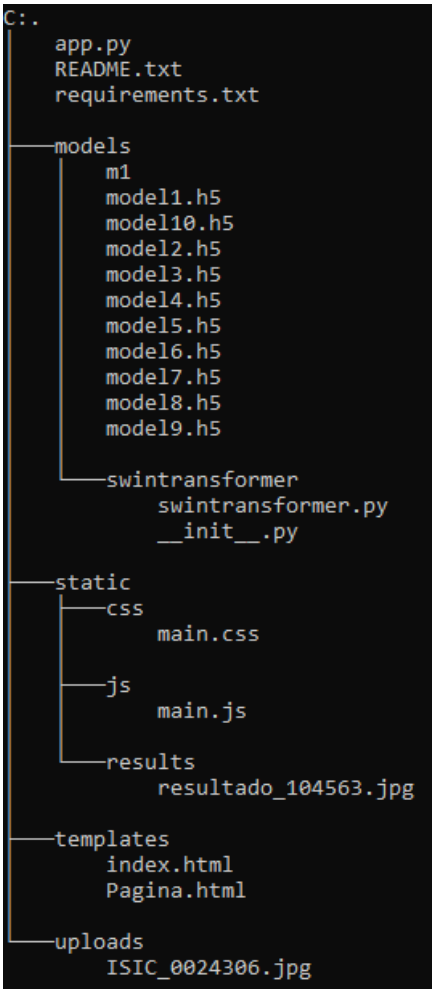


RAM


CPU-Z
CPU
Mainboard
Memory
SPD
Graphics
Bench
About
General
Type
DDR4
Size
64 GBytes
Channel #
Dual
DC Mode
Uncore Frequency
1894.9 MHz
Timings
DRAM Frequency
1197.1 MHz
FSB:DRAM
3:36
CAS# Latency (CL)
17.0 clocks
RAS# to CAS# Delay (tRCD)
17 clocks
RAS# Precharge (tRP)
17 clocks
Cycle Time (tRAS)
39 clocks
Row Refresh Cycle Time (tRFC)
420 clocks
Command Rate (CR)
2T
DRAM Idle Timer
Total CAS# (tRDRAM)
Row To Column (tRCD)
CPU-Z
Ver. 2.01.0.x64
Tools
Validate
Close


CPU-Z
CPU
Mainboard
Memory
SPD
Graphics
Bench
About
Memory Slot Selection
Slot #1
DDR4
Module Size
16 GBytes
Max Bandwidth
DDR4-2666 (1333 MHz)
SPD Ext.
Module Manuf.
Ramaxel Technology
Week/Year
17 / 18
DRAM Manuf.
Micron Technology
Ranks
Dual
Part Number
RMSA3300ME78HBF-2666
Correction
Serial Number
11DE430E
Registered
Timings Table
JEDEC #11
JEDEC #12
JEDEC #13
JEDEC #14
Frequency
1333 MHz
1333 MHz
1333 MHz
1333 MHz
CAS# Latency
20.0
21.0
22.0
23.0
RAS# to CAS#
19
19
19
19
RAS# Precharge
19
19
19
19
tRAS
43
43
43
43
tRC
61
61
61
61
Command Rate
Voltage
1.20 V
1.20 V
1.20 V
1.20 V
CPU-Z
Ver. 2.01.0.x64
Tools
Validate
Close

Anexo 2. Árbol de directorios y archivos de la aplicación



- **app.py**. Fichero con la implementación de la aplicación en código Python.
- **README**. Fichero con las instrucciones para instalar el software, configurar el entorno y utilizar la aplicación desde la web o a través de la API.
- **requirements.txt**. Fichero con las librerías y versiones requeridas para poder ejecutar la aplicación.
- **models**. Directorio que contiene el archivo con la matriz de pesos y los archivos con los pesos de los modelos ajustados.
    - **m1**. Archivo con la matriz de pesos de todas las redes.
    - **model{1-10}**. Ficheros con los pesos de cada modelo entrenado.

- **swintransformer**. Directorio con los ficheros de la implementación de la red Swin Transformer.
  - **swintransformer.py**. Archivo con la implementación en código Pyhton de la red Swin Transformer de rishigami (https://github.com/rishigami/Swin-Transformer-TF).
  - **__init__.py**. Archivo con el código Python de la importación de la clase swintransformer.

- **static**. Directorio de archivos estáticos de la aplicación web.
  - **css**. Directorio con los ficheros de estilo en cascada.
    - **main.css**. Fichero con el código de estilo en cascada (*Cascading Style Sheets*) de la aplicación web.
  - **js**. Directorio con los ficheros de Javascript de la aplicación web y de la API.
    - **main.js**. Archivo con el código Javascript de la aplicación web y de la API.
  - **results**. Directorio para guardar las gráficas con los resultados de las clasificaciones en formato JPG.
    - **resultado_{*id*}.jpg**. Fichero con la gráfica del resultado de una clasificación. Cada archivo tiene un identificador (id) único.

- **templates**. Directorio con las plantillas de la aplicación web.
  - **index.html**. Plantilla con el índice de la aplicación web.
  - **Pagina.html**. Plantilla de la página de la aplicación web.

- **uploads**. Directorio que contiene las imágenes clasificadas con la aplicación web.
  - **{*nombre_imagen*}.jpg**. Imagen subida para clasificar. Mantiene el nombre original.