UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Máster en Análisis y Visualización de Datos Masivos

Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Transfer Learning en la detección de cáncer de piel.

Trabajo Fin de Máster

Presentado por: Martínez González, Carlos

Director/a: Marti, Julio Marcelo

Resumen

El crecimiento de las temperaturas provocadas por el cambio climático y el uso recreativo de la exposición al sol han desembocado en una tendencia creciente de la incidencia del cáncer de piel en los últimos años. La rápida detección de este tipo de infección es crucial para la salud del paciente. Este proceso requiere de gran experiencia y control sobre los parámetros que indican infección.

Gracias a la continua evolución de los sistemas de información y el poder que aporta una buena gestión de los datos en la actualidad, muchos hospitales y organizaciones sanitarias han comenzado a desarrollar herramientas que faciliten la labor del médico en este tipo de tareas.

Analizando este contexto, se decide estudiar la utilidad de la inteligencia artificial en el diagnóstico de los distintos tipos de cáncer de piel a partir de imágenes, favoreciendo la rápida detección y actuación de un especialista. En concreto, este proyecto presenta una investigación sobre el aprendizaje por transferencia (*Transfer Learning*) a partir de una comparativa de resultados.

Para ello, se ha generado un conjunto de datos específico a partir de datos procedentes de diversas fuentes de acceso libre (*ISIC*, *PH2* y *EDRA*) y un correcto preprocesamiento. El desarrollo se ha realizado utilizando *Python* y una serie de librerías para el manejo de datos (*Pandas*, *Numpy*, etc.) y otras para la visión artificial (*TensorFlow*, *Keras*, *Skicit-Learn*, etc...).

Palabras Clave: cáncer de piel, melanoma, lunares, transfer learning, redes neuronales, computer vision, big data.

Abstract

Rising temperatures caused by climate change and the recreational use of sun exposure have led to an increasing trend in the incidence of skin cancer in recent years. Early detection of this type of infection is crucial for the patient's health. This process requires a great deal of experience and control over the parameters that indicate infection.

Thanks to the continuous evolution of information systems and the power of good data management today, many hospitals and healthcare organizations have begun to develop tools that facilitate the doctor's work in this type of task.

Analyzing this context, it was decided to study the usefulness of artificial intelligence in the diagnosis of different types of skin cancer from images, favoring the rapid detection and action of a specialist. Specifically, this project presents an investigation on Transfer Learning based on a comparative of results.

For this purpose, a specific dataset has been generated from data from various open access sources (*ISIC*, *PH2* and *EDRA*) and a correct pre-processing. The development has been carried out using *Python* and a series of libraries for data management (*Pandas*, *Numpy*, etc.) and others for artificial vision (*TensorFlow*, *Keras*, *Skicit-Learn*, etc...).

Keywords: skin cancer, melanoma, moles, transfer learning, neural networks, computer vision, big data.

Índice de contenidos

1. Introducción 8
1.1 Motivación 8
1.2. Planteamiento del trabajo 9
1.3. Estructura del proyecto 10
2. Contexto y estado del arte 12
2.1. La piel y su estudio 12
2.1.1. Cáncer de piel 12
2.1.2. Detección y diagnóstico del cáncer de piel 13
2.2. Computer visión 14
2.2.1. Modelo computacional de una imagen 15
2.2.2. Pre-procesado 16
2.2.3. Clasificación de imágenes 17
2.3. Inteligencia Artificial en sanidad 23
2.3.1. Proyectos existentes 25
3. Objetivos concretos y metodología de trabajo 30
3.1. Objetivo general 30
3.2. Objetivos específicos 30
3.3. Metodología del trabajo 31
4. Desarrollo del modelo 33
4.1. Descripción 33
4.2. Obtención y comprensión de los datos 33
4.3. Preparación de los datos 35
4.3.1. Homogeneización del conjunto 35
4.3.2. Pre-procesado 36
4.4. Modelado 42
4.4.1. Elección de los modelos pre-entrenados 42

4.4.2. Desarrollo del modelo general 44
4.5. Evaluación 61
4.5.1. Validación clínica 61
4.5.2. Evaluación del modelo 62
5. Conclusiones y trabajo futuro 71
5.1. Conclusiones 71
5.2. Líneas de trabajo futuro 72
6. Bibliografía 74
Anexos 82
Anexo I. Python 82
Librerías para CV 83
Anexo II. Código del proyecto 86
Homogeneización de los datos 86
Pre-procesado 86
Escoger modelo pre-entrenado 86
Desarrollo de los modelos 86

Índice de tablas

Tabla 1: Estructura del archivo metadata. 35

Tabla 2: Distribución de imágenes por clase. 36

Tabla 3: Distribución final del dataset. 36

Tabla 4: Precisión del modelo y número de parámetros. 43

Tabla 5: Adquisición de los datos. 46

Tabla 6: Arquitectura del modelo con MobileNet. 49

Tabla 7: Etapas de entrenamiento (MobileNet). 52

Tabla 8: Etapas de entrenamiento (Xception). 53

Tabla 9: Etapas de entrenamiento (ResNet). 55

Tabla 10: Arquitectura del modelo con MobileNet tras el Fine Tunning. 55

Tabla 11: Etapas de entrenamiento tras el Fine Tunning (MobileNet). 57

Tabla 12: Etapas de entrenamiento tras el Fine Tunning (Xception). 58

Tabla 13: Etapas de entrenamiento tras el Fine Tunning Pt 1 (ResNet). 59

Tabla 14 :Etapas de entrenamiento tras el Fine Tunning Pt 2 (ResNet). 60

Tabla 14: Métricas de clasificación (MobileNet). 67

Tabla 15: Métricas de clasificación (Xception). 67

Tabla 16: Métricas de clasificación (ResNet). 68

Tabla 17: Métricas de clasificación (CNN). 69

Índice de figuras

Figura 1: Diagrama de píxeles de una imagen (Levin, 2020). 15
Figura 2: Array de píxeles (Mihajlovic, 2019). 16
Figura 3: Hiperplanos posibles (Gandhi, 2018). 18
Figura 4: Ejemplo KNN (Rath, 2019). 19
Figura 5: Comparativa entre neurona biológica y artificial (Khan, 2019). 20
Figura 6: Arquitectura ANN (Lelli, 2019). 20
Figura 7: Arquitectura CNN (Saha, 2018). 21
Figura 8: Convolución y función de activación sobre una sección (Na8, 2018). 21
Figura 9: Ejemplos de transferencia de conocimiento (Zhuang, 2020). 22
Figura 10: Evaluación de SkinVision (SkinVision, 2021). 25
Figura 11: Google AI-powered dernatology tool (Bui, 2020). 26
Figura 12: Arquitectura de Google AI-powered dermatology tool (Liu, 2020). 27
Figura 13: Fases de la metodología CRISP-DM (Moreira, 2018). 31
Figura 14: Imagen re-escalada. 37
Figura 15: Imagen filtrada por la mediana. 38
Figura 16: Imagen en escala de grises. 38
Figura 17: Imagen con corrección gamma. 39
Figura 18: Imagen binarizada. 40
Figura 19: Imagen con los bordes binarizados. 41
Figura 20: Segmentación de la imagen. (Izq: Superposición de color. Dcha: Adición de bordes.) 41
Figura 21: Dispersión de la precisión según el tamaño del modelo. 44
Figura 22: Fases del modelo. 47
Figura 23: Fase de extracción de características. 48
Figura 24: Fase de clasificación. 48
Figura 25: Evolución de la pérdida (MobileNet). 51
Figura 26: Evolución de la precisión (MobileNet). 52

Figura 27: Evolución de la pérdida (Xception)..... 53
Figura 28: Evolución de la precisión (Xception). 53
Figura 29: Evolución de la pérdida (ResNet)..... 54
Figura 30: Evolución de la precisión (ResNet). 54
Figura 31: Evolución de la pérdida tras el Fine Tunning (MobileNet)..... 56
Figura 32: Evolución de la precisión tras el Fine Tunning (MobileNet). 56
Figura 33: Evolución de la pérdida tras el Fine Tunning (Xception)..... 57
Figura 34: Evolución de la precisión tras el Fine Tunning (Xception). 58
Figura 35: Evolución de la pérdida tras el Fine Tunning (ResNet)..... 59
Figura 36: Evolución de la precisión tras el Fine Tunning (ResNet). 59
Figura 37: Matriz de confusión (MobileNet)..... 64
Figura 38: Matriz de confusión (Xception)..... 65
Figura 39: Matriz de confusión (ResNet)..... 65

1. Introducción

1.1 Motivación

El cáncer de piel es uno de los tipos de tumor que más casos supone al año, en algunos países, los diagnósticos de cáncer de piel superan al número de casos del resto de cánceres combinados (Bray et al. 2021). Existen diferentes formas de cáncer en la piel, siendo el melanoma el que más riesgo supone para el enfermo y el que más preocupa en la actualidad, debido a su tendencia creciente en el número de casos durante la última década (American Cancer Society, 2021).

En la mayoría de los casos, el cáncer de piel viene provocado por la sobreexposición a la radiación UV. Actividades cada vez más comunes como la exposición recreativa a la luz solar o las fuentes artificiales de bronceado son factores de riesgo importantes. Pero también la disminución de los niveles de ozono debido al calentamiento global es alarmante, ya que supone la perdida de un filtro natural ante este tipo de radiación. Es decir, ni los hábitos del ser humano, ni la situación actual invitan a ser optimistas sobre esta enfermedad.

Los lunares, marcas, manchas, cambios de aspecto o de sensación en la piel son claros indicadores de melanomas. Los lunares, manchas o cambios de pigmentación en la piel suelen aparecer en la infancia o juventud y, por lo general, mantendrán su tamaño. Aunque suelen ser meros cambios en el color de la piel inofensivos y benignos, los dermatólogos recomiendan la revisión de estos y de su evolución.

La extirpación temprana del cáncer de tipo melanoma es trascendental en la supervivencia del paciente. De un 92% de probabilidad de vivir en 5 años si se localiza en etapas tempranas, a menos de un 66% si este se encuentra en una etapa más desarrollada. Actualmente, este examen ha de ser realizado por un experto al que no siempre se tiene acceso o se llega tarde (American Cancer Society, 2021, p.24-26).

Este análisis o examen está basado en aspectos morfológicos y de evolución en el lunar, la bien conocida regla ABCDE (Asimetría, borde, color, diámetro y evolución). En la mayoría de las ocasiones se necesita la precisión que proporciona el dermatoscopio.

Compartir y obtener datos es tarea fácil en la actualidad. La digitalización de la información supone una fuente de valores y herramientas que ayudan al análisis y solución de situaciones de alta complejidad. La inteligencia artificial, cuyo fundamento

está en el aprendizaje a partir de datos, ha alcanzado un progreso inimaginable años atrás (De Alba, 2019).

Uno de los sectores que más se ha beneficiado del avance de los sistemas de aprendizaje automático ha sido la medicina y, más concretamente, la dermatología. Durante los últimos años han aparecido una gran cantidad de soluciones que ayudan a la detección de enfermedades.

Herramientas que, utilizando datos sobre lunares o manchas en la piel, permiten a un médico no especializado obtener cierto control sobre la peligrosidad de futuros registros. De modo se puede decidir con seguridad y gestionar un examen de especialista de urgencia o se descarte la peligrosidad de la mancha en cuestión.

Ya que los factores que se analizan para el diagnóstico de este tipo de enfermedades son observables, estos sistemas se basan en el entrenamiento y evaluación de modelos de *Computer Vision* (Khan, 2020), a partir de imágenes. Siendo esta la rama del *Machine Learning* que aprende de ver y examinar imágenes.

El *Transfer Learning* es una de las técnicas que más inquietud han planteado en los últimos años en la clasificación de imágenes. Ya que se aprovecha el aprendizaje de modelos entrenados con bases de datos enormes, sin relación directa con el problema a solucionar, para que los sistemas específicos 'aprendan a aprender' (Zhuang et al, 2020).

1.2. Planteamiento del trabajo

Es en este contexto que se plantea realizar un análisis de la utilidad y fiabilidad del *Transfer Learning* en la detección de los distintos tipos de cáncer de piel. Para ello, se ha desarrollado un modelo base que se alimentará de diferentes modelos pre-entrenados y de un *dataset* de imágenes de lunares y manchas que ha sido previamente clasificado. Tratando de comprobar si verdaderamente esta técnica se puede plantear como una herramienta sanitaria que pueda ayudar a médicos de manera segura.

El modelo se ha desarrollado con el lenguaje de programación *Python* (Beklemysheva, 2020), sobre el que se han utilizado diversas librerías para el manejo de los datos y el desarrollo de la inteligencia artificial, que se recogen en el Anexo I. Python.

Al tratar imágenes personales con información sobre el estado de salud de pacientes, pueden considerarse datos delicados y con un alto riesgo de identificación, es decir, podrían no respetar las leyes de privacidad y protección de datos de carácter personal.

Para solucionar este posible problema habrá que buscar datos anónimos o anonimizar datos obtenidos, de modo que no se exponga información personal de los sujetos.

Previo al desarrollo del modelo es esencial la limpieza del conjunto de datos obtenidos. Este ha de estar formado por imágenes que eviten el sesgo del modelo. Será importante eliminar aquellas que sean idénticas o muy parecidas o las que no aporten información de valor como pueden ser lunares de niños, ya que siempre serán clasificados como benignos, alimentando el modelo con datos demasiado optimistas.

Otro paso que realizar sobre el *dataset* antes de alimentar el modelo es el pre-procesado de datos. Hay imágenes que se toman con diferente calidad, luz o color, evitando homogeneidad en los aspectos a comprobar. Por ello, redimensionar imágenes, eliminar ruido, segmentación y corrección de color son acciones que ayudan al modelo a predecir de manera más realista.

Con el conjunto de datos preparado, se desarrolla y alimenta un modelo que permita especificar la probabilidad de que un lunar padezca alguno de los tipos de cáncer, que se detallan más adelante, a partir de una imagen macroscópica de este. En definitiva, un sistema de clasificación de imágenes. Se ha evaluado la precisión de la herramienta y comparado para diferentes modelos pre-entrenados con el fin de analizar la utilidad final que esta pueda aportar sobre el campo.

1.3. Estructura del proyecto

En el presente documento se presentan los diferentes pasos seguidos durante el desarrollo de la herramienta planteada ordenados y compactados en capítulos.

En esta introducción se plantea el proyecto y los conceptos sobre los que se basará. A continuación, se ha investigado sobre los diferentes temas que conciernen al proyecto, de modo que se puedan aprovechar aportaciones anteriores y desarrollar una mejora sobre ellas.

Tras este análisis se describe el desarrollo del modelo general y su implementación, detallando los diferentes procesos que este conlleva. Para comprobar la utilidad de la herramienta será precisa la evaluación y análisis de los resultados obtenidos para cada uno de los modelos pre-entrenados.

Por último, se presenta un capítulo en el que se desarrollan las conclusiones obtenidas durante el desarrollo del proyecto y se planteará esta herramienta y el *Transfer Learning* en vista al futuro.

El contenido de este proyecto está basado en la investigación, por tanto, estará fundamentado en un conjunto de referencias a artículos, webs y libros relacionados con la materia a tratar. Estas referencias se especifican en el último capítulo.

2. Contexto y estado del arte

En este capítulo se describen detalladamente los diferentes sectores de estudio de los que se nutre el proyecto. Con la información explicada y contextualizada, se podrán obtener los objetivos y las aportaciones que este pueda suponer en la situación actual.

2.1. La piel y su estudio

La piel es el órgano más extenso del cuerpo, la primera línea de defensa del resto de órganos y partes del cuerpo ante agentes externos, reguladora de temperatura y una alarma ante el estado de salud del interior. El cuidado de esta es imprescindible para evitar que esos factores externos se conviertan en dañinos para la salud (Dermaten, 2020).

La dermatología es la especialidad médica que se encarga del estudio y conocimiento de la piel humana, el cabello, las uñas y las enfermedades que les afectan. Además del tratamiento y diagnóstico de enfermedades, una de las funciones más importantes que presenta esta rama es la prevención de dichas afecciones.

El dermatólogo, por tanto, es el médico especializado en dermatología. Pero la complejidad de un órgano tan extenso y expuesto como es la piel conlleva una cantidad inmensa de posibles enfermedades o daños. El dominio de esta rama es considerado uno de los más complejos de la medicina, y suele ir acompañado de una especialización interna (AAD, 2021).

Por ello, cuando aparece una enfermedad relativa a la piel, se suele recomendar acudir a un dermatólogo con experiencia y especializado. En muchas enfermedades, como en el cáncer de piel, el tiempo de reacción es fundamental para salvar la vida del paciente.

2.1.1. Cáncer de piel

Las células humanas crecen y se dividen formando células nuevas según el cuerpo necesita. Cuando las células envejecen, mueren y son sustituidas por otras. El cáncer consiste en la aparición de células anormales que se dividen de manera incontrolada (Mayo Clinic, 2019). Las células que deberían morir sobreviven y crean nuevas células innecesarias, invadiendo y dañando tejidos corporales. Este tipo de expansión invasiva se conoce como metástasis (National Cancer Institute, 2021).

Se trata de una enfermedad genética que puede aparecer en cualquier parte del cuerpo humano y puede ser heredada o aparecer por errores en la división celular ante exposiciones a ciertos elementos externos como el tabaco o la radiación del sol.

La piel se compone de dos capas principales: la epidermis y la dermis. La primera está formada por las células escamosas, debajo de estas se encuentran las células basales y los melanocitos, encargados del color. La dermis es la capa más interna y está formada por vasos sanguíneos y linfáticos, folículos pilosos y glándulas (MedicineNet, 2002).

En el caso del cáncer de piel, estas células incontrolables se generan en la epidermis. Suele estar provocado por la exposición de la piel a la radiación ultravioleta del sol o de las camas de bronceado. Hábitos que han provocado un crecimiento importante en la incidencia de este tipo de neoplastia, convirtiéndola en uno de los canceres más comunes en gran cantidad de países. Existen los siguientes tipos de cáncer de piel (Castañeda, 2016):

- **Carcinoma basocelular (CBC):** Es la variante más frecuente de este tipo de enfermedad. Se origina en las células basales y apéndices de la epidermis. Caracterizado por un crecimiento lento, ser localmente invasivo y no conllevar metástasis, es decir, no suele ser mortal.

- **Carcinoma espinocelular o epidermoide:** También conocido como cáncer de células escamosas (**CCE**), debido a que es en esta capa de la epidermis donde se genera. Más concretamente, se trata de la trasformación maligna de los queratinocitos. Son prácticamente inofensivos y curables cuando se detectan a tiempo, pero conllevan metástasis en fases más avanzadas y pueden suponer un peligro en la vida del paciente.

- **Melanoma:** Se trata de la variante más peligrosa de cáncer cutáneo. Su nombre se debe a que se origina en los melanocitos, encargados de la pigmentación de la piel. Es por ello, que suelen generarse sobre lunares o manchas. Se pueden curar si se detectan en una fase temprana, si no es así pueden y suelen conllevar la muerte del paciente.

2.1.2. Detección y diagnóstico del cáncer de piel

Para la detección de cualquiera de los tipos anteriores de cáncer se suele recomendar una auto inspección de la piel para observar posibles cambios, y el análisis de un dermatólogo especializado una vez al año. Este examen suele ir acompañado de dermatoscopia y, si es preciso, una pequeña biopsia (American Cancer Society, 2019).

En el caso de los CBC y CCE la detección es más sencilla por la localización, ya que suelen presentarse en zonas de constante impacto con la radiación UV (cara, extremidades…), y por la morfología distinguible que presentan. El basocelular se suele dar como áreas planas de color blanquecino o abultadas rojizas, presenta picazón y suelen ser parecidas a cicatrices. En el caso del espinocelular, aparecen como llagas que no se cierran y suelen conllevar costra y sangrado constante (Castañeda, 2016).

Es en los de mayor peligro, los melanomas, donde se encuentra la dificultad. Debido a que cualquier lunar, mancha o cambio de la piel puede ser señal de melanoma. Los signos que buscan los dermatólogos para su detección están basados en el acrónimo ABCDE (Simon, 2020):

- **Asimetría (A):** Si la mitad del lunar no mantiene cierta relación con la otra mitad.
- **Borde (B):** La aparición de bordes difusos, poco definidos o irregulares.
- **Color (C):** Si aparecen colores no uniformes o sombras de color marrón o negro.
- **Diámetro (D):** Si el lunar tiene más de 6 milímetros de ancho.
- **Evolución (E):** Si la mancha o lunar ha cambiado en tamaño, color o forma durante un periodo de tiempo.

Los lunares de un individuo suelen tener cierta similitud en estos aspectos. La aparición de una mancha que no respete este patrón puede considerarse igualmente peligrosa. A esta técnica se le conoce como la **regla del patito feo** (López-Sabater, 2019).

Como se puede observar, el cáncer de piel se detecta a partir de características físicas o aspectos visuales, en ocasiones con ayuda de un dermatoscopio. En la actualidad, la visión artificial está lo suficientemente avanzada como para realizar este tipo de clasificación. Y este tipo de tecnología podría dotar al médico general de cierto control sobre la situación. Esto aceleraría procesos y, como se ha detallado, el tiempo es un factor determinante en la vida del paciente.

2.2. Computer visión

La visión artificial es una de las ramas de *machine learning* que más auge está teniendo en los últimos años. El *machine learning* o aprendizaje automático (ML en adelante) es la rama de la inteligencia artificial cuya pretensión es que la máquina aprenda. En concreto, la visión artificial, busca replicar partes del sistema de la visión humana, como son la identificación y contextualización de situaciones a partir de imágenes (Le, 2018).

Además de la enorme cantidad de datos visuales que se generan hoy en día, un factor importante para que se desarrolle el campo de *computer vision* (CV en adelante) han sido la evolución del hardware con las tarjetas gráficas dedicadas (*GPU3*) y el desarrollo del *Deep Learning*. Un conjunto de sucesos que ha facilitado un aumento en la precisión de este tipo de sistemas (Esteva et al, 2021).

Este tipo de tecnología es ya habitual en la conducción autónoma o en sistemas de seguridad (Puentes, 2021). Y cada vez más en el campo de la salud gracias a la rapidez, precisión o la capacidad en el reconocimiento de patrones que son complicados de ver para un humano (Bushkovskyi, 2020).

Para que la utilización de CV sea viable es importante tener una cantidad importante de imágenes que comparar. Pero, estos datos no siempre están limpios o mantienen una relación entre ellos porque se han tomado con distintas cámaras o diferente perspectiva. La limpieza y normalización de estos se conoce como pre-procesado y es una tarea importante para conseguir que los patrones encontrados y el modelo sean lo más certeros posibles.

2.2.1. Modelo computacional de una imagen

Antes de entrar en el mundo del pre-procesado de los datos es importante la comprensión de 'cómo ve' las imágenes el ordenador, el modo en el que representa cada punto y lo almacena, para así tener control sobre los procesos que se van a realizar sobre estos.

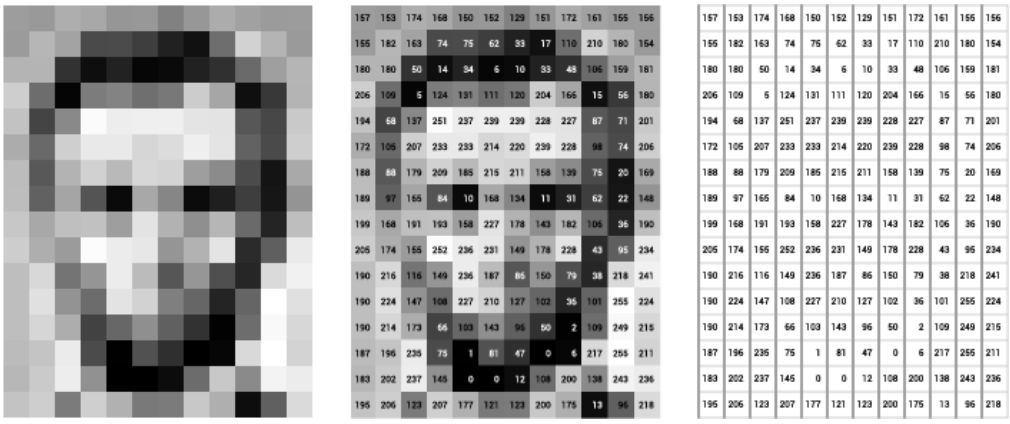


Figura 1: Diagrama de píxeles de una imagen (Levin, 2020).

Una imagen digital está formada por píxeles. Matemáticamente, está definida por la función *f(x,y)* donde *x* e *y* son las dos coordenadas horizontal y vertical. El valor de la función en cualquier punto supone el valor de un píxel. Aunque una imagen es una

matriz bidimensional, el ordenador almacena los píxeles en un array unidimensional, cuyos valores oscilan entre 0 (blanco) y 255 (negro) como se puede ver en la Figura 1 (Levin, 2020).

```
{157, 153, 174, 168, 150, 152, 129, 151, 172, 161, 155, 156,
 155, 182, 163,  74,  75,  62,  33,  17, 110, 210, 180, 154,
 180, 180,  50,  14,  34,   6,  10,  33,  48, 106, 159, 181,
 206, 109,   5, 124, 131, 111, 120, 204, 166,  15,  56, 180,
 194,  68, 137, 251, 237, 239, 239, 228, 227,  87,  71, 201,
 172, 105, 207, 233, 233, 214, 220, 239, 228,  98,  74, 206,
 188,  88, 179, 209, 185, 215, 211, 158, 139,  75,  20, 169,
 189,  97, 165,  84,  10, 168, 134,  11,  31,  62,  22, 148,
 199, 168, 191, 193, 158, 227, 178, 143, 182, 106,  36, 190,
 205, 174, 155, 252, 236, 231, 149, 178, 228,  43,  95, 234,
 190, 216, 116, 149, 236, 187,  86, 150,  79,  38, 218, 241,
 190, 224, 147, 108, 227, 210, 127, 102,  36, 101, 255, 224,
 190, 214, 173,  66, 103, 143,  96,  50,   2, 109, 249, 215,
 187, 196, 235,  75,   1,  81,  47,   0,   6, 217, 255, 211,
 183, 202, 237, 145,   0,   0,  12, 108, 200, 138, 243, 236,
 195, 206, 123, 207, 177, 121, 123, 200, 175,  13,  96, 218};
```

Figura 2: Array de píxeles (Mihajlovic, 2019).

El modelo de representación de color más habitual en imágenes digitales es el RGB (*red*, *green* and *blue*), que se basa en representar cada píxel con tres valores, uno por color, manteniendo la misma escala 0 a 255 sobre el color correspondiente (Mihajlovic, 2019). De modo que, si una imagen en escala de grises se almacena como se muestra en la Figura 2, una imagen a color supondrá el triple de valores para almacenar y manejar.

2.2.2. Pre-procesado

Como se ha descrito anteriormente, este paso es muy importante, ya que los pasos posteriores dependerán de él. Consiste en la utilización de diferentes algoritmos para eliminar distorsiones no deseadas, mejorar características para el posterior procesamiento y eliminación de datos no deseados (Tadakaluru, 2020).

- **Re-escalado:** Se suele cambiar el tamaño de la imagen con el fin de que todas las imágenes del *dataset* mantengan las mismas medidas.
- **Corrección de brillo:** Esta transformación se encarga de normalizar la iluminación y el color para cada una de las imágenes. Este tipo de transformación será útil para homogeneizar el *dataset*.
- **Transformaciones geométricas:** Se utilizan transformaciones como la rotación, el escalado o la traslación para eliminar distorsiones geométricas que puedan suceder durante la captura de la imagen.

- **Filtrado y segmentación:** El objetivo de este tipo de transformación es obtener información de interés como bordes, esquinas o manchas.
- **Transformada de Fourier y restauración de imagen:** Este tipo de transformaciones se utilizan para filtrado, reconstrucción y compresión de imágenes.

Estas son algunos de los posibles tipos de transformación que se pueden realizar sobre imágenes. Dentro de cada uno existen algoritmos específicos que favorecerán a la detección de la imagen en cierto aspecto.

En este proyecto, se tratará de normalizar y corregir la iluminación de las imágenes, eliminar ruidos como el vello y separar el lunar del fondo (segmentación) para obtener los parámetros de interés.

2.2.3. Clasificación de imágenes

Existen diferentes aplicaciones dentro de CV. Para este proyecto, interesa la clasificación de imágenes. Mediante la clasificación, un ordenador puede analizar una imagen e identificar su 'clase' (Jain, 2019). En este caso, esta clase hará referencia al grado de malignidad que presente un lunar.

Para un humano es una tarea innata y natural, pero, como se ha descrito anteriormente, el ordenador tiene una concepción diferente de las imágenes. Por ello, se precisará de una base de datos de entrenamiento que le permita aprender acerca de los diferentes casos y las características que definen a cada imagen.

Dependiendo de si el conjunto de entrenamiento está previamente clasificado o no, las técnicas de clasificación se considerarán de aprendizaje **supervisado** o **no supervisado**, respectivamente (Deepan, 2020).

Además del pre-procesado descrito en la sección anterior, para la clasificación de imágenes se han de llevar a cabo las siguientes acciones (Sanghvi, 2020):

- **Detección de un objeto:** Localización de un objeto, segmentación de la imagen y la identificación de la posición del objeto de interés. En este proyecto, al tratarse de imágenes macroscópicas, suelen ser un proceso relativamente sencillo.
- **Extracción de características y entrenamiento:** Se utilizan clasificadores para identificar patrones de las imágenes, características que podrían ser únicas para

una clase en particular y que, posteriormente, ayudarán al modelo a diferenciar entre diferentes clases. Esto es lo que se conoce como entrenamiento del modelo.

- **Clasificación:** En este paso se introducen imágenes nuevas al modelo, han de ser datos no utilizados para el entrenamiento. El modelo ha de ser capaz de detectar la clase de la imagen en base a las características de esta nueva imagen y la referencia que haya creado gracias al resto de datos.

El tipo de clasificador que se escoja tendrá ciertas ventajas o desventajas. Los clasificadores son algoritmos que toman un vector de características de la imagen y, en base a este, genera una salida indicando la clase. Cuando la salida es continua, es decir, un número real, a la técnica se le denomina regresión. A continuación, se recogen las técnicas de aprendizaje máquina más utilizadas en la clasificación y regresión de imágenes.

Support Vector Machines (SVM)

Son algoritmos de aprendizaje automático supervisado utilizados tanto para clasificación como para regresión. Proporciona versatilidad tanto con variables continuas como discretas (Sanghvi, 2020).

Su objetivo es encontrar el hiperplano en un espacio de N dimensiones (N es el número de características) que mejor clasifique a los datos (Gandhi, 2018). El hiperplano óptimo será el que tenga el máximo margen, es decir, mayor distancia entre los puntos de distintas clases. Los vectores de apoyo son los que están más cerca del hiperplano y se encargan de ajustar su orientación y posición.

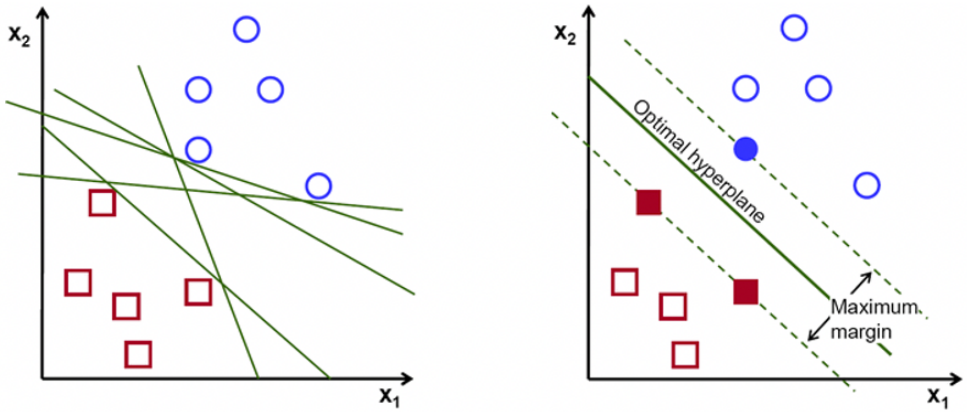


Figura 3: Hiperplanos posibles (Gandhi, 2018).

Pero la separación entre clases no va a ser siempre separable linealmente como se muestra en Figura 3. Es por ello por lo que los SVM se basan en una función *kernel*. Esta función se encarga de mapear los datos originales a un espacio de mayor dimensión de modo que sean separables (Afonja, 2017). Esta función puede ser de diferentes tipos, *Linear*, *Gaussian* y *Polynomial* son los más usados.

K Nearest Neighbors

Es uno de los algoritmos de aprendizaje automático más sencillo. Útil tanto para clasificación y como para regresión. Es un algoritmo de aprendizaje supervisado, no paramétrico y *lazy* (perezoso), es decir, se aproxima la función sin realizar cálculos hasta la evaluación (Harrison, 2018).

Este algoritmo se basa en que los atributos similares (misma clase) son lo que más cerca se encuentran entre sí, es decir, se utiliza una medida de distancia entre valores que determina la clase. Funciona de modo que se seleccionan el número k de vecinos, al introducir nuevos datos se calcula la distancia entre los nuevos datos y los datos de entrenamiento, ya clasificados, y se le asigna la clase que más k vecinos cercanos tenga como se puede observar en la Figura 4 (Dwivedi, 2020). Las medidas de distancia más usadas son la euclídea y la Manhattan.

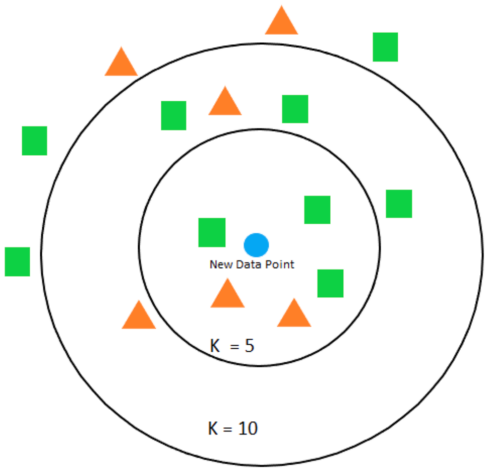


Figura 4: Ejemplo KNN (Rath, 2019).

Artificial Neural Networks (ANN)

Esta técnica utiliza algoritmos estadísticos inspirados en el funcionamiento del cerebro humano (Wang, 2003). De igual manera que ocurre con las neuronas biológicas, una ANN consiste en un conjunto de elementos de procesamiento interconectados, llamados nodos.

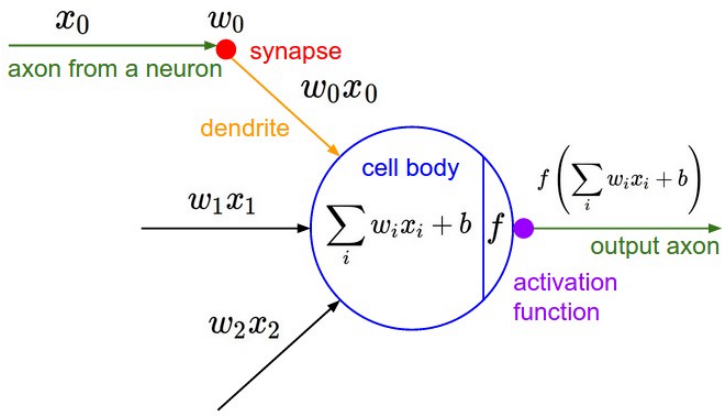


Figura 5: Comparativa entre neurona biológica y artificial (Khan, 2019).

Una ANN suele estar compuesta por una capa de entrada, una o varias capas *hidden* (escondidas) y una capa de salida, conectadas de diferentes maneras (Figura 6). Estas conexiones tienen valores numéricos llamados pesos que se van alterando sistemáticamente con cada nueva entrada, ajustando el modelo a la salida deseada.

La actividad de una red neuronal viene determinada por las funciones de transferencia o activación, la regla de aprendizaje y por la propia arquitectura. La función de activación tiene como función acotar el valor de la neurona para que la red neuronal no quede paralizada por neuronas divergentes. La regla de aprendizaje se encarga de ajustar los pesos según cierto criterio.

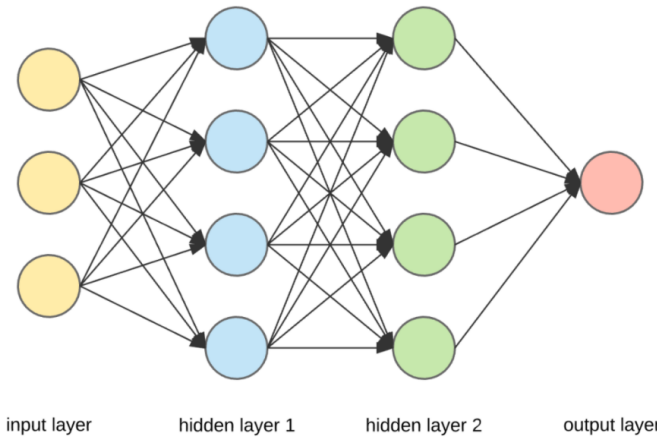


Figura 6: Arquitectura ANN (Lelli, 2019).

Convolutional Neural Networks (CNN)

Es uno de los tipos de ANN más utilizado para la clasificación de imágenes sobre aspectos médicos (Tarasenko, 2020). Este tipo de red neuronal se caracteriza por que su arquitectura está inspirada en la corteza visual, dónde las neuronas individuales responden a estímulos focalizados en una región (Campo Receptivo).

Son algoritmos multietapa que aprenden características de forma jerárquica y automática. Primero identifican características de bajo nivel y luego aprenden a reconocer y combinar estas características para aprender patrones más complicados (Jmour, 2018).

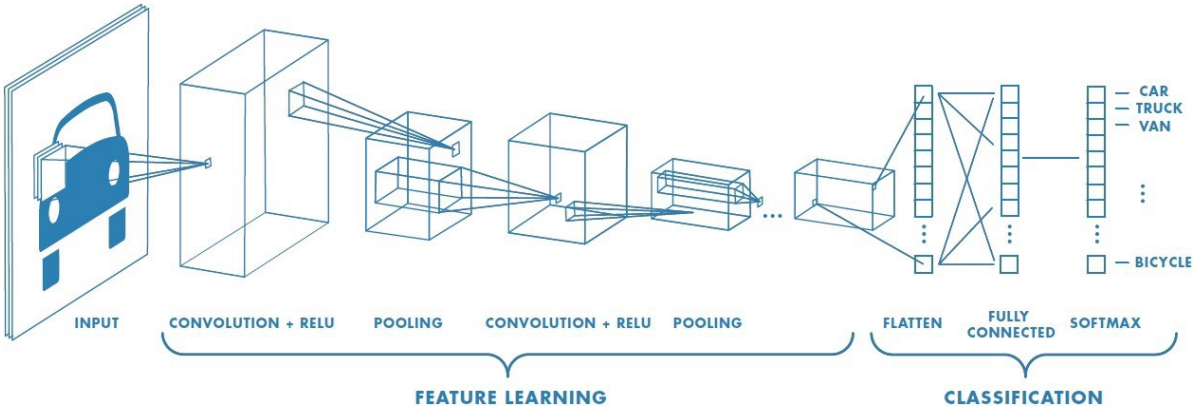


Figura 7: Arquitectura CNN (Saha, 2018).

Cuando se habla de clasificación de imágenes, las redes convolucionales constan de dos fases (Figura 7):

- **Aprendizaje de características:** Esta fase recibe la imagen de entrada, se encarga de extraer características mediante diferentes 'filtros'. Está formada por capas convolucionales y capas de agrupación o *pooling* (Saha, 2018). La cantidad y disposición de estas capas son factores importantes que afectarán sobre la salida final.

  Las capas convolucionales utilizan la matriz de píxeles de entrada y la multiplican por una matriz más pequeña de valores o pesos aleatorios (kernel) que irán ajustándose. Una convolución es una matriz de salida que, unida a la función de activación, actúa como mapa de detalle, tal y como muestra la Figura 8. A más convoluciones, mayor precisión sobre las características de la imagen.

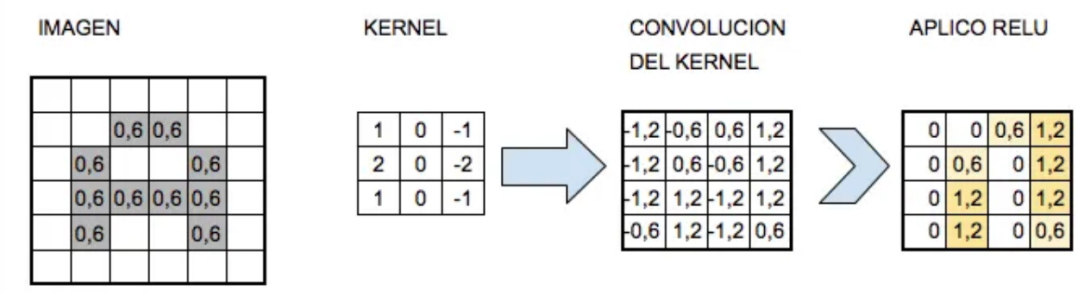


Figura 8: Convolución y función de activación sobre una sección (Na8, 2018).

Las capas de agrupación se encargan de reducir el número de parámetros de la capa que recibe como entrada (Calvo, 2017). Para ello, utiliza estadísticos como el promedio (Average Pooling) o el máximo (Max Pooling) sobre una región fija. Esta reducción de dimensión disminuye el peso computacional del algoritmo y, en muchas ocasiones, es útil para extraer características dominantes o eliminar ruido.

Esta parte de la CNN aporta como salida un vector que concatena todas las informaciones y características obtenidas.

- **Clasificación:** Esta fase tiene como arquitectura una capa totalmente conectada (normalmente un *perceptrón multicapa*) que recibe la salida de la capa convolucional y, mediante iteraciones, consigue combinaciones de las características de alto nivel, realizando así la clasificación.

Transfer Learning

El aprendizaje por transferencia, más conocido como *Transfer Learning*, es una técnica de ML basada en la psicología educativa y en el modo en que los humanos transfieren conocimiento (Zhuang et al, 2020). Su objetivo es aprovechar el conocimiento previo sobre un dominio origen mejorando el rendimiento del aprendizaje en el dominio objetivo (Wei et al, 2018). Como se muestra en la Figura 9, aprender a llevar una moto compartirá con la bici el dominio de los vehículos de dos ruedas.

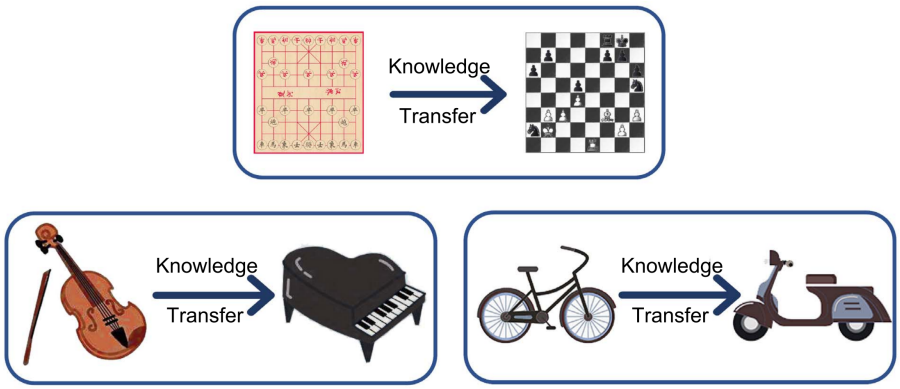


Figura 9: Ejemplos de transferencia de conocimiento (Zhuang, 2020).

La transferencia de conocimiento puede afectar negativamente. Esto surge cuando la similitud entre los dominios no es la que se espera o cuando el dominio no influye en el aprendizaje. Cuando, cómo y qué transferir son cuestiones importantes para evitar esta transferencia negativa (Pan, 2010).

Si se entiende el dominio como el ámbito de estudio, con su espacio de características específico, y la tarea el resultado que se obtiene introduciendo entradas a cierto dominio, se puede afirmar que existen un par dominio-tarea origen y objetivo. Con estos conceptos, Pan y Chang (2010) dividen el *Transfer Learning* en tres subconjuntos:

- **Inductivo:** La tarea destino es diferente a la tarea origen, sin importar los dominios. Se precisa que algunos de los datos estén etiquetados. Este método se subdivide a su vez en dos grupos.

    a. Si se utilizan muchos datos etiquetados en el dominio origen. La funcionalidad es muy similar al del aprendizaje multitarea (*CNN*) enfocándose únicamente en la tarea objetivo.

    b. Se utilizan datos no etiquetados en el origen. El espacio de etiquetas entre origen y objetivo son diferentes, por lo que, igual que en el caso anterior, estará centrado en la tarea objetivo.

- **Transductivo:** Las tareas destino y objetivo son las mismas, los dominios diferentes. Este método se subdivide a su vez en dos grupos.

    a. Los espacios de características entre dominio origen y objetivo son diferentes.

    b. Los espacios de características entre dominios son iguales.

- **No supervisado:** La tarea objetivo no está relacionada con la tarea origen. Los datos no están etiquetados ni en origen, ni en objetivo.

En el caso de la clasificación de imágenes, al tratarse de un problema de aprendizaje supervisado o semisupervisado, se suele aplicar el *Transfer Learning Inductivo*. Para este tipo de técnica se utilizan modelos pre-entrenados mediante bases de datos de imágenes enormes, como es el conocido *ImageNet*. De este modo, se potencia la extracción de características para, en las capas superiores, realizar la tarea de clasificación sobre el dominio objetivo.

2.3. Inteligencia Artificial en sanidad

El constante desarrollo de la Inteligencia Artificial ha demostrado la capacidad de muchos campos para beneficiarse de los conocimientos que las técnicas IA obtienen de los datos (Esteva et al, 2021).

Uno de los sectores que está liderando la implementación y el desarrollo de IA es la medicina. La naturaleza de reconocimiento de patrones visuales en tareas de diagnóstico de algunas ramas como la oftalmología o la dermatología, y la creciente disponibilidad de imágenes estructuradas, han fomentado la evolución de la visión por ordenador. Este tipo de herramientas están consiguiendo niveles de precisión en el diagnóstico iguales o superiores al de los expertos (Bhattad, 2020).

En dermatología, algunas entidades han fomentado la creación de herramientas con repositorios de imágenes de libre acceso y la proposición de retos IA que animan a los equipos de desarrollo a competir. La incorporación de este tipo de algoritmos puede suponer un apoyo para médicos no especializados, tanto en el diagnóstico como en la asistencia y la gestión de situaciones de urgencia.

La IA tiene el potencial para mejorar la medicina y, a la misma vez, puede ser una herramienta peligrosa si comete errores. Cuanto más se utilice, más datos y mayor precisión tendrán los algoritmos. Será importante comprender los datos y el entorno sobre los que se construye un modelo. Al tratarse de herramientas de intervención médica, el desarrollo de estas implica una evaluación equivalente a la de las pruebas o los tratamientos habituales (Watson et al, 2019).

A continuación, se plantean las consideraciones que hay que tener en cuenta a la hora de aplicar IA en sanidad (Esteva et al, 2021):

- **Evaluación de los datos:** La calidad de los datos determina la calidad del modelo. Es importante detectar las desigualdades entre datos y realizar las operaciones oportunas para la reducción o supresión de sesgos.

- **Planificación de las limitaciones del modelo:** Es importante conocer los casos concretos en los que el modelo falla. Esto permitirá dirigir la obtención de nuevos datos y controlar el entrenamiento para mejorar los resultados.

- **Participación de la comunidad:** La colaboración de pacientes, médicos y de todos aquellos implicados en estos sectores, es muy importante para que este tipo de tecnología siga evolucionando.

- **Creación de confianza:** La confianza provendrá de los ensayos clínicos que validen el algoritmo en el mundo real, donde existen factores humanos y sociales difícilmente calculables. También será importante ofrecer seguridad sobre datos sensibles para mantener la confianza del paciente.

2.3.1. Proyectos existentes

Como se ha descrito, innovar en este sector es una tarea complicada. En la actualidad existen gran cantidad de aplicaciones y artículos que desarrollan sistemas de clasificación para problemas dermatológicos, pero hay pocas que presenten validación clínica. Con un desarrollo adecuado y su correspondiente evaluación, se deberían alcanzar niveles suficientemente altos de acierto en el diagnóstico como para ser funcionales.

En esta sección, se recogen algunas de las muchas herramientas relacionadas con este proyecto, con el fin de analizar la situación e incluso encontrar una solución que pueda mejorar las limitaciones que las caracterizan.

SkinVision

Se trata de una aplicación que evalúa el estado de manchas o lunares mediante una imagen del lunar en cuestión, como se muestra en la Figura 10, tomada desde el *smartphone* (SkinVision, 2021) (Peverelli, 2017).

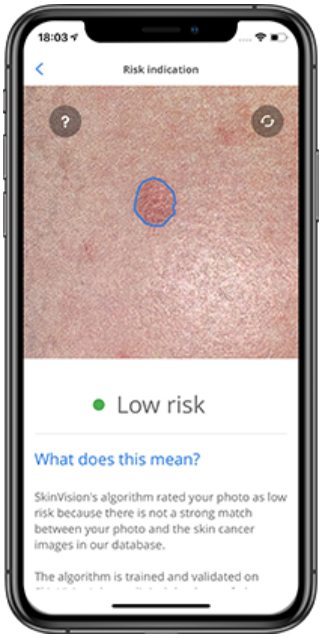


Figura 10: Evaluación de SkinVision (SkinVision, 2021).

Además, permite realizar un seguimiento de las manchas guardando las fotos en la aplicación y ofrece recomendaciones sobre los pasos a seguir tras aportar el resultado de una evaluación.

La herramienta se basa en una *Support Vector Machine* (SVM) con núcleo radial. Para la elección de los parámetros del SVM, se utiliza la función de optimización del enjambre

de partículas (Udrea, 2019), que imita el movimiento y crecimiento de las partículas en la naturaleza. Esto permite predecir la posible evolución del sistema en base a las características que se escojan.

En definitiva, una herramienta que soluciona el problema planteado en el proyecto y que se ha descargado en más de 1,2 millones de dispositivos. En cuanto a los problemas que se pueden encontrar, destacan:

- La complejidad del algoritmo requiere de una buena conexión a internet y se toma 30 segundos hasta aportar la evaluación.
- La precisión de la herramienta es del 73%, un valor que a nivel sanitario no aporta la confianza suficiente (Pozzebon, 2014).
- Existen numerosos artículos que demuestran la sobreestimación de la herramienta (Deeks, 2020).

Google AI-powered dermatology tool

La gran multinacional está desarrollando una aplicación basada en web que utiliza IA para la detección de enfermedades en piel, pelo y uñas. Funciona introduciendo tres imágenes de la zona afectada desde distintos ángulos y ciertos parámetros sobre la posible afección (Bui, 2021). Como resultado, aportará información revisada por dermatólogos, preguntas y respuestas comunes e imágenes coincidentes de la web, como se puede observar en la Figura 11.

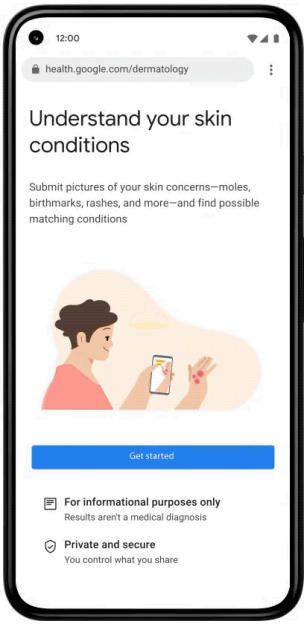


Figura 11: Google AI-powered dernatology tool (Bui, 2020).

Para generar un modelo flexible y sin sesgos, tiene en cuenta factores como la edad, el sexo, el tipo de piel, etc. Además, se ha utilizado una base de datos con 65.000 imágenes clasificadas por afección y grupo demográfico.

La arquitectura que presenta esta tecnología es la mostrada en la Figura 12, un Deep Learning System (DLS) compuesto por (Liu, 2020):

- Una red pre-entrenada *Inception-v4* con ponderaciones compartidas para cada una de las tres imágenes, para luego ser reducidas mediante una capa que obtiene un valor medio de los resultados.
- Los metadatos pasarán por una capa de extracción de características.
- Todos estos datos se concatenan para ser pasados por dos funciones Softmax que aportarán dos resultados. El principal aportará una probabilidad entre 27 enfermedades y el secundario una probabilidad entre 419.

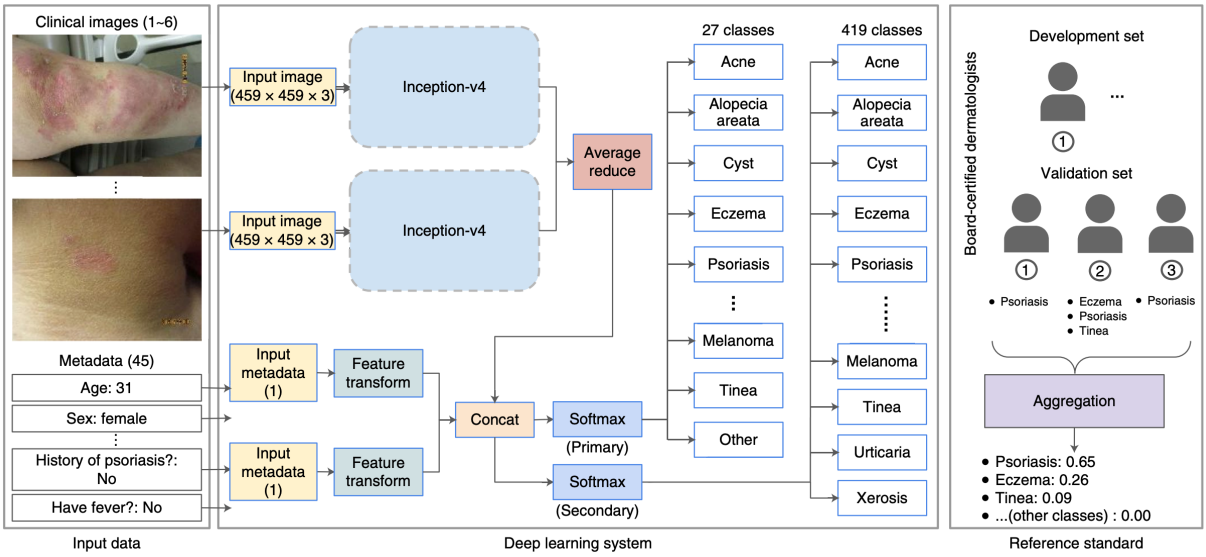


Figura 12: Arquitectura de Google AI-powered dermatology tool (Liu, 2020).

Aunque el proyecto lleva 3 años en desarrollo, ya ha sido validado clínicamente en la UE. Para conseguir esto, 40 clínicos no especialistas interpretaron imágenes sin clasificar. La mitad utilizaron la herramienta como ayuda, mientras que la otra mitad no. Los resultados demostraron que la ayuda de la IA mejoró la probabilidad de predicción de los médicos de cabecera y enfermeros en 26 afecciones (Jain et al, 2021).

Se puede concluir que esta herramienta es, además de más actual, más completa que la descrita anteriormente. Presenta cierto grado de validación y aporta un alto nivel confianza profesional. La utilización de varias imágenes y metadatos para el diagnóstico permite tener más control sobre aspectos como la evolución o la diferencia con otros

lunares. En definitiva, Google está desarrollando una aplicación con una perspectiva prometedora y, no pudiendo encontrar errores realistas, se convierte en un ejemplo interesante a largo plazo.

Artículos de investigación

Además de las diferentes aplicaciones y herramientas, se han encontrado estudios y artículos que investigan clasificaciones en imágenes y aplicación de IA en medicina que pueden aportar enfoques de interés para el desarrollo del presente proyecto. A continuación, se presentan los más interesantes:

- El primer proyecto (ALKolifi, 2019) desarrolla un sistema de clasificación que diferencia entre melanoma, eczema y psoriasis que utiliza una red convolucional pre-entrenada para la fase de extracción de características, y un SVM para la clasificación de 100 imágenes. Si bien es cierto que el *dataset* de entrenamiento es muy limitado y, el modelo clasificador, podría considerarse sesgado, el resultado de la evaluación es del 100% de precisión.

- El segundo de los proyectos (Zhang et al, 2019) basa su modelo en una versión optimizada de una red neuronal convolucional. Normalmente, la función de optimización que se utiliza es la del descenso de gradiente. Pero en este caso, el autor, utiliza la optimización de ballena (WOA) mediante el mecanismo de vuelo de Levy.

  Este algoritmo comienza con un conjunto de soluciones aleatorias que se irá comparando hasta alcanzar el óptimo global del problema. El mecanismo de vuelo evita uno de los problemas que presenta el WOA que es la convergencia prematura y con ello la precisión de la optimización.

  Para el entrenamiento utiliza un dataset de 22.000 imágenes y los resultados han sido muy superiores a los obtenidos con otros algoritmos. Esto no significa que suponga una solución al problema de la validación clínica, aunque pueda intuirse un resultado positivo en este aspecto.

- El tercer proyecto (Tschandl, 2018) se basa, de igual modo que el primero, en la segmentación y obtención de características de manera automatizada. Para el desarrollo del modelo se combinan las siguientes técnicas:

  a. Pre-entrenamiento de los pesos de una red neuronal convolucional completamente conectada.

b. Ampliación de la arquitectura LinkNet para reutilizar las capas superiores de ResNet.

c. Post-procesamiento de las máscaras de segmentación de lesiones cutáneas con *Conditional Random Fields* (CRFs), que es un tipo de modelado estadístico que clasifica teniendo en cuenta el contexto.

Para evaluar la herramienta se ha comparado el resultado obtenido por la red neuronal y tres dermatólogos con años de experiencia, obteniendo la primera mejores resultados.

- El último de los artículos (Phillips et al, 2019) es un estudio que compara los resultados que se obtienen con *Deep Ensemble for the Recognition of Malignancy* (DERM), centrado en la detección de melanomas, y el análisis de médicos.

  Esta herramienta consta de una red neuronal con una arquitectura específica, desarrollada por *Skin Analytics* para el problema en cuestión. El resultado de la evaluación demostró que este tipo de modelos tienen el potencial necesario para ser utilizado como herramienta de apoyo en el diagnóstico de melanomas.

Se pueden encontrar gran cantidad de herramientas con redes específicas y muy buenos resultados, cada vez más enfocados en el *Transfer Learning*. Aprovechando la fase de extracción de características en manos de la red pre-entrenada se aprovecha el rendimiento de una cantidad de datos superior y se obtienen mejores resultados en la predicción.

Ante tal cantidad de herramientas con tan buenos resultados, se considera complicado mejorar la situación actual en este sector en un proyecto de estas dimensiones, y es por esto que se decide estudiar la utilidad del aprendizaje por transferencia en la detección del cáncer de piel. Para ello, se realizará un análisis de los diferentes modelos pre-entrenados que existen y se analizará exhaustivamente aquellos que propongan mejores resultados.

3. Objetivos concretos y metodología de trabajo

En este capítulo se recogen los objetivos generales y específicos, definidos en base al contexto recogido en el capítulo anterior. Además, se plantea la metodología que se va a llevar a cabo durante el desarrollo del proyecto.

3.1. Objetivo general

El objetivo general que se busca con este proyecto es: **Analizar el uso del aprendizaje por transferencia como sistema de clasificación de imágenes para el diagnóstico de cáncer de piel**. Para el desarrollo de este sistema se han utilizado diversos conjuntos de datos con imágenes de lunares y manchas clasificadas por médicos especializados, aportados por diversas entidades (ISIC, PH2 y EDRA) con el fin de investigar en este ámbito. Con estas imágenes se entrenará y evaluarán los diferentes modelos pre-entrenados, con el fin de obtener la precisión que presentan, compararlos y analizar la viabilidad clínica que puedan presentar.

3.2. Objetivos específicos

Los objetivos específicos que se han definido para el desarrollo de este proyecto son los siguientes:

- Elaborar un conjunto de datos limpio y homogéneo que permita entrenar los modelos sin generar sesgos.
- Explorar los diferentes modelos pre-entrenados existentes, evaluarlos para el conjunto generado y escoger aquellos que mejor proyección muestren.
- Determinar los parámetros y características que se utilizarán para generar el sistema base.
- Construir el sistema general de clasificación.
- Entrenar y validar el sistema con parte del *dataset* generado para cada situación.
- Evaluar los diferentes modelos con el conjunto de datos restante y obtener los parámetros de calidad que aporten.
- Comparativa de los resultados y conclusiones.

3.3. Metodología del trabajo

Para el desarrollo de este proyecto se va a utilizar una de las metodologías más utilizadas para el desarrollo de proyectos que conllevan inteligencia artificial (Moreira, 2018). Esta metodología es CRISP-DM (CRoss-Industry Standar Process for Data Mining) y consta de las seis fases que se muestran en la Figura 13. A continuación, se describen cada una de ellas de manera más detallada:

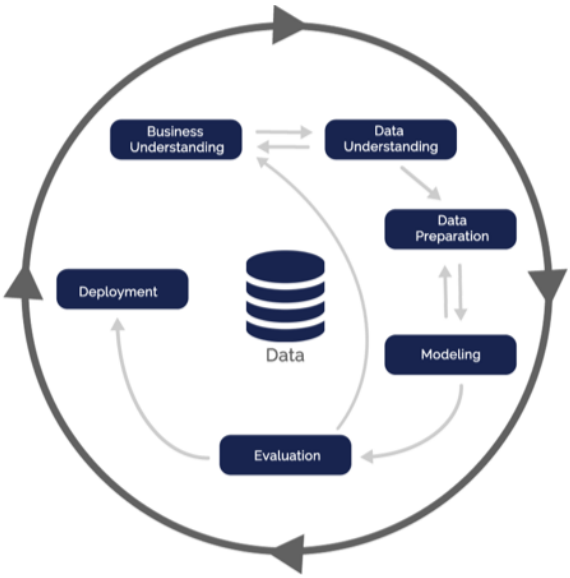


Figura 13: Fases de la metodología CRISP-DM (Moreira, 2018).

Entendimiento del tema

Esta etapa es la recogida en cada uno de los apartados anteriores. En base a un estudio exhaustivo de todo ámbito que rodea al proyecto, como es 2. Contexto y estado del arte, se elaboran una serie de requisitos sobre los que este se desarrollará, 3. Objetivos concretos y metodología de trabajo.

Comprensión de los datos

En este caso se comienza con la recogida de datos que vayan a ser de utilidad para el desarrollo del proyecto. Estos datos suelen obtenerse de bases de datos públicas, pero también se pueden obtener de fuentes privadas si se hace el estudio de privacidad que este conlleva. Será importante comprender este conjunto de datos y los procesos que estos requieren. Como son la limpieza del conjunto, la segmentación y el filtrado en el caso de las imágenes.

Preparación de los datos

Esta fase es en la que se aplicarán todos los procesos que se necesiten realizar sobre los datos para prepararlos para la construcción del algoritmo.

Modelado

Con los datos ya tratados, en esta etapa se realizan todas las operaciones específicas que se requieren para obtener el sistema, en este caso de clasificación, incluido el entrenamiento.

Evaluación

En esta fase se comprobará el nivel de precisión de los diferentes modelos, se realizarán pruebas en un entorno real y se sacarán conclusiones en base a las expectativas que se hayan generado.

Implementación

En esta última etapa es en la que se guarda y descarga el modelo activo y funcional. En una situación comercial habría que entregar dicho modelo o bien desarrollar un artículo que recoja la investigación realizada. Al tratarse de un proyecto de investigación, que no propone una solución software, el resultado de la fase de producción es esta memoria.

4. Desarrollo del modelo

Tras el análisis y comprensión del problema que abarca este proyecto, en este capítulo, se detallan el resto de las fases de la metodología descrita en el apartado anterior.

4.1. Descripción

El proyecto consiste en el estudio de los resultados que aporta una herramienta general que, con diferentes modelos pre-entrenados, permita clasificar imágenes en una de las siguientes cuatro clases:

- **Melanoma.**
- **Carcinoma de células escamosas.**
- **Carcinoma basocelular.**
- **Mancha benigna o nevus.**

Para el desarrollo de esta herramienta se utilizará el lenguaje de programación *Python*. Como se presenta en el Anexo I. Python, esto se debe a lo intuitivo que resulta para el manejo de datos, el hecho de ser multiplataforma y, sobretodo, la amplia comunidad que lo representa. Cada uno de los procesos que aparecen en este capítulo se pueden encontrar en diferentes *scripts* recogidos y proporcionados en el Anexo II. Código del proyecto.

4.2. Obtención y comprensión de los datos

Para el desarrollo de un modelo fiable y sin resultados sesgados, se precisa una base de datos extensa y dedicada al problema en cuestión. Se encontraron gran cantidad de bases de datos con imágenes de enfermedades de piel, muchas de ellas sin clasificar y otras muchas no relacionadas con el proyecto.

Por ello, para la construcción de la base de datos se han utilizado tres conjuntos diferentes de los cuales se han obtenido todas las imágenes de cada una de las clases mencionadas en el apartado anterior. Los tres conjuntos de datos cumplen los requisitos de privacidad que este tipo de información sensible precisa. Los *datasets* utilizados son los siguientes:

- **International Skin Imaging Collaboration (ISIC):** Se trata de una asociación diseñada para reunir y compartir imágenes digitales de la piel con el fin de investigar sobre enfermedades dermatológicas y reducir la tasa de mortalidad en estas. Esta asociación presenta un archivo de código abierto con hasta 150.000 imágenes proporcionadas por centros médicos especializados de todo el mundo, de las cuales casi 70.000 son públicas. Cada imagen se encuentra asociada a un grupo de metadatos que la describen clínicamente (ISIC, 2020). Se trata de datos estructurados, por lo que se pueden realizar consultas y filtrados específicos.

  Este *dataset* conforma el grueso del conjunto que se ha generado para este proyecto. Por ello, para la categorización y nomenclatura de clases e imágenes se ha utilizado el criterio de ISIC. Los datos se han obtenido gracias a un *downloader* generado por Avineri (2020) que se puede obtener en el siguiente enlace: https://github.com/GalAvineri/ISIC-Archive-Downloader

- **PH2:** Se trata de una base de datos de imágenes dermatoscópicas adquiridas en el Servicio de Dermatología del Hospital Pedro Hispano en Portugal. Conformado por 200 imágenes centradas en el diagnóstico de melanomas. Por ello solo encontramos melanomas y nevus de distintos tipos (Mendonca et al, 2013). Además de las originales, presenta las imágenes segmentadas, es decir, con un pre-procesado previo.

  Los datos se pueden encontrar en la siguiente dirección: https://www.fc.up.pt/addi/ph2%20database.html

- **The Interactive Atlas of Dermoscopy (EDRA):** Este conjunto de datos se crea con el fin de formar al personal médico en las lesiones cutáneas. Consta de 2045 imágenes. Cada caso tiene datos clínicos, resultados histopatológicos, diagnóstico y nivel de dificultad en la detección que se presenta en forma de ficheros *csv*. Los diagnósticos incluyen varias enfermedades, como el melanoma o la queratosis serborreica, y subtipos dentro de ellas (Argenziano et al, 2002).

  El conjunto de datos se ha obtenido gracias al siguiente enlace: https://derm.cs.sfu.ca/Download.html

4.3. Preparación de los datos

En este apartado se describe el proceso por el que los datos son preparados específicamente para ser introducidos en un modelo de clasificación de imágenes. Esto constará dos fases: **Homogeneización** de los tres *datasets* y el **Pre-procesado** de los datos homogeneizados.

4.3.1. Homogeneización del conjunto

Cada uno de los conjuntos utilizados contienen *metadatas* diferentes, con nomenclaturas distintas. Este apartado recoge el proceso por el que se aporta criterio y formato a los tres *datasets* de modo que no haya errores de consistencia. Se eliminarán todas aquellas columnas que no sean de interés para este estudio concreto. Este proceso ha sido desarrollado con la ayuda de librerías específicas para el manejo de datos como son *NumPy* (NumPy, 2021) o *Pandas* (Pandas, 2021), se puede acceder al código en el apartado Homogeneización de los datos del Anexo II. Código del proyecto.

Al tratarse de una herramienta destinada a la clasificación de imágenes, solamente se necesitará la identificación de la imagen y las etiquetas o *labels*. Los *labels* hacen referencia a la clase que pertenece, es decir, al tipo de enfermedad que presenta la imagen. Como criterio general para esta variable se ha escogido el utilizado en ISIC, ya que es el conjunto que más imágenes aporta. Este nombra las cuatro clases en inglés: **melanoma**, **basal cell carcinoma**, **nevus** y **squamous cell carcinoma**. Todas las imágenes que se correspondían con una clase diferente fueron descartadas.

Como identificador se han creado nombres relacionados para añadir las imágenes a un nuevo *dataset* conjunto. El nombre tiene como formato estándar la palabra 'SKIN_' seguida de un número.

id	label
SKIN_0	nevus
SKIN_1	nevus
SKIN_2	basal cell carcinoma
SKIN_3	nevus
SKIN_4	nevus
...	...
SKIN_38592	nevus
SKIN_38593	nevus
SKIN_38594	nevus
SKIN_38595	nevus
SKIN_38596	melanoma

Tabla 1: Estructura del archivo metadata.

Como se puede observar en la Tabla 1,se mantiene un *dataset* de más de 38.000 imágenes. Pero, al analizar el número de imágenes por cada una de las clases, se observa una desproporción clara entre ellas, donde los *nevus* o imágenes benignas predominan, pudiendo provocar un sesgo positivo en el modelo (Tabla 2).

basal cell carcinoma	3438
melanoma	5890
nevus	28613
squamous cell carcinoma	1525

Tabla 2: Distribución de imágenes por clase.

La única manera de equilibrar el modelo ante el posible sesgo es aproximar o igualar el número de imágenes del resto de clases a aquella que presente la cantidad menor, que es el carcinoma de células escamosas. Por tanto, el procedimiento consiste en eliminar imágenes hasta llegar a este valor para cada clase. De este modo, se obtiene un *dataset* final cercano a las 6000 imágenes (Tabla 3).

num imágenes	6100
basal cell carcinoma	1525
melanoma	1525
nevus	1525
squamous cell carcinoma	1525

Tabla 3: Distribución final del dataset.

El conjunto final de datos mantiene la homogeneidad que se andaba buscando para el entrenamiento y evaluación de un modelo destinado a la clasificación sin sesgos. Ahora habrá que tratar las imágenes para destacar la enfermedad de los elementos ruidosos, es decir, habrá que realizar el pre-procesado.

4.3.2. Pre-procesado

En esta sección se describen cada una de las operaciones realizadas sobre las imágenes de modo que se consideren preparadas para ser introducidas en el sistema de clasificación.

En este caso concreto, el objetivo es resaltar la zona de la posible lesión, y que el resto de los elementos (pelos, otras marcas, fondo…) no influyan en la clasificación. Es importante seguir el orden en el que se describen los algoritmos para obtener un resultado óptimo.

Para el desarrollo de la mayoría de los procesos se ha utilizado la librería *OpenCV*, destinada principalmente al desarrollo de *Computer Vision* y que se describe más detalladamente en el Anexo I. Python. El Anexo II. Código del proyecto, presenta el código en el apartado de Pre-procesado.

Re-escalado

Este proceso consiste en la redimensión de las imágenes a un tamaño estándar (Figura 14). Esto se realiza para evitar imágenes demasiado grandes que influyan en la carga de trabajo del sistema. No se debe re-escalar a tamaños demasiado pequeños, pues se perdería el detalle, que es lo que interesa estudiar.

No hay forma de saber cual es el tamaño perfecto para una imagen. Además, al tratarse de imágenes capturadas con diferentes cámaras, tienen diferentes tamaños. Realizar pruebas de rendimiento en el modelo para diferentes tamaños será importante para encontrar el re-escalado correcto. Normalmente se hace un cambio de tamaño progresivo, de más pequeño a más grande (Nelson, 2020). En el caso de algunos modelos pre-entrenados, el tamaño de imagen es un parámetro estático. Para este proyecto, se ha establecido en (224, 224).

Para realizar esta tarea se ha utilizado la función *resize()* de *OpenCV*. Como argumentos se han introducido la imagen o el directorio donde encontrarla, el tamaño al que escalar y el tipo de interpolación que se utilizará, en este caso lineal.

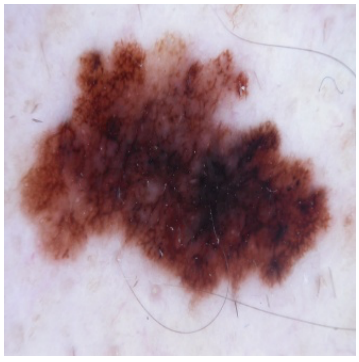


Figura 14: Imagen re-escalada.

Filtro de la mediana

El filtrado consiste en la supresión de ciertos aspectos en base a un criterio. Una buena metáfora es que el filtrado es como una red o una malla que al pasarla por los datos se queda con cierta parte del contenido.

En concreto, el filtro de la mediana se encarga de eliminar ruido preservando los bordes. Como se puede observar en la Figura 15, es muy útil para eliminar vellos y pequeñas impurezas que pueden confundir en la clasificación. Funciona de modo que cada entrada es sustituida por el valor de la mediana de las entradas adyacentes. A este conjunto de valores utilizados en cada sustitución se le conoce como ventana.

La función utilizada es *medianBlur()* de *OpenCV*. Esta función recibe la imagen re-escalada en el paso previo y un valor de tamaño para la ventana. A mayor tamaño, mayor número de píxeles en el cálculo y, con ello, más brusco será el filtrado.

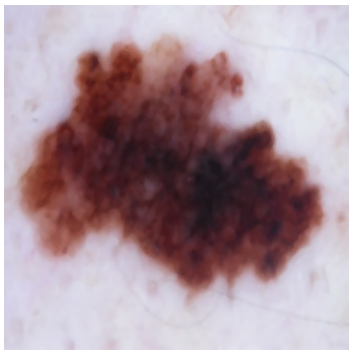


Figura 15: Imagen filtrada por la mediana.

Escala de grises

Esta operación supone el cambio de la escala color estándar, RGB, a escala de grises (Figura 16). Esto significa reducir la cantidad de canales de información de 3 (uno por color) a un único canal de 0 a 255. El objetivo de este paso mejorar la detección del contraste (que es más medible en esta escala), de modo que se pueda binarizar posteriormente.

Se ha utilizado la función *cvtColor()* de *OpenCV*. Esta recibe la imagen filtrada anteriormente y el tipo de cambio de escala, en este caso COLOR_BGR2GRAY, y cambia del espacio color RGB al espacio de grises.

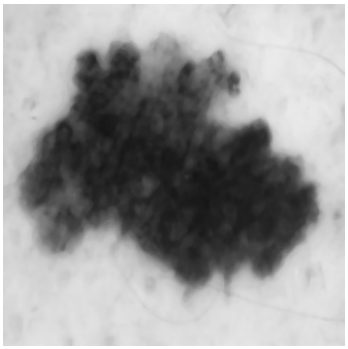


Figura 16: Imagen en escala de grises.

Corrección de iluminación Gamma

Este proceso consiste en resaltar ciertas zonas de la imagen. Se trata de potenciar el rango dinámico de modo que las zonas oscuras y las zonas claras resalten más. Este proceso pretende, al igual que el caso anterior, mejorar la imagen de cara a la posterior binarización.

La corrección Gamma es una operación de codificación y decodificación de luminancia en imágenes. Es un tipo de transformada de ley potencial, basada en la siguiente ecuación: $O = I\hat{}(\frac{1}{G})$. Donde *I* es la imagen de entrada escalada al rango (0,1), *O* la imagen de salida y G el escalar gamma. Un valor de G < 1 oscurece la imagen, un valor G > 1 la aclara, y un valor G = 1 mantiene la imagen original (Rosebrock, 2015).

Para el desarrollo de esta operación se ha generado una función específica que recibe como parámetros la imagen y el valor gamma. Esta función genera una tabla que mapea cada posible valor de píxel a su valor corregido con gamma correspondiente. Para obtener la imagen final, se utiliza la función *LUT()* de *OpenCV*. Esta función recibe dos argumentos, una matriz de entrada (imagen original) y una tabla de búsqueda. Los valores de la matriz actúan como índices en la tabla de búsqueda, los valores obtenidos se utilizan para generar una matriz de salida (Figura 17).

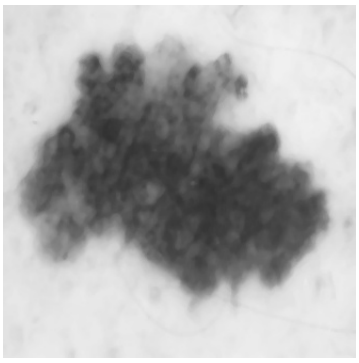


Figura 17: Imagen con corrección gamma.

Binarización

Este proceso consiste en transformar el valor de cada píxel del espectro de grises a un espectro de 0 (Negro) a 1 (Blanco). Con esta operación se resalta una zona de interés, en este caso se separará el lunar o posible lesión del fondo de la imagen.

Se ha utilizado la función *threshold()* de *OpenCV*. Esta función mapea cualquier foto en, que ha de estar en escala de grises, según cierto umbral. Como argumentos se introducen la imagen origen, el valor del umbral, el valor máximo que se asignará a los

píxeles que superen el umbral y el tipo de umbralización que ofrece la librería. En este caso, se utilizará la binarización inversa, es decir, los píxeles más oscuros (lunar) se devolverán blancos, mientras que los más claros (fondo de la imagen) se devolverán negros.

A este tipo de umbralización se le ha añadido el método *Otsu*. Esta operación evita tener que elegir el valor de umbral, ya que se determina automáticamente (OpenCV, 2020). El método consigue el valor óptimo obteniendo el valor medio que proporciona el histograma de la imagen. Es por ello que como cuarto argumento de la función aparecerá *cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU*.

La función *threshold()* devuelve dos salidas. La primera es el umbral que se ha utilizado y la segunda es la imagen binarizada, un ejemplo se muestra con la Figura 18.

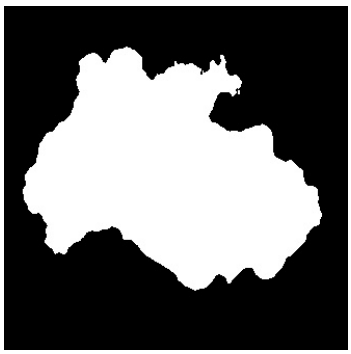


Figura 18: Imagen binarizada.

Bordes binarizados

Ya que los borde es una de las variables a tener en cuenta para la detección de la enfermedad, seré interesante remarcar el borde de la mancha o lunar como se muestra en la Figura 19. Para conseguirlo se utilizará la función *Canny()* que ofrece *OpenCV*.

Se trata de una función que engloba diferentes procesos como reducción de ruido, encontrar el gradiente de intensidad de la imagen, supresión de todos los píxeles que no supongan un máximo en el gradiente obtenido y la binarización por histéresis, que consiste en un proceso de selección de aristas en función a un umbral mínimo y máximo, según el valor del píxel y su conectividad (OpenCV, 2021, agosto 13).

Como argumentos se introducen la imagen origen, en este caso la obtenida en el apartado anterior, y un valor mínimo y máximo, que se utilizarán para la histéresis.

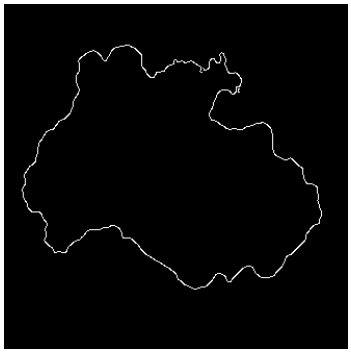


Figura 19: Imagen con los bordes binarizados.

Segmentación

Esta etapa finaliza el proceso de pre-procesado. Consta de dos operaciones, la superposición de color y la adición del borde. El resultado final será el mostrado en la Figura 20, dónde se puede observar el lunar perfectamente diferenciado del fondo y sus bordes remarcados.

La superposición de color consiste en utilizar la imagen binarizada y añadir en la zona blanca el color original del lunar, ya que es un factor importante que analizar. Esto se consigue con la función *bitwise_and()* de *OpenCV*, que actúa como una conjunción o función AND a nivel de bits. Como argumentos se introducen los dos *arrays* que se corresponden con las imágenes a comparar, que en este caso serán la misma imagen. Esto sucede porque se introducirá un tercer parámetro opcional que actúa como máscara, que será la imagen binarizada. Esta función devolverá la imagen con el fondo negro y la zona afectada coloreada.

Para añadir el borde, se utilizará la función *add()* de *OpenCV*, que realiza la suma de dos *arrays*. Los argumentos utilizados será el array obtenido con la superposición de color y el que se obtiene de la función *bitwise_not()*, que se utiliza para realizar una conversión de tipo para la imagen con el borde binarizado (OpenCV, 2021, Abril 2).

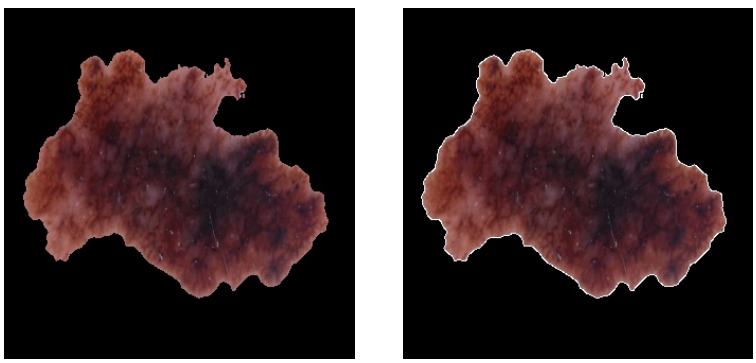


Figura 20: Segmentación de la imagen. (Izq: Superposición de color. Dcha: Adición de bordes.)

4.4. Modelado

En este apartado se va a describir el proceso de desarrollo del sistema general y como se han escogido e integrado diferentes modelos de *transfer learning* proporcionados por la librería *Keras* (Keras, 2021). Como se ha explicado anteriormente, este tipo de técnica requiere de cierto control sobre el conocimiento que y cómo se va a transferir.

Para este proyecto interesa que la transferencia se realice sobre la tarea origen, es decir, el aprendizaje de extracción de características, con un *dataset* más amplio que el obtenido para la tarea objetivo.

Esto se consigue congelando las capas superiores del modelo pre-entrenado (encargadas de la tarea de clasificación), de modo que no se pierda la información que éstas han obtenido. Se añaden capas nuevas que aprenderán a convertir la extracción de características obtenida en predicciones sobre el nuevo conjunto de datos. Además, se puede re-entrenar el modelo con *fine-tunning*, lo que permite adaptar las características pre-entrenadas al *dataset* específico de manera incremental.

4.4.1. Elección de los modelos pre-entrenados

Entornos como *TensorFlow* y *Pytorch* ofrecen un amplio catálogo de modelos pre-entrenados con la base de datos de *ImageNet*, el conjunto más utilizado para la clasificación de imágenes con este tipo de técnica. Escoger qué modelo utilizar es una tarea complicada.

Como se trata de un proyecto basado en el estudio de los datos, se ha utilizado el planteamiento que propone Leo (2020) en el que esta elección está completamente basada en los datos y dominio objetivo. Se buscarán los mejores modelos pre-entrenados de la *TensorFlow Keras API*, que será el entorno sobre el que se desarrollará el modelo general, para el *dataset* específico generado. El criterio estará basado en el valor de precisión que se obtenga. Se puede acceder al código completo mediante el enlace que se proporciona en Escoger modelo pre-entrenado dentro del Anexo II. Código del proyecto.

En primer lugar, se importan las librerías necesarias, se establece el tamaño del lote, en este caso 32, y se divide el *dataset* en un conjunto de entrenamiento y otro de validación. Se almacenan los modelos pre-entrenados en un objeto *list* de Python, listando todas las funciones dentro de *tf.keras.applications*, que es la librería que contiene estos modelos.

Para obtener las medidas a analizar, se utiliza un bucle que recorre todo el conjunto de modelos listado anteriormente. En cada iteración, se construye un modelo vacío al que le añadimos el modelo pre-entrenado descargado sin capas de salida y con los pesos congelados. Se le añade una capa densamente conectada, con función de activación *softmax* y se compila con función de pérdidas *categorical_crossentropy*. Por último, se realiza el entrenamiento del modelo construido para tres épocas y se guardan las medidas.

model_name	num_model_params	validation_accuracy
MobileNetV3Small	1529968	0.323232
EfficientNetB6	40960143	0.446970
EfficientNetB0	4049571	0.446970
EfficientNetB1	6575239	0.446970
EfficientNetB2	7768569	0.446970
EfficientNetB3	10783535	0.446970
EfficientNetB4	17673823	0.446970
EfficientNetB5	28513527	0.446970
EfficientNetB7	64097687	0.446970
MobileNetV3Large	4226432	0.459848
ResNet152	58370944	0.463889
ResNet101	42658176	0.465909
ResNet50	23587712	0.576768
VGG19	20024384	0.636869
VGG19	20024384	0.636869
VGG16	14714688	0.657576
InceptionV3	21802784	0.683081
NASNetMobile	4269716	0.704293
MobileNetV2	2257984	0.716414
DenseNet169	12642880	0.723485
InceptionResNetV2	54336736	0.730303
ResNet50V2	23564800	0.734343
DenseNet201	18321984	0.734343
ResNet152V2	58331648	0.735354
DenseNet121	7037504	0.735606
Xception	20861480	0.737374
ResNet101V2	42626560	0.737374
NASNetLarge	84916818	0.743182
MobileNet	3228864	0.746465

Tabla 4: Precisión del modelo y número de parámetros.

Para observar el resultado, se realizan dos visualizaciones. La primera (Tabla 4) se trata de una tabla en la que se muestran el nombre del modelo y su id, el número de parámetros y la precisión que consigue. El orden de la tabla es ascendente según la precisión obtenida, es decir, los últimos valores, son los que mejor resultado han presentado. La segunda (Figura 21) se trata de una gráfica de dispersión en la que las abscisas son el número de parámetros y las ordenadas la precisión obtenida para el *dataset* específico en la tarea de validación.

Observando las visualizaciones planteadas podemos considerar *MobileNet* como el modelo que más precisión presenta para los datos obtenidos. Y aunque encontramos hasta 12 modelos con un valor de precisión superior al 70%, solo *MobileNetV2* presenta un número inferior de parámetros.

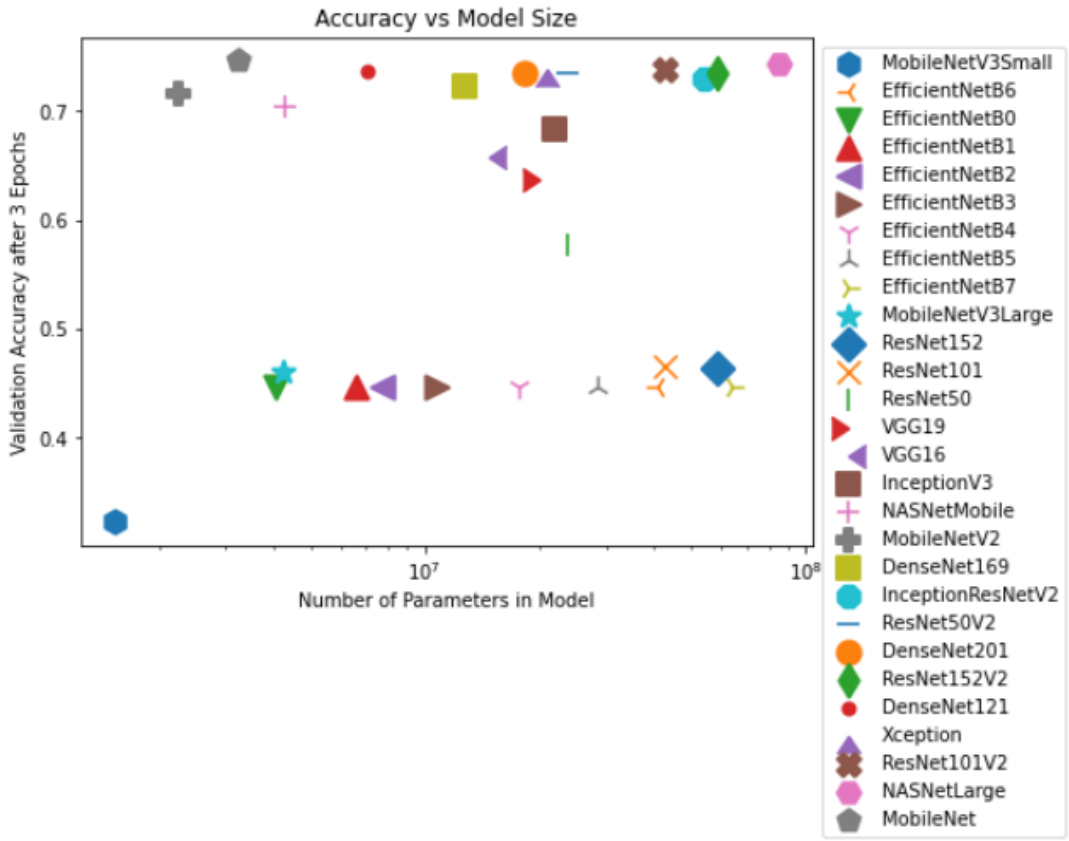


Figura 21: Dispersión de la precisión según el tamaño del modelo.

En este contexto, *MobileNet* se puede considerar el modelo pre-entrenado óptimo para realizar el aprendizaje de características en este proyecto, debido a que presenta una carga de trabajo pequeña y valores de precisión muy positivos. Pero se trata de una evaluación sobre 3 etapas, es decir, a corto plazo. Muchos de estos modelos van mejorando con el paso de las etapas sin conllevar sobreajuste.

Por esto, se ha decidido analizar los resultados para 3 de los modelos que se encuentran en esta franja superior a 0.7. Y la decisión se tomará para distintos números de parámetros. Para un número de parámetros pequeño se tomará *MobileNet*, para un número medio se comprobará *Xception* y para un número grande de parámetros se utilizará *ResNet101V2*, que se nombrará como *ResNet* de aquí en adelante.

4.4.2. Desarrollo del modelo general

Con los modelos de transferencia seleccionados, se comenzará la fase de desarrollo del sistema específico que tiene este proyecto como objetivo. Este proceso sigue pasos no muy diferentes a los mencionados en la sección anterior, pero que han de ser detallados.

El código referente a este apartado se encuentra dividido en tres *scripts*, uno por cada modelo pre-entrenado a analizar. Para acceder a ellos habrá que utilizar los enlaces proporcionados en el apartado Desarrollo de los modelos del Anexo II. Código del proyecto.

El desarrollo de un modelo de *machine learning* (Solawetz, 2020) se puede dividir en tres procesos principales, que son el entrenamiento, la validación y la evaluación. El entrenamiento y la validación, que se describen a continuación, se encargan de que el modelo aprenda, por ello, suelen ir de la mano. La evaluación se encarga de indicar lo preciso que es el algoritmo en un contexto más realista y se describe más detalladamente en el siguiente apartado.

Acceso a los datos

El sobreajuste ocurre cuando un modelo se encuentra ajustado a los datos de entrenamiento, de modo que no obtiene resultados positivos con datos externos a este conjunto (Gonzalo, 2020).

Para evitar esto es muy importante el proceso de evaluación. Y para el desarrollo óptimo de este, los datos han de separarse en subconjuntos dedicados a cada uno de los procesos nombrados anteriormente (Agrawal, 2021).

- Los **datos de entrenamiento** se utilizan para entrenar el modelo.
- Los **datos de validación** se encargan de ajustar pesos e hiperparámetros.
- Los **datos de evaluación** indican la precisión del modelo en un caso de uso.

Para realizar esta división en subconjuntos existen diferentes fórmulas. En este caso, aunque no se ha especificado, esta división se ha realizado durante la homogeneización. Se ha utilizado la función *sample()* de *Pandas*, que obtiene un conjunto aleatorio de elementos del *dataframe* que lo ha llamado. Precisa de un argumento que indique el número o porcentaje de elementos a obtener.

Para este modelo se ha utilizado un 70% de los datos para el entrenamiento, lo que supone cerca de 1000 imágenes por clase, y un 15% tanto para validación como para evaluación (Tabla 5), cerca de 200 imágenes por clase.

```
Num train images: 4132
Num validation images: 885
Num test images: 883
Num classes: 4
```

Tabla 5: Adquisición de los datos.

El primer paso que se debe realizar en el código previo a la construcción de un modelo es el acceso al conjunto de datos. Al tratarse de imágenes, se ha aprovechado la función *image_dataset_from_directory()* de *TensorFlow*. Esta función almacena en un *array* el directorio de cada imagen que conforma el *dataset*. La división los datos de cada clase en carpetas específicas es de gran importancia, ya que esta función aprovecha dicha organización para etiquetar cada imagen.

La función precisa del directorio principal de la base de datos como argumento. En este caso, se generarán tres variables, uno para cada proceso de los descritos al inicio del apartado. Es decir, se utilizarán los directorios de las tres carpetas que se generaron con *sample()*.

Como última acción dentro de esta sección, se ha realizado un procesado a nivel de formato para la correcta interpretación de los datos por parte del modelo. Este proceso ha consistido en generar un objeto iterable mediante la función *map()*, que aplica una función (primer argumento) a los elementos de un objeto (segundo argumento). En este caso, se han normalizado las imágenes y codificado las etiquetas al formato *one hot*. Este formato consiste en la utilización de una columna para cada clase diferente. La columna clase a la que pertenezca un registro estará marcada en uno, mientras que el resto de las columnas estarán en 0 para ese registro.

Este proceso se realizará para cada uno de los conjuntos de datos, de modo que se generan tres variables que serán las que se van a utilizar como entradas en entrenamiento, validación y evaluación del sistema.

Construcción del modelo

El proceso de construcción de la herramienta consiste en generar un modelo vacío (por cada modelo pre-entrenado) sobre el que se incluirán una primera capa de entrada, una capa que contendrá el modelo pre-entrenado con las capas de entrenamiento congeladas y las capas de salida que se consideren.

Antes de comenzar a construir el modelo, se descargará el modelo de transferencia en una variable, para ser manejada posteriormente. Estos modelos son ofrecidos en el

catálogo de modelos pre-entrenados de *Keras*. Es por esto y la facilidad que propone esta librería para el desarrollo de *Deep Learning* que el modelo tendrá su base en la API entrenable de *TensorFlow Keras*.

Para descargarlos se instanciará cada uno de los modelos, que se encuentran en *applications* dentro de la librería de *Keras*. Estos modelos precisan de una serie de parámetros que permiten personalizar la arquitectura. Para todos los casos, se pondrá el argumento *inclued_top* en *False*, que indica la no inclusión de capas superiores totalmente conectadas, y se introducirá el *input_shape*, que dependerá del tamaño de la imagen. El argumento *weights* que hace referencia a los pesos pre-entrenados, utiliza como valor predeterminado '*imagenet'*. Y, por último, como valor de *pooling* se introducirá '*avg*', que indica que se aplicará agrupación promedio global de la salida del último bloque convolucional, de modo que la salida de esta capa será un tensor 2D.

Lo descrito hasta ahora, genera una capa que, como el resto de los tipos de capa, contiene un atributo *trainable*, y este habrá que ponerlo en *False*. De este modo se marcan todos los pesos como no entrenables, es decir, se congela la capa. Lo que implica que la capa no se actualizará durante el entrenamiento.

Con el modelo pre-entrenado descargado y preparado, se puede comenzar a diseñar y completar el modelo vacío que se puede diferenciar en dos partes, la fase de extracción de características y la fase de clasificación (Figura 22). Para ello, se utilizará un modelo secuencial, mediante la función *Sequential()* de *Keras*. Un modelo secuencial es aquel en el que cada capa tiene un tensor de entrada y otro de salida (Chollet, 2020). La arquitectura de la red se definirá introduciendo cada una de las capas en una lista que será pasada como argumento a esta función.

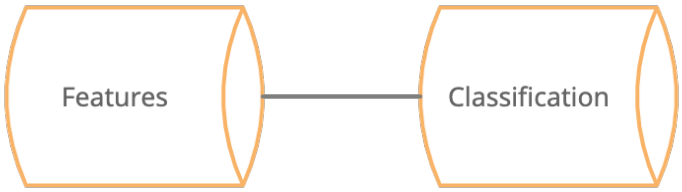


Figura 22: Fases del modelo.

La primera de las capas que se definen es del tipo *InputLayer()* y se trata del punto de entrada de la red. Sólo precisa del formato o *input_shape* como entrada de la función.

A continuación, se ha introducido una capa de aumento de datos mediante otro modelo secuencial, que acumula diferentes transformaciones. Esta técnica genera nuevas

entradas alterando datos existentes. Es muy usual utilizarla con *Transfer Learning*, ya que este tipo de aprendizaje suele ir acompañado de *datasets* limitados. La transformación que se ha utilizado es el volteado, el cambio de contraste y la traslación de la imagen de manera aleatoria. De este modo, se consigue una extracción de características más exhaustiva.

La tercera de las capas será el modelo pre-entrenado que se vaya a comprobar y que se descargó y preparó anteriormente. Esta será la última de las capas que se encargan de la extracción de características (Figura 23).

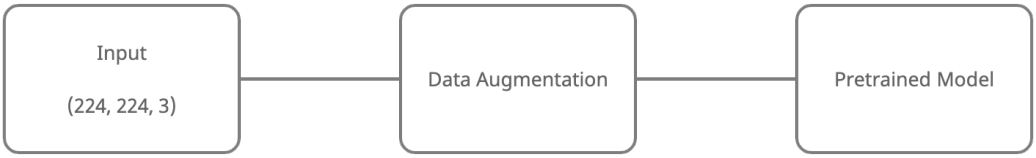


Figura 23: Fase de extracción de características.

La etapa de clasificación (Figura 24) comienza con una capa *Dense()*, es decir, totalmente conectada. Todas las salidas de la capa anterior serán introducidas en cada neurona o nodo. Este tipo de capas precisa como argumentos se introducirán el tamaño de salida, la función de activación y, si se quiere, una función regularizadora. En este caso se utilizan 64 nodos como salida, lo que implica que las características extraídas por el modelo pre-entrenado serán reducidas a 64 valores. Y para esta reducción se utiliza la función de activación ReLu y una función regularizadora que será aplicada sobre la matriz de pesos.

A continuación, se ha utilizado *Dropout()*. Esta capa elimina ciertos nodos de la red, en cada fase del entrenamiento, con una probabilidad de *1 – p*, donde *p* se introduce como argumento a la función. Esta capa ayuda a evitar el sobreajuste eliminando la codependencia entre neuronas que se genera durante el entrenamiento (Budhiraja, 2016).

Figura 24: Fase de clasificación.

Como última capa destinada a la clasificación, se incluirá una capa totalmente conectada, *Dense()*, que tendrá 4 nodos de salida, uno por cada clase. Utiliza la función de activación *softmax*, una generalización de la función logística, cuya salida es una distribución categórica sobre K diferentes salidas, en este caso K es el número de clases. En resumen, esta última capa aportará un valor de probabilidad por cada clase, sumando entre todos los valores 1.

Tras completar la definición y el diseño del modelo, habrá que **construirlo**. Esto se consigue mediante la función *build()* y se encarga de inicializar los pesos de las capas, por ello, solo se llama una vez.

En la Tabla 6, se muestra un resumen del modelo para *MobileNet* y sus capas. Es el resultado de utilizar la función *summary()*. Se pueden observar las capas, su orden, el formato de salida, el número de parámetros por capa, el número de parámetros totales y diferenciar los parámetros entrenables de aquellos no entrenables.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_52 (Sequential)   (None, 224, 224, 3)       0
_________________________________________________________________
mobilenet_1.00_224 (Function (None, 1024)              3228864
_________________________________________________________________
dense_40 (Dense)             (None, 64)                65600
_________________________________________________________________
dropout_20 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_41 (Dense)             (None, 4)                 260
=================================================================
Total params: 3,294,724
Trainable params: 65,860
Non-trainable params: 3,228,864
_________________________________________________________________
```

Tabla 6: Arquitectura del modelo con MobileNet.

El último paso antes de comenzar con la etapa de entrenamiento es la **compilación** del modelo. Mediante este proceso se definen la función de pérdida, optimizador y métricas. La función de pérdida evalúa lo bien que el algoritmo modela los datos y, con la ayuda del optimizador, aprende a reducir el error (Parmar, 2018). En definitiva, son argumentos imprescindibles para que se pueda desarrollar el entrenamiento.

Para conseguir la compilación del modelo habrá que llamar a la función *compile()* que contiene cualquier modelo de *Keras* con los tres argumentos descritos. En este caso, el optimizador utilizado ha sido Adam, la función de pérdidas ha sido la entropía cruzada y cómo métrica '*accuracy*', que calcula la frecuencia con la que las predicciones son iguales a las etiquetas.

Adam es un optimizador basado en el descenso de gradiente estocástico que sólo precisa de gradientes de primer orden y tiene pocos requisitos de memoria (Kingma, 2017). Esta función se utiliza para actualizar los pesos basándose en los datos de entrenamiento y se puede encontrar en el catálogo de optimizadores de *Keras* llamando a la función *Adam()*. Necesita de un valor de aprendizaje cuyo valor predeterminado es 0.001.

La entropía cruzada es muy común en problemas de clasificación de más de dos clases. Esta técnica compara la clase predicha con la clase deseada y calcula la pérdida en función de lo lejos que esté el valor esperado (Koech, 2020). La función utilizada es *CategoricalCrossentropy()*, proporcionada por *Keras*.

Entrenamiento y validación

Una vez compilado el modelo, se podrá comenzar con el entrenamiento y la validación del modelo, son dos etapas que se realizan a la vez. El entrenamiento consiste en la construcción de un sistema clasificador a partir de iterar sobre el modelo el conjunto de datos separado para este proceso. La validación se va realizando al final de cada iteración del entrenamiento y se encarga de estabilizar los pesos y evitar que ocurra sobreajuste con los datos de entrenamiento, por ello que se utiliza un conjunto específico diferente.

En esta fase, por tanto, se introducirán dos de los conjuntos de datos que se separaron y prepararon en etapas anteriores. El desarrollo de este proceso está diferenciado en épocas, que son pasadas por el conjunto de datos completo, es decir, iteraciones. Para la selección del número de épocas es indispensable un conocimiento profundo tanto de la situación objetivo como del conjunto de datos utilizado. Normalmente, para conjuntos de datos grandes, las épocas se separan en pasos que utilizan pequeñas subdivisiones de los datos, llamados lotes.

Realizar este proceso es muy simple gracias a la función *fit()* que contienen los modelos en *Keras*. Esta función recibe como argumentos el conjunto de entrenamiento, las épocas, los pasos por época y los datos de validación. En este caso, el número de épocas ha sido 10, el tamaño del lote 32 imágenes y el número de pasos por época es el tamaño del conjunto de entrenamiento entre el tamaño del lote, es decir, $4132/32 = 129$ pasos.

Además, se ha añadido el *early stopping* como función de *callback*. Los callback o funciones de retorno ayudan a tener control del entrenamiento mientras ejecuta. *Early stopping* realiza una parada en el entrenamiento cuando comienza a detectarse

sobreajuste. *Keras* proporciona una función *EarlyStopping()* que requiere como argumentos la métrica que se va a monitorear y un número de épocas base que sobre los que no se aplicará la parada. Este modelo parará su ejecución en el momento que se sucedan 3 épocas sin mejoras en la precisión de la etapa de validación.

Resultados del entrenamiento

En el presente subapartado se recogen los resultados obtenidos al llamar a la función *fit()* para cada uno de los modelos. Los resultados se muestran mediante tres visualizaciones. Las dos primeras son una gráfica de líneas que mostrará el progreso por épocas de precisión y pérdida tanto en la tarea de entrenamiento como en la de validación. La segunda es una visualización equivalente, pero en forma de tabla, de modo que se puedan observar y analizar los valores exactos de progreso para cada tarea.

***MobileNet*:** Como muestra la Tabla 7, los valores de precisión y pérdida obtenidos para este modelo no son tan positivos como se esperaban tras la fase de elección de modelos.

El valor máximo de precisión en la etapa de validación es del 68%, mientras que en la fase de entrenamiento llega hasta 71%. Además de no ser valores muy altos, esta diferencia entre las dos fases indica, aunque mínimo, cierto grado de sobreajuste.

En la Figura 25 y la Figura 26 se puede observar que la progresión de la fase de entrenamiento mantiene una proyección interesante (positiva para la precisión y negativa para la pérdida) mientras que la de la fase de validación, aunque mejora, se mantiene más estática.

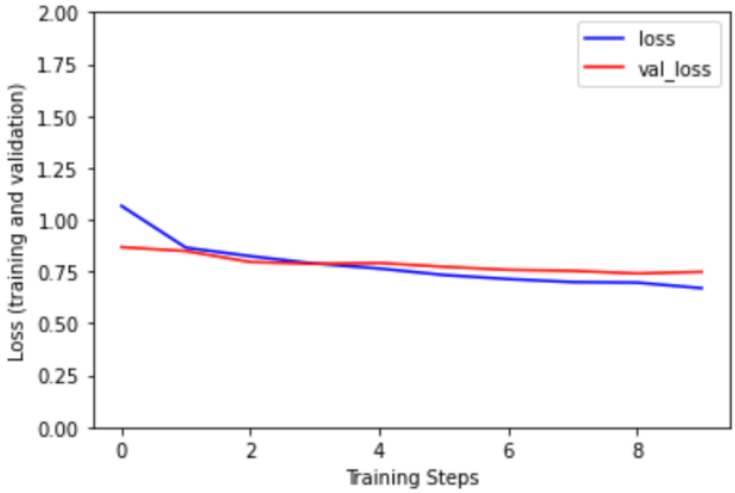


Figura 25: Evolución de la pérdida (MobileNet).

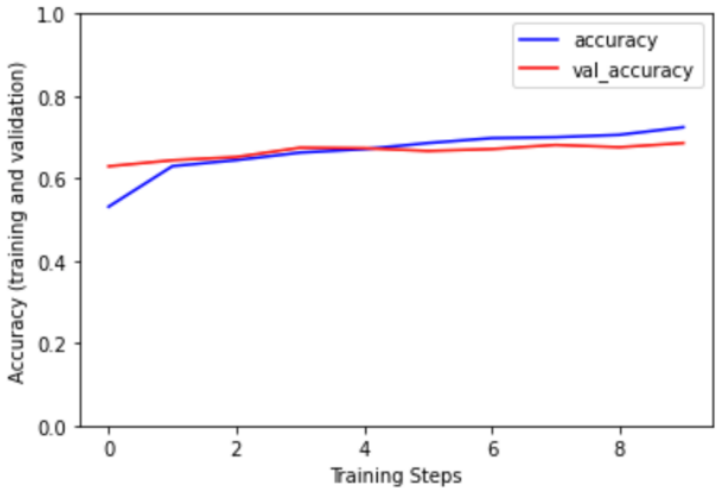


Figura 26: Evolución de la precisión (MobileNet).

```
Epoch 1/10
129/129 [==] - loss: 1.0662 - accuracy: 0.5312 - val_loss: 0.8672 - val_accuracy: 0.6294
Epoch 2/10
129/129 [==] - loss: 0.8647 - accuracy: 0.6298 - val_loss: 0.8483 - val_accuracy: 0.6441
Epoch 3/10
129/129 [==] - loss: 0.8239 - accuracy: 0.6444 - val_loss: 0.7959 - val_accuracy: 0.6520
Epoch 4/10
129/129 [==] - loss: 0.7881 - accuracy: 0.6627 - val_loss: 0.7870 - val_accuracy: 0.6746
Epoch 5/10
129/129 [==] - loss: 0.7642 - accuracy: 0.6710 - val_loss: 0.7912 - val_accuracy: 0.6734
Epoch 6/10
129/129 [==] - loss: 0.7332 - accuracy: 0.6859 - val_loss: 0.7728 - val_accuracy: 0.6667
Epoch 7/10
129/129 [==] - loss: 0.7136 - accuracy: 0.6978 - val_loss: 0.7580 - val_accuracy: 0.6712
Epoch 8/10
129/129 [==] - loss: 0.6990 - accuracy: 0.7000 - val_loss: 0.7531 - val_accuracy: 0.6814
Epoch 9/10
129/129 [==] - loss: 0.6964 - accuracy: 0.7059 - val_loss: 0.7402 - val_accuracy: 0.6757
Epoch 10/10
129/129 [==] - loss: 0.6697 - accuracy: 0.7241 - val_loss: 0.7482 - val_accuracy: 0.6859
```

Tabla 7: Etapas de entrenamiento (MobileNet).

***Xception*:** Para este modelo tampoco se consiguen valores de precisión muy optimistas. El valor máximo de precisión en la fase de validación alcanza el 65% y en la fase de entrenamiento se llega al 70% (Tabla 8).

Es decir, el sobreajuste es levemente mayor que en el caso anterior y los resultados son peores, pero también se observa un crecimiento más progresivo, como bien muestra las gráficas de la Figura 27 y Figura 28, lo que puede implicar mejores resultados a la larga.

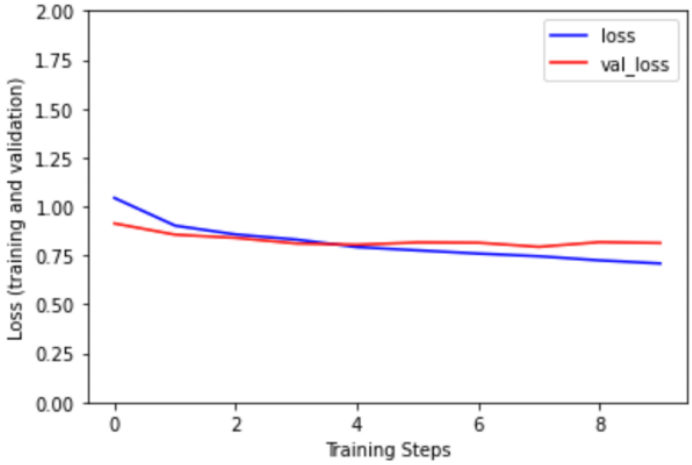


Figura 27: Evolución de la pérdida (Xception).

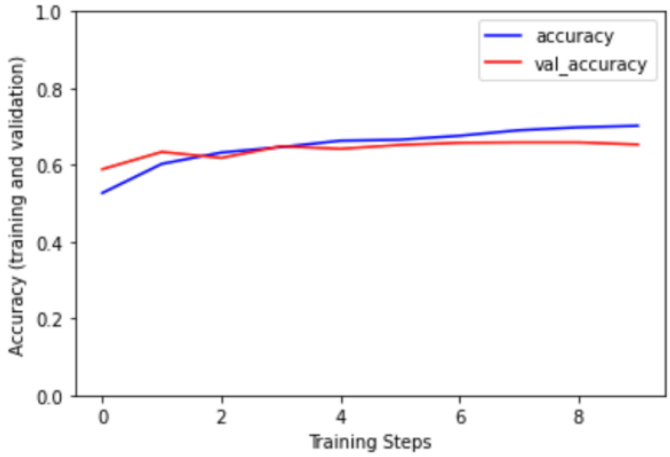


Figura 28: Evolución de la precisión (Xception).

```
Epoch 1/10
129/129 [==- loss: 1.0436 - accuracy: 0.5266 - val_loss: 0.9134 - val_accuracy: 0.5887
Epoch 2/10
129/129 [== - loss: 0.9021 - accuracy: 0.6027 - val_loss: 0.8560 - val_accuracy: 0.6339
Epoch 3/10
129/129 [== - loss: 0.8571 - accuracy: 0.6322 - val_loss: 0.8396 - val_accuracy: 0.6181
Epoch 4/10
129/129 [== - loss: 0.8309 - accuracy: 0.6463 - val_loss: 0.8110 - val_accuracy: 0.6486
Epoch 5/10
129/129 [== - loss: 0.7925 - accuracy: 0.6629 - val_loss: 0.8045 - val_accuracy: 0.6418
Epoch 6/10
129/129 [== - loss: 0.7758 - accuracy: 0.6656 - val_loss: 0.8164 - val_accuracy: 0.6520
Epoch 7/10
129/129 [== - loss: 0.7594 - accuracy: 0.6759 - val_loss: 0.8148 - val_accuracy: 0.6576
Epoch 8/10
129/129 [== - loss: 0.7458 - accuracy: 0.6902 - val_loss: 0.7939 - val_accuracy: 0.6588
Epoch 9/10
129/129 [== - loss: 0.7246 - accuracy: 0.6978 - val_loss: 0.8178 - val_accuracy: 0.6588
Epoch 10/10
129/129 [== - loss: 0.7089 - accuracy: 0.7020 - val_loss: 0.8134 - val_accuracy: 0.6531
```

Tabla 8: Etapas de entrenamiento (Xception).

***ResNet*:** En este caso se observan valores de pérdida enormes y de precisión demasiado bajos (Tabla 9). Además, la progresión no mantiene consistencia, está repleta de saltos, como muestran la Figura 29 y la Figura 30.

Un factor curioso es que el sobreajuste es bastante pronunciado en casi toda la evaluación, a excepción de la última de las etapas, que se podría considerar buena en un contexto tan malo como el que plantea este modelo. Pero también es el primero de los modelos que ha sido parado por el *Early Stopping* a las 4 épocas. El modelo podría mejorar a largo plazo.

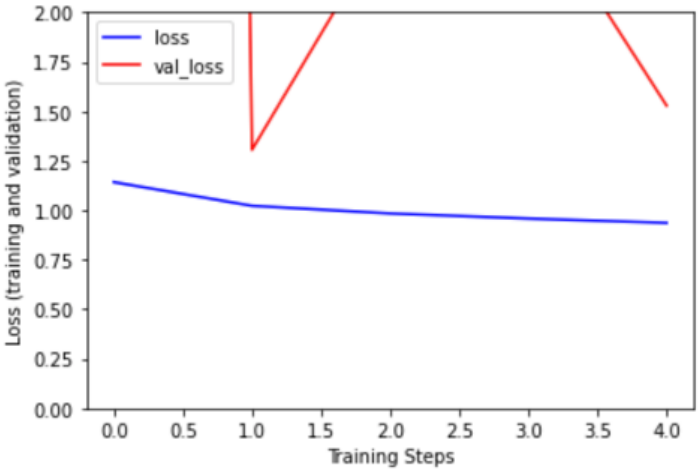


Figura 29: Evolución de la pérdida (ResNet).

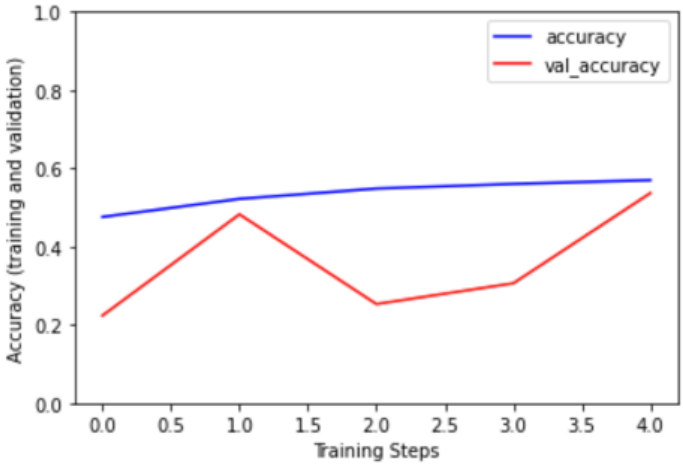


Figura 30: Evolución de la precisión (ResNet).

```
Epoch 1/10
129/129 [==loss: 1.1422 - accuracy: 0.4758 - val_loss: 46.7550 - val_accuracy: 0.2237
Epoch 2/10
129/129 [==- loss: 1.0226 - accuracy: 0.5220 - val_loss: 1.3055 - val_accuracy: 0.4825
Epoch 3/10
129/129 [==- loss: 0.9841 - accuracy: 0.5483 - val_loss: 2.4790 - val_accuracy: 0.2531
Epoch 4/10
129/129 [==- loss: 0.9583 - accuracy: 0.5602 - val_loss: 2.6178 - val_accuracy: 0.3062
Epoch 5/10
129/129 [==- loss: 0.9366 - accuracy: 0.5698 - val_loss: 1.5287 - val_accuracy: 0.5367
```

Tabla 9: Etapas de entrenamiento (ResNet).

Fine Tunning

Un paso opcional que puede mejorar incrementalmente el resultado del entrenamiento es el *fine-tuning*. Este proceso consiste en, tras alcanzar la convergencia en el entrenamiento, descongelar las capas del modelo pre-entrenado y recompilar el conjunto de capas diseñado. Una vez construido este nuevo modelo descongelado, reentrenarlo.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_56 (Sequential)   (None, 224, 224, 3)       0
_________________________________________________________________
mobilenet_1.00_224 (Function (None, 1024)              3228864
_________________________________________________________________
dense_44 (Dense)             (None, 64)                65600
_________________________________________________________________
dropout_22 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_45 (Dense)             (None, 4)                 260
=================================================================
Total params: 3,294,724
Trainable params: 3,272,836
Non-trainable params: 21,888
_________________________________________________________________
```

Tabla 10: Arquitectura del modelo con MobileNet tras el Fine Tunning.

Se puede observar como se altera el número de parámetros entrenables en el resumen mostrado en la Tabla 10 tras hacer *summary()* sobre el modelo de *MobileNet*.

Es importante que el modelo alcance la convergencia antes de realizar esta acción, para evitar que se mezclen capas y se destruyan características, por ello se introduce el *EarlyStopping*. Otra faceta a tener en cuenta es la utilización de una tasa de aprendizaje muy baja en el optimizador, menor que la utilizada en el paso anterior. Para todos los casos, se han realizado 15 épocas con *EarlyStopping* y una tasa de aprendizaje de 0.00001.

Resultados tras el *Fine Tunning*

Este subapartado es equivalente al de la primera etapa de entrenamiento. En este caso se recogerán los resultados al finalizar la etapa de *Fine Tunning*.

***MobileNet*:** Tras la aplicación del *Fine Tunning,* con *MobileNet* se observa una mejora más progresiva en la etapa de validación (Tabla 11), llegando a un valor de precisión de 71%, sin subidas y bajadas, como ocurría en el paso anterior.

Aún con la mejora, la etapa de entrenamiento sigue siendo más productiva y consigue una mejora más prominente, lo que indica que el modelo está sobreajustando, ya que la diferencia entre ambas fases se ha incrementado (Figura 31 y Figura 32**¡Error! No se encuentra el origen de la referencia.**).

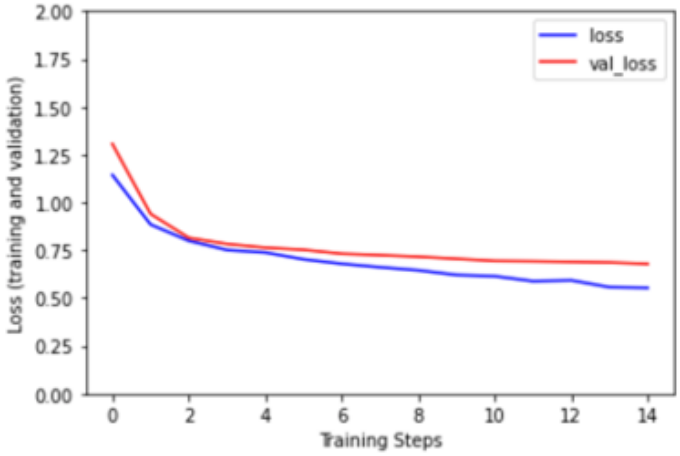


Figura 31: Evolución de la pérdida tras el Fine Tunning (MobileNet).

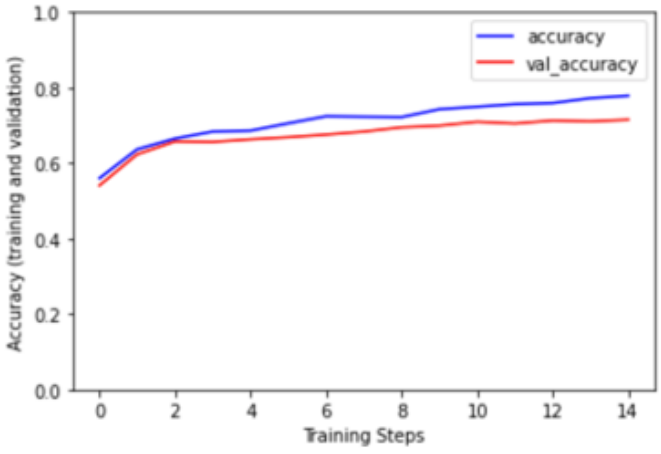


Figura 32: Evolución de la precisión tras el Fine Tunning (MobileNet).

```
Epoch 1/15
129/129 [==- loss: 1.1439 - accuracy: 0.5603 - val_loss: 1.3059 - val_accuracy: 0.5412
Epoch 2/15
129/129 [==- loss: 0.8848 - accuracy: 0.6371 - val_loss: 0.9403 - val_accuracy: 0.6237
Epoch 3/15
129/129 [==- loss: 0.7991 - accuracy: 0.6654 - val_loss: 0.8148 - val_accuracy: 0.6576
Epoch 4/15
129/129 [==- loss: 0.7508 - accuracy: 0.6837 - val_loss: 0.7823 - val_accuracy: 0.6565
Epoch 5/15
129/129 [==- loss: 0.7380 - accuracy: 0.6861 - val_loss: 0.7633 - val_accuracy: 0.6633
Epoch 6/15
129/129 [==- loss: 0.7027 - accuracy: 0.7056 - val_loss: 0.7521 - val_accuracy: 0.6689
Epoch 7/15
129/129 [==- loss: 0.6795 - accuracy: 0.7244 - val_loss: 0.7316 - val_accuracy: 0.6757
Epoch 8/15
129/129 [==- loss: 0.6614 - accuracy: 0.7232 - val_loss: 0.7243 - val_accuracy: 0.6836
Epoch 9/15
129/129 [==- loss: 0.6455 - accuracy: 0.7222 - val_loss: 0.7162 - val_accuracy: 0.6949
Epoch 10/15
129/129 [==- loss: 0.6210 - accuracy: 0.7432 - val_loss: 0.7054 - val_accuracy: 0.6994
Epoch 11/15
129/129 [==- loss: 0.6143 - accuracy: 0.7498 - val_loss: 0.6946 - val_accuracy: 0.7096
Epoch 12/15
129/129 [==- loss: 0.5885 - accuracy: 0.7566 - val_loss: 0.6925 - val_accuracy: 0.7051
Epoch 13/15
129/129 [==- loss: 0.5928 - accuracy: 0.7595 - val_loss: 0.6894 - val_accuracy: 0.7130
Epoch 14/15
129/129 [==- loss: 0.5568 - accuracy: 0.7722 - val_loss: 0.6865 - val_accuracy: 0.7107
Epoch 15/15
129/129 [==- loss: 0.5532 - accuracy: 0.7785 - val_loss: 0.6788 - val_accuracy: 0.7153
```

Tabla 11: Etapas de entrenamiento tras el Fine Tunning (MobileNet).

Xception: La aplicación del *Fine Tunning* ha supuesto una mejora considerable en la precisión del modelo *Xception*, superando los resultados aportados por el modelo anterior y alcanzando un valor de precisión para la fase de validación de un 74% (Tabla 12), un nivel que se puede comenzar a considerar optimista.

Con el mismo carácter de crecimiento progresivo que se observó en el paso anterior, en las gráficas de la Figura 33 y la Figura 34, se muestra el aumento del sobreajuste, aunque de una manera menos prominente que en la fase anterior.

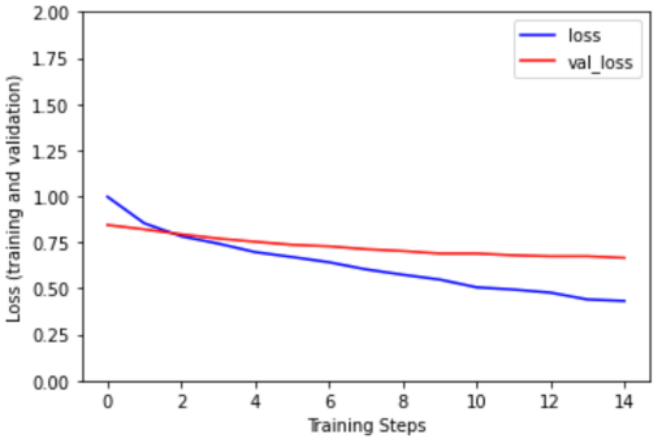


Figura 33: Evolución de la pérdida tras el Fine Tunning (Xception).

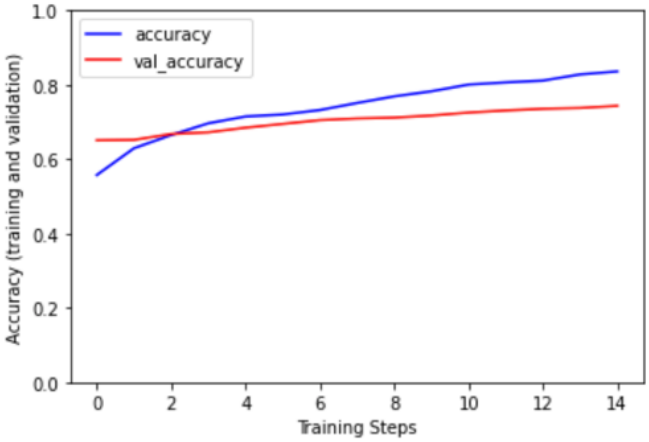


Figura 34: Evolución de la precisión tras el Fine Tunning (Xception).

```
Epoch 1/15
129/129 [==- loss: 0.9972 - accuracy: 0.5574 - val_loss: 0.8441 - val_accuracy: 0.6508
Epoch 2/15
129/129 [==- loss: 0.8531 - accuracy: 0.6295 - val_loss: 0.8202 - val_accuracy: 0.6520
Epoch 3/15
129/129 [==- loss: 0.7827 - accuracy: 0.6649 - val_loss: 0.7937 - val_accuracy: 0.6678
Epoch 4/15
129/129 [==- loss: 0.7432 - accuracy: 0.6966 - val_loss: 0.7706 - val_accuracy: 0.6723
Epoch 5/15
129/129 [==- loss: 0.6969 - accuracy: 0.7149 - val_loss: 0.7532 - val_accuracy: 0.6847
Epoch 6/15
129/129 [==- loss: 0.6705 - accuracy: 0.7202 - val_loss: 0.7365 - val_accuracy: 0.6949
Epoch 7/15
129/129 [==- loss: 0.6420 - accuracy: 0.7322 - val_loss: 0.7283 - val_accuracy: 0.7051
Epoch 8/15
129/129 [==- loss: 0.6033 - accuracy: 0.7512 - val_loss: 0.7129 - val_accuracy: 0.7096
Epoch 9/15
129/129 [==- loss: 0.5744 - accuracy: 0.7690 - val_loss: 0.7027 - val_accuracy: 0.7119
Epoch 10/15
129/129 [==- loss: 0.5481 - accuracy: 0.7827 - val_loss: 0.6893 - val_accuracy: 0.7175
Epoch 11/15
129/129 [==- loss: 0.5059 - accuracy: 0.8007 - val_loss: 0.6893 - val_accuracy: 0.7254
Epoch 12/15
129/129 [==- loss: 0.4938 - accuracy: 0.8066 - val_loss: 0.6796 - val_accuracy: 0.7311
Epoch 13/15
129/129 [==- loss: 0.4771 - accuracy: 0.8115 - val_loss: 0.6746 - val_accuracy: 0.7356
Epoch 14/15
129/129 [==- loss: 0.4401 - accuracy: 0.8280 - val_loss: 0.6747 - val_accuracy: 0.7379
Epoch 15/15
129/129 [==- loss: 0.4321 - accuracy: 0.8361 - val_loss: 0.6656 - val_accuracy: 0.7435
```

Tabla 12: Etapas de entrenamiento tras el Fine Tunning (Xception).

***ResNet*:** En este caso la mejora de precisión y pérdida tras el *Fine Tunning* es bastante considerable teniendo en cuenta los resultados que se habían obtenido en a fase anterior. Pero a pesar de mejorar los registros en un contexto aislado, se obtiene una precisión máxima de 69% (Tabla 13 y Tabla 14), un valor que queda por debajo de los modelos anteriores.

Aunque los resultados no son los mejores, el nivel de sobreajuste es un factor que destacar para este modelo, ya que es inexistente durante este número de épocas. Si se observan la Figura 35 y la Figura 36, las gráficas presentan un alto nivel de convergencia y una progresión para entrenamiento y validación muy similar.

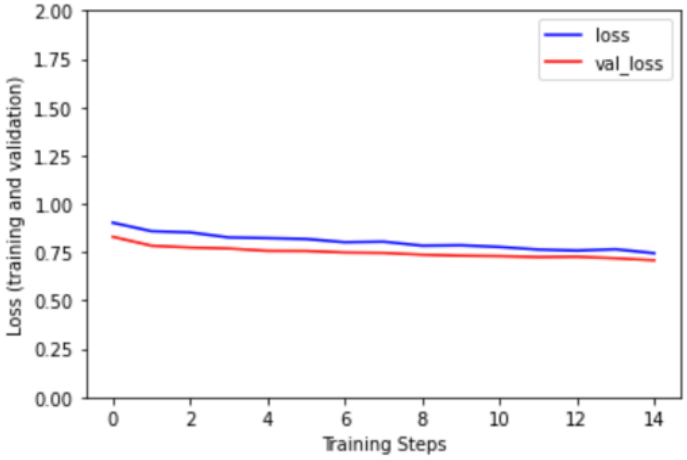


Figura 35: Evolución de la pérdida tras el Fine Tunning (ResNet).

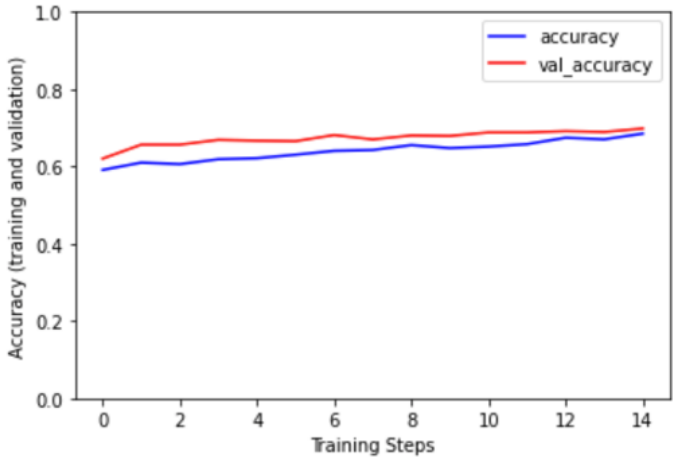


Figura 36: Evolución de la precisión tras el Fine Tunning (ResNet).

```
Epoch 1/15
129/129 [==- loss: 0.9025 - accuracy: 0.5911 - val_loss: 0.8295 - val_accuracy: 0.6203
Epoch 2/15
129/129 [==- loss: 0.8585 - accuracy: 0.6100 - val_loss: 0.7835 - val_accuracy: 0.6565
Epoch 3/15
129/129 [==- loss: 0.8522 - accuracy: 0.6061 - val_loss: 0.7741 - val_accuracy: 0.6565
Epoch 4/15
129/129 [==- loss: 0.8262 - accuracy: 0.6188 - val_loss: 0.7690 - val_accuracy: 0.6689
Epoch 5/15
129/129 [==- loss: 0.8226 - accuracy: 0.6215 - val_loss: 0.7569 - val_accuracy: 0.6667
Epoch 6/15
129/129 [==- loss: 0.8180 - accuracy: 0.6305 - val_loss: 0.7563 - val_accuracy: 0.6655
Epoch 7/15
129/129 [==- loss: 0.8014 - accuracy: 0.6405 - val_loss: 0.7483 - val_accuracy: 0.6814
```

Tabla 13: Etapas de entrenamiento tras el Fine Tunning Pt 1 (ResNet).

```
Epoch 8/15
129/129 [==- loss: 0.8043 - accuracy: 0.6427 - val_loss: 0.7461 - val_accuracy: 0.6701
Epoch 9/15
129/129 [==- loss: 0.7837 - accuracy: 0.6554 - val_loss: 0.7369 - val_accuracy: 0.6802
Epoch 10/15
129/129 [==- loss: 0.7858 - accuracy: 0.6476 - val_loss: 0.7316 - val_accuracy: 0.6791
Epoch 11/15
129/129 [==- loss: 0.7770 - accuracy: 0.6515 - val_loss: 0.7295 - val_accuracy: 0.6881
Epoch 12/15
129/129 [==- loss: 0.7636 - accuracy: 0.6578 - val_loss: 0.7246 - val_accuracy: 0.6881
Epoch 13/15
129/129 [==- loss: 0.7584 - accuracy: 0.6744 - val_loss: 0.7262 - val_accuracy: 0.6915
Epoch 14/15
129/129 [==- loss: 0.7651 - accuracy: 0.6702 - val_loss: 0.7185 - val_accuracy: 0.6893
Epoch 15/15
129/129 [==- loss: 0.7446 - accuracy: 0.6851 - val_loss: 0.7081 - val_accuracy: 0.6983
```

Tabla 14 :Etapas de entrenamiento tras el Fine Tunning Pt 2 (ResNet).

Análisis de los resultados

Tras la primera parte del entrenamiento y validación de cada uno de los modelos, en la Tabla 7 se ha podido observar que los niveles de precisión y pérdida de *MobileNet* han sido levemente mejores que los aportados por *Xception* (Tabla 8) y muy por encima ambos del bajo nivel de consistencia que se ha encontrado para *ResNet* en la Tabla 9.

Pero concluir que *MobileNet* es el modelo que mejor funciona para este *dataset* en esta fase era precipitado y así se ha mostrado tras el *Fine Tunning*. Una fase que ha beneficiado considerablemente al modelo *ResNet* y así lo muestran la Figura 35 y la Figura 36, obteniendo un crecimiento constante y valores nulos de sobreajuste. También *Xception* se ha visto beneficiado con el nivel más alto de precisión en la tarea de validación (Tabla 12), pero con valores de sobreajuste grandes. Algo similar le ha pasado al modelo con *MobileNet* (Tabla 11 y Figura 32), aunque con valores más reducidos tanto de precisión, como de sobreajuste.

Con estos resultados, se puede concluir que *MobileNet* es el modelo pre-entrenado que aporta un entrenamiento más consistente. Aunque un entrenamiento largo, de muchas épocas, podría dar lugar a *ResNet* como un modelo válido.

De nuevo, estas conclusiones son precipitadas, pues los periodos de entrenamiento y validación pueden conducir a error, ya que estos resultados hacen referencia a la precisión de la fase de entrenamiento. Pero si introducimos nuevos datos, ¿la precisión y la diferencia entre modelos será la misma? Para obtener unas valoraciones concluyentes sobre cada uno de los casos, habrá que situarlos en un contexto real, es decir, realizar una evaluación sobre datos desconocidos, que no hayan sido utilizados para el entrenamiento o validación.

4.5. Evaluación

Los proyectos que desarrollan software suelen ir acompañados de una serie de requisitos de funcionamiento que suelen acordarse con el cliente antes de comenzar. Para considerar que se están cumpliendo los objetivos se realizan una serie de pruebas de evaluación, que ayudan a detectar defectos y fallos que arreglar. En el caso de la medicina, en muchos casos, el paciente se juega la vida. Es decir, los fallos son inadmisibles.

El aprendizaje automático se puede resumir como la aportación de datos a un modelo para que este genere una lógica. La evaluación dentro de este tipo de proyectos es una fase del desarrollo y consiste en garantizar que la lógica es coherente en cualquier situación.

Este proyecto no conlleva la creación de un software como tal, plantea el estudio de la aplicación de una técnica sobre cierto problema. Por ello, se van a aplicar las dos concepciones de evaluación de un modo analítico. La primera recogerá los requisitos que ha de tener una herramienta destinada a resolver problemas en sanidad y comprobar si fuese posible desarrollarla como tal. Mientras que la segunda está destinada a los resultados de cada modelo sobre el conjunto de datos de test y al análisis de estos.

4.5.1. Validación clínica

La validación clínica es el proceso por el cual una herramienta se considera apta para ser utilizada como solución o instrumento de ayuda en un entorno clínico o sanitario. En este caso concreto, la validez clínica será la capacidad del modelo para diagnosticar la presencia o ausencia de enfermedad.

La validación de nuevas tecnologías requiere de un estudio a gran escala y se han de analizar todas las situaciones críticas para así encontrar las posibles fuentes de interferencia.

No existe un protocolo de validación clínica establecido u ofrecido por un organismo oficial de autoridad. Por ello, en este tipo de herramientas las pruebas suelen comenzar con la comparativa entre la precisión de las predicciones obtenidas con la herramienta y los diagnósticos de médicos especialistas. Tras esto se realizan pruebas de campo en diferentes clínicas, para comprobar si es funcional en situaciones reales.

Además de conseguir valores que demuestren la fiabilidad y seguridad de la herramienta, una parte importante para conseguir la validación clínica es la creación de confianza en el sector científico que se abarca y en el paciente que va a ser atendido.

Debido a la dificultad que supone organizar y conseguir la ayuda de médicos dispuestos a colaborar, la dimensión temporal que un análisis de este tipo conlleva y la imposibilidad en la generación de confianza, esta parte de la evaluación no es viable en un proyecto de este tipo.

4.5.2. Evaluación del modelo

En esta sección se recoge un análisis de la funcionalidad y precisión de los modelos en una situación desconocida. Es decir, se probarán los algoritmos con un conjunto de datos específico que no haya sido utilizado anteriormente y se estudiará el resultado con el fin de aportar la seguridad comentada anteriormente.

El código desarrollado para este proceso se encuentra al final de los *scripts* de cada uno de los modelos pre-entrenados que se nombraron para el apartado anterior. Se pueden encontrar en el apartado Desarrollo de los modelos del Anexo II. Código del proyecto.

El primer paso en este proceso es la obtención de un *array* de predicciones para las imágenes del conjunto de datos test obtenido anteriormente. Esto es muy sencillo mediante *Keras*, ya que el propio modelo, si ha sido entrenado, contiene un método llamado *predict()*. Esta función devuelve un *array* de Numpy con una predicción de clase para cada uno de los datos, en este caso imágenes, pasados como atributo. Es decir, habrá que separar las etiquetas de las imágenes para pasar estas últimas como único argumento.

Este array de predicciones que aporta cada modelo se utilizará para obtener una serie de métricas, con el fin de realizar un examen objetivo de estos resultados (Billa, 2019). Para el análisis de cada una de las métricas de evaluación se han de comprender una serie de términos:

- ***True Positive* (TP):** Elementos que son clasificados como pertenecientes a la clase que les corresponde.
- ***False Positive* (FP):** Elementos que son clasificados como pertenecientes a una clase que no es la correcta.
- ***True Negative* (TN):** Elementos que son clasificados como no pertenecientes a una clase de manera correcta.

- ***False Negative* (FN):** Elementos que son clasificados como no pertenecientes a una clase que sí les corresponde.

Con estos conceptos como base de la evaluación aparecen diferentes técnicas que permiten evaluar la calidad de un modelo. Para el desarrollo de estas se ha utilizado la librería *Scikits-learn* que, mediante *metrics*, proporciona una serie de funciones que permiten la evaluación de modelos. Encontramos una breve descripción de este en el apartado de librerías destinadas a *computer vision* del Anexo I. Python.

Precisión de clasificación

Se trata del valor más básico para el análisis de la calidad del modelo. Es la métrica que se obtiene en cada una de las iteraciones durante el entrenamiento y sobre la que se ha concluido anteriormente. Consiste en una ratio entre las predicciones positivas (TP + TN) y el total de predicciones, se recoge con la Ecuación 1.

$$Accuracy = \frac{Total\ Positive\ Prediction}{Total\ Prediction}$$

Ecuación 1: Precisión de clasificación.

Esta medida no se considera óptima para la evaluación, ya que puede existir una alta tasa de verdaderos negativos (TN) y que el modelo no clasifique correctamente la ratio de predicciones positivas (TP).

Matriz de confusión

Esta técnica consiste en la construcción de una tabla de $N \times N$ valores, dónde *N* es el número de clases diferentes (Billa, 2019), en este caso 4. Uno de los ejes hace referencia a la clase que predice el modelo y el otro es la clase real. De este modo, se pueden observar las predicciones positivas y negativas para cada una de las clases.

Para el desarrollo de la matriz de confusión se han utilizado las funciones *confussion_matrix()* y *ConfusionMatrixDisplay()* de *Skicit-learn*, que generan y muestran por pantalla la matriz. Los parámetros de la primera función serán las etiquetas reales y las obtenidas mediante la predicción descrita anteriormente. La segunda precisará del resultado obtenido de *confussion_matrix()* y el nombre de las clases, que en este caso se van a reducir a **bcc**, **mel**, **nev** y **scc** para que el resultado mostrado sea más legible e interpretable.

Resultados de la matriz de confusión

En el presente subapartado se muestran las matrices de confusión obtenidas para los tres modelos pre-entrenados en forma de tabla con un gradiente de color que permite ver más claramente máximos y mínimos.

***MobileNet*:** Con la primera visualización, mostrada en la Figura 37, se puede observar la escasa certeza que presenta el algoritmo, de 883 imágenes solo ha clasificado correctamente 235 imágenes.

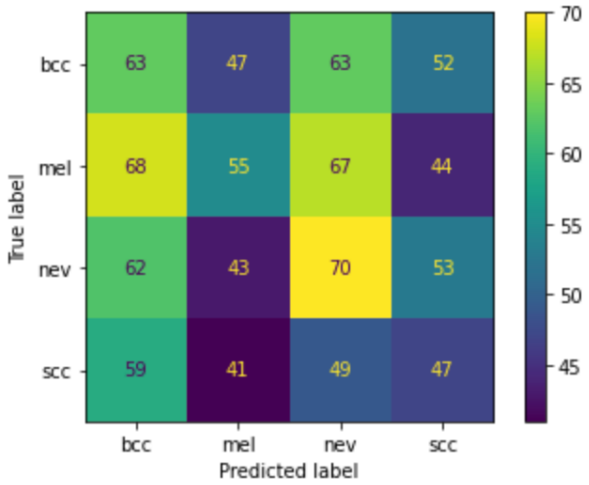


Figura 37: Matriz de confusión (MobileNet).

Además, ninguna de las clases presenta un resultado destacable como se puede observar en la diagonal de la matriz. Se encuentran valores muy cercanos en todas las celdas de la matriz, que viran entre 40 y 70. La única clase que presenta un número superior de TP que FN o FP es el *nevus*, aunque con una diferencia muy pequeña.

Estos resultados llevan a la conclusión de que el modelo ha sobreajustado y la poca eficiencia del modelo. No se podría considerar una solución real, y menos en un contexto tan estricto y preciso como es la sanidad.

***Xception*:** Los resultados para este modelo son todavía más homogéneos para todas y cada una de las celdas (Figura 38). El valor más optimista hace referencia a falsos negativos, ya que se trata de imágenes clasificadas como benignas, pero que en realidad son melanomas. Este dato es algo bastante negativo, ya que se estaría diagnosticando como sano a un paciente enfermo que precisa de una atención médica inmediata.

Con *squamous cell carcinoma* como la clase mejor clasificada, la diagonal de verdaderos positivos presenta valores intermedios que, al no estar muy lejos del resto

de valores, hacen indicar de la poca capacidad del modelo para ser seguro en la clasificación, algo que no puede permitirse en una herramienta que pretende convertirse en apoyo médico.

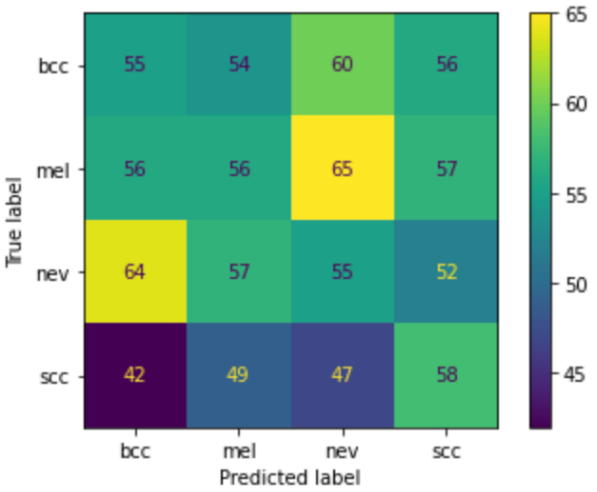


Figura 38: Matriz de confusión (Xception).

***ResNet*:** En la Figura 39, se puede observar una homogeneidad superior a la mostrada en la matriz anterior. Sólo se diferencia un *outlier* y hace referencia a melanomas mal clasificados como nevus, es decir, falsos negativos.

Ninguna de las clases presenta un valor alto de clasificaciones correctas y esto, del mismo modo que en los casos anteriores, lleva a la conclusión de que el modelo no proporciona un nivel de acierto que permita ser optimista.

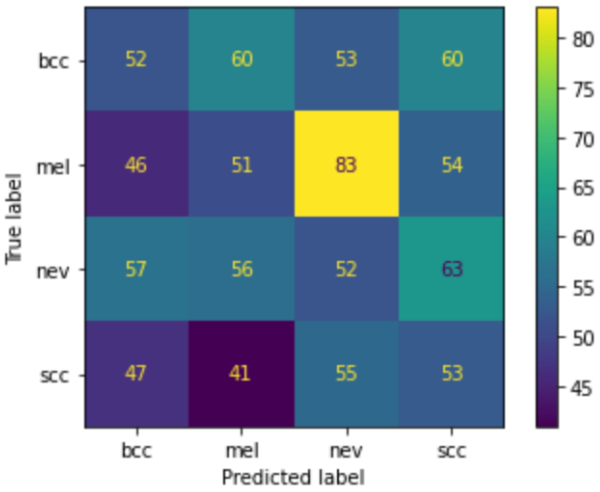


Figura 39: Matriz de confusión (ResNet).

Métricas de clasificación

Además de observar sobre que clases se están cometiendo errores, con la matriz de confusión como representación del resultado, se pueden obtener nuevas métricas que permiten analizar la ratio y calidad de predicciones para cada una de las clases (Heras, 2020).

Una es la ***precisión***, que indica la calidad de la predicción, calcula el porcentaje de aciertos sobre las predicciones de la clase positiva (Ecuación 2). Otra métrica es el ***recall*** o ***True Positive Rate*** (TPR), que mide la cantidad, calcula el porcentaje de la clase positiva que se ha podido identificar (Ecuación 3).

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Ecuación 2: Precisión de evaluación.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Ecuación 3: Recall.

Pero cuando se habla de modelos de predicción, normalmente se habla de una sola medida de evaluación. Esta medida es el ***F1-score***, que combina ambas métricas a partir de la media armónica (Ecuación 4). Esta medida aporta un valor de precisión único que servirá como indicativo general de la calidad y la precisión del modelo.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Ecuación 4: F1-score.

Para obtener estas métricas, la librería *Skicit-learn* proporciona una función que muestra un reporte de la clasificación según las predicciones obtenidas. Esta función es *classification_report()* y precisará del *array* de predicciones, del conjunto de etiquetas reales y del nombre de las clases para utilizar en la visualización.

Resultados de las métricas de clasificación

A continuación, se han obtenido las métricas definidas para cada uno de los modelos. De este modo, se podrán comparar los resultados de precisión real que estos obtienen en una situación externa a la de entrenamiento.

MobileNet: Como ya se intuyó con la matriz de confusión, la Tabla 15 muestra resultados de F1 extremadamente bajos, no llegando ni al 30% de precisión.

Destaca el resultado para la clase nevus sobre el resto, es decir, clasifica mejor las imágenes benignas. Y el valor de precisión para la clase melanoma, aunque el bajo nivel de *recall* hace que su F1 sea más bajo, lo que indica una mejor tasa de FP que de FN. Aún con ello, resultados tan bajos reflejan la imposibilidad de utilizar un modelo con esta simpleza como solución clínica.

	precision	recall	f1-score	support
bcc	0.25	0.28	0.26	225
mel	0.30	0.24	0.26	234
nev	0.28	0.31	0.29	228
scc	0.24	0.24	0.24	196
accuracy			0.27	883
macro avg	0.27	0.27	0.26	883
weighted avg	0.27	0.27	0.27	883

Tabla 15: Métricas de clasificación (MobileNet).

Xception: Para este modelo encontramos mas homogeneidad en los resultados que en el caso anterior, pero siguen siendo muy bajos. De hecho, son peores, aunque con una diferencia escasa.

La clase que mejor resultados ha dado es *squamous cell carcinoma*, como se adelantó con la matriz de confusión, gracias a un mejor valor de *recall*, es decir, una mejor identificación de la clase.

Del mismo modo que con el modelo anterior, este algoritmo no se puede considerar como una solución para ninguno de los problemas planteados.

	precision	recall	f1-score	support
bcc	0.25	0.24	0.25	225
mel	0.26	0.24	0.25	234
nev	0.24	0.24	0.24	228
scc	0.26	0.30	0.28	196
accuracy			0.25	883
macro avg	0.25	0.26	0.25	883
weighted avg	0.25	0.25	0.25	883

Tabla 16: Métricas de clasificación (Xception).

ResNet: El último de los modelos a analizar no mejora los resultados anteriores, sino que los empeora, aunque con una diferencia pequeña de nuevo. No se llega al 25% de precisión con F1.

Las clases presentan menos diferencia de valores entre ellas, lo que indica mayor homogeneidad y menos sesgo para resultados que no pueden considerarse óptimos. Siendo, del mismo modo que con *Xception*, *squamous cell carcinoma* la que mejor resultado ofrece.

De nuevo, se observan resultados demasiado bajos como para plantear el modelo como válido para el diagnóstico clínico mediante la clasificación de imágenes.

	precision	recall	f1-score	support
bcc	0.26	0.23	0.24	225
mel	0.25	0.22	0.23	234
nev	0.21	0.23	0.22	228
scc	0.23	0.27	0.25	196
accuracy			0.24	883
macro avg	0.24	0.24	0.24	883
weighted avg	0.24	0.24	0.24	883

Tabla 17: Métricas de clasificación (ResNet).

Evaluación general

Realizando un balance general de los resultados obtenidos, se puede concluir que ninguno de los modelos se puede plantear como una solución realista a los problemas planteados.

Sí cabe destacar que, con los parámetros seleccionados para el modelo general planteado, *MobileNet* se ha visto por encima de los otros dos casos, con resultados levemente superiores.

Cambiando parámetros del modelo general se ha comprobado que, alterando la arquitectura hacia un algoritmo con más épocas se podría observar una mejora en *ResNet*, algo que se venía intuyendo en su proyección. Y la utilización de imágenes más grandes y capas de neuronas más extensas, mejoran considerablemente la evolución de entrenamiento de *Xception*. Aunque no parece que la influencia de esto fuese a ser grande sobre los resultados de la evaluación.

Recordando los valores de precisión que planteaban los proyectos mencionados en 2.3.1. Proyectos existentes dentro de 2. Contexto y estado del arte, todos por encima del 70%, no es realista plantear a los modelos generados en este proyecto y el *Transfer Learning* como competidores o como una solución viable en un contexto clínico.

Y aunque muchos de estos proyectos planteaban y utilizaban el *Transfer Learning* como parte importante en la extracción de características, los algoritmos y modelos eran

mucho más complejos. Es por esto que la comparativa entre los modelos existentes recogidos y los generados en este proyecto no juegan un papel equitativo.

Una comparación realista y útil para lo que se trata de analizar en este proyecto, que es si el Transfer Learning se puede plantear como una solución en tareas de clasificación complejas como es el diagnóstico médico, es con un modelo básico y habitual en la clasificación de imágenes, que utilice cualquiera de las técnicas planteadas en 2.2.3. Clasificación de imágenes.

Para realizar este análisis se ha generado una red neuronal convolucional (*CNN*) sobre la que se introducirán los mismos datos que los que se utilizaron durante la fase de desarrollo (TensorFlow, 2021).

El modelo está conformado por la misma estructura de capas que se utilizó para el modelo general, cambiando la capa pre-entrenada por la red convolucional. La red está formada por tres capas convolucionales de dos dimensiones, *Conv2D()*, con salida de 16, 32 y 64 neuronas respectivamente y función de activación ReLu, y alternando entre cada capa se han utilizado capas de agrupamiento *MaxPooling2D()*. Al terminar, se utiliza una capa *Flatten()* que convierte la salida al formato necesario añadiendo una dimensión extra.

	precision	recall	f1-score	support
bcc	0.23	0.04	0.07	225
mel	1.00	0.00	0.01	234
nev	0.26	0.54	0.36	228
scc	0.23	0.43	0.30	196
accuracy			0.25	883
macro avg	0.43	0.26	0.18	883
weighted avg	0.44	0.25	0.18	883

Tabla 18: Métricas de clasificación (CNN).

Si bien es cierto que la precisión F1 general del modelo es similar a la obtenida por los modelos pre-entrenados (25%), el valor de la *CNN* no es realista. Ya que los resultados obtenidos, mostrados en la Tabla 18, reflejan un claro sesgo del modelo hacia las imágenes benignas y el *squamous cell carcinoma*, que obtienen por encima del 30% de precisión. Mientras que las otras dos clases no llegan a un 10% de precisión.

Esto hace ver que la utilización de *Transfer Learning* en clasificación de imágenes es una opción bastante útil para evitar modelos sesgados con respecto a redes neuronales convolucionales, con valores de precisión más o menos homogéneos para todas las

clases. Y pueden ser una base importante sobre la que sustentar algoritmos más complejos.

5. Conclusiones y trabajo futuro

Llegados al final de la memoria, y con ello, del desarrollo del proyecto, en este capítulo se exponen una serie de conclusiones en base al trabajo previo existente y con cierto poder de mejoría en el futuro. En todo proceso tecnológico, la innovación y la renovación son parte del proceso de desarrollo.

5.1. Conclusiones

Este proyecto nace como una idea demasiado ambiciosa, en la que se pretendía obtener una solución útil en el campo de la tecnología clínica, con vistas a un desarrollo enfocado al futuro.

Tras el estudio del contexto y del arte que rodea el campo de la Inteligencia Artificial en dermatología, se observó la dificultad que esta primera idea iba a suponer. Con ello, se plantea la comparativa de tecnologías interesantes en el desarrollo de modelos de *Deep Learning*. Ahí es dónde apareció el *Transfer Learning*, una técnica cada vez más usada, con una filosofía de aprendizaje interesante y con gran cantidad de modelos pre-entrenados. Esto ha permitido basar el proyecto en un análisis del posible impacto de esta técnica en la clasificación de imágenes para el diagnóstico de enfermedades.

A día de hoy, con el proyecto finalizado y un desarrollo de 6 meses por delante se ha llegado a las siguientes conclusiones:

- Se han cumplido los objetivos planteados.
- Se ha conseguido realizar una comparativa entre diferentes modelos pre-entrenados, encontrando en *MobileNet* los mejores resultados, con una arquitectura que precisa de menos parámetros y, por ello, tiempo de ejecución.
- El *Transfer Learning* se puede considerar como una técnica interesante y de gran utilizada en la clasificación de imágenes, ya que mejora la manera de extraer características y, junto al *Fine Tunning*, consigue modelos bastante equitativos, con poco sesgo.
- Más concretamente, el Transfer Learning se puede considerar como una técnica de utilidad en el diagnóstico de cáncer de piel si se acompaña de una arquitectura y procesado de imágenes correcto.

- A pesar de cumplir los objetivos, el análisis de resultados ha sido mucho más pesimista de los que se pensó al inicio del proyecto.
- Los resultados no se acercan a los que obtienen otros proyectos como los mostrados en 2.3.1. Proyectos existentes.
- La clasificación de los modelos desarrollados no es buena y, como herramienta, ninguno se puede considerar solución a ninguno de los problemas planteados.
- Los resultados obtenidos hacen ver problemas de sobreajuste en los diferentes modelos.
- A pesar de obtener una base de datos grande, se tuvieron que eliminar muchas de las imágenes, reduciendo la capacidad de entrenamiento.
- El pre-procesado no consiguió resultados del todo positivos en muchas de las imágenes. Hubo que limpiar a mano previamente para conseguir la segmentación que se buscaba.
- Los resultados de las imágenes sin pre-procesar han sido iguales o mejores. Por lo que se podría considerar innecesarios en el desarrollo de este tipo de técnicas.

5.2. Líneas de trabajo futuro

Debido a la limitación temporal que supone un proyecto de este tipo, el desarrollo de la arquitectura del modelo general, sobre el que se han generado las tres soluciones, ha sido limitado y enfocado a un análisis superficial de la situación.

En base a este carácter analítico y a la cantidad enorme de datos que se generan cada día, se han podido plantear diversas líneas futuras para el *Transfer Learning*:

- **Ampliación de la base de datos:** El desarrollo de este tipo de herramientas y proyectos fomentan la obtención, gestión y acceso a datos que, en gran cantidad, pueden ser muy útiles para mejorar los resultados.
- **Mejora del pre-procesado:** El pre-procesado planteado en este proyecto está basado en la detección de melanomas. Es por lo que, para muchas de las imágenes, el algoritmo no funciona como se esperaba. Realizar un procesado más general puede ser una opción interesante. Aunque bien es cierto que se puede incluso evitar pre-procesar, ya que el *Transfer Learning* no parece precisar de ello.

- **Aumentar el número de clases:** Sería interesante, en un futuro, poder abarcar un número importante de clases.

- **Elaboración de una arquitectura más compleja:** El desarrollo de arquitecturas de datos más elaboradas puede suponer mejoras considerables en los resultados de este tipo de técnicas. La introducción de múltiples entradas de datos o algoritmos específicos en la tarea de clasificación posteriores a la extracción de características son ejemplos.

- **Introducción de metadatos:** La inclusión de datos de interés como la edad del paciente, el sexo y el tipo de piel pueden ayudar en el diagnóstico de este tipo de enfermedades. Esto es una mejora bastante segura ya que Google ha desarrollado esto y ha obtenido muy buenos resultados.

6. Bibliografía

Afonja, T., (2017). *Kernel Functions*. https://towardsdatascience.com/kernel-function-6f1d2be6091

Agrawal, S., (2021). *How to split data into three sets (train, validation, and test) And why?* https://towardsdatascience.com/how-to-split-data-into-three-sets-train-validation-and-test-and-why-e50d22d3e54c

ALKolifi, N.S., (2019). A Method of Skin Disease Detection Using Image Processing and Machine Learning. *Procedia Computer Science, 163*, 85-92. DOI 10.1016/j.procs.2019.12.090

American Academy of Dermatology (AAD), (2021). *What is a dermatologist?* https://www.aad.org/public/fad/what-is-a-derm

American Cancer Society, (2021). *Cancer Facts and Figures 2021*. https://www.cancer.org/content/dam/cancer-org/research/cancer-facts-and-statistics/annual-cancer-facts-and-figures/2021/cancer-facts-and-figures-2021.pdf

American Cancer Society, (2021, febrero 12). Tasas de supervivencia del cáncer de piel tipo melanoma. *Detección temprana, diagnóstico y clasificación por etapas*, 24–26. https://www.cancer.org/content/dam/CRC/PDF/Public/8994.00.pdf

American Cancer Society, (2019). Tests for Melanoma Skin Cancer. *Melanoma Skin Cancer Early Detection, Diagnosis, and Staging*. https://www.cancer.org/content/dam/CRC/PDF/Public/8825.00.pdf

Argenziano, G., Soyer, H. P., De Giorgi, V., Piccolo, D., Carli, P., & Delfino, M. (2002). *Dermoscopy: a tutorial. EDRA, Medical Publishing & New Media, 35*.

Avineri, G., Talmor, O., (2020). ISIC-Archive-Downloader. https://github.com/GalAvineri/ISIC-Archive-Downloader

Bhattad, P. B., Jain, V., (2020). Artificial Intelligence in Modern Medicine – The Evolving Necessity of the Present and Role in Transforming the Future of Medical Care. *Cureus, 12(5)*. DOI 10.7759/cureus.8041

Beklemysheva, A., (2020). *Why Use Python for AI and Machine Learning?* https://steelkiwi.com/blog/python-for-ai-and-machine-learning/

Billa, M., (2019, octubre 12). *Testers guide for Testing Machine Learning Models.* https://medium.com/analytics-vidhya/testers-guide-for-testing-machine-learning-models-e7e5cea81264

Brahmbhatt, S., (2013). *Practical OpenCV,* 3-5. https://books.google.es/books?hl=es&lr=&id=_5sQAwAAQBAJ&oi=fnd&pg=PA2&dq=openCV&ots=8UlNbYOaf1&sig=2iMzC6jbg10XWRongDSr6jdYC6o#v=onepage&q=openCV&f=false

Bray, F., Jemal, A., Soerjomataram, I., Laversanne, M., Siegel, R.L., Ferlay, J., Sung, H., (2021). Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries. *CA: A Cancer Journal for Clinicians, 71*, 209–249. DOI 10.3322/caac.21660

Budhiraja, A., (2016, diciembre 15). *Dropout in (Deep) Machine learning*. https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

Bui, P., Liu, Y., (2021). *Using AI to help find answers to common skin conditions*. https://blog.google/technology/health/ai-dermatology-preview-io-2021/

Bushkovskyi, O., (2020, marzo 12). *Computer Vision for Healthcare*. https://theappsolutions.com/blog/machine-learning/computer-vision-for-healthcare/

Calvo, D., (2017, julio 20). *Red Neuronal Convolucional CNN*. https://www.diegocalvo.es/red-neuronal-convolucional/

Castañeda, P., Eljure, J., (2016). El cáncer de piel, un problema actual. *Revista de la Facultad de Medicina de la UNAM, 59*.

Chollet, F., (2020). *The Sequential model*. https://keras.io/guides/sequential_model/

De Alba, A. G., (2019). *La evolución de la inteligencia artificial en las últimas décadas*. https://planetachatbot.com/la-evoluci%C3%B3n-de-la-inteligencia-artificial-en-las-%C3%BAltimas-d%C3%A9cadas-f86e4714160d

Deepan, P., Sudha, L.R., (2020). Chapter 8 - Object Classification of Remote Sensing Image Using Deep Convolutional Neural Network. *The Cognitive Approach in Cloud Computing and Internet of Things Technologies for Surveillance Tracking Systems*, 107-120. DOI 10.1016/B978-0-12-816385-6.00008-8

Dermaten, (2020). *¿Qué es la dermatología?* https://dermaten.es/que-es-la-dermatologia/

Deeks, J.J., Dinnes, J., Williams, H.C., (2020). Sensitivity and specificity of SkinVision are likely to have been overestimated. *Journal of The European Academy of Dermatology and Venereology, 34*, 582-583. DOI 10.1111/jdv.16382

Dwivedi, R., (2020). *How Does K-nearest Neighbor Works In Machine Learning Classification Problem?* https://www.analyticssteps.com/blogs/how-does-k-nearest-neighbor-works-machine-learning-classification-problem

Esteva, A., Chou, K., Yeung, S., Naik, N., Madani, A., Mottaghi, A., Liu, Y., Topol, E., Dean, J., Socher, R., (2021). Deep learning-enabled medical computer vision. *npj Digital Medicine, 5*. DOI 10.1038/s41746-020-00376-2

Fernández, A., (2012). *Python 3 al descubierto.* ISBN: 978-607-707-718-3

Fernández, S. P., Díaz, P. S., (2010). Pruebas Diagnósticas: Sensibilidad y especificidad. *Cuaderno de Atención Primaria 2003; 10: 120-124*. https://www.fisterra.com/mbe/investiga/pruebas_diagnosticas/pruebas_diagnosticas.asp

Gandhi, R., (2018). *Support Vector Machine — Introduction to Machine Learning Algorithms*. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

González, R., (2014). *Python para todos*, 7-9. http://mundogeek.net/tutorial-python/

Gonzalo, A., (2020). *¿Qué es el sobreajuste u overfitting y por qué debemos evitarlo?* https://machinelearningparatodos.com/que-es-el-sobreajuste-u-overfitting-y-por-que-debemos-evitarlo/

Harrison, O., (2018). *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

Heras, J. M., (2020). *Precision, Recall, F1, Accuracy en clasificación*. https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., (2017, abril 17). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.* https://arxiv.org/pdf/1704.04861.pdf

ISIC, (2020). *Content and Layout of the Archive*. https://www.isic-archive.com/#!/topWithHeader/tightContentTop/about/isicArchiveContent

Jain A, Way D, Gupta V, et al, (2021). Development and Assessment of an Artificial Intelligence–Based Tool for Skin Condition Diagnosis by Primary Care Physicians and Nurse Practitioners in Teledermatology Practices. *JAMA Netw Open*, *4(4):e217249*. DOI 10.1001/jamanetworkopen.2021.7249

Jain, T., (2019). *Basics of Image Classification Techniques in Machine Learning*. https://iq.opengenus.org/basics-of-machine-learning-image-classification-techniques/

Jmour, N., Zayen, S., & Abdelkrim, A. (2018). Convolutional neural networks for image classification. *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*. DOI 10.1109/ASET.2018.8379889

Keras, (2021). *Keras Applications.* https://keras.io/api/applications/

Khan, A. I., Al-Habsi, S., (2020). Machine Learning in Computer Vision. *Procedia Computer Science, 167*, 1444–1451. DOI 10.1016/j.procs.2020.03.355

Khan, S., (2019). *Activation Functions.* https://pytech-solution.blogspot.com/2019/01/activation-functions.html

Kingma, D. P., Ba, J. L., (2014). Adam: A method for Stochastic Optimization. *International Conference on Learning Representations.* https://arxiv.org/pdf/1412.6980.pdf

Koech, K. E., (2020). *Cross-Entropy Loss Function*. https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e

Le, J., (2018). *The 5 Computer Vision Techniques That Will Change How You See the World*. https://heartbeat.fritz.ai/the-5-computer-vision-techniques-that-will-change-how-you-see-the-world-1ee19334354b

Lelli, F., (2019). *Neural Networks: The Basics and Collection of Youtube Videos*. https://francescolelli.info/tutorial/neural-networks-a-collection-of-youtube-videos-for-learning-the-basics/

Leo, M. S., (2020). How to Choose the Best Keras Pre-Trained Model for Image Classification. https://towardsdatascience.com/how-to-choose-the-best-keras-pre-trained-model-for-image-classification-b850ca4428d4

Levin, G., Dorsey, B., (2020). *Image Proccessing and Computer Vision.* https://openframeworks.cc/ofBook/chapters/foreword.html

Liu, Y., Jain, A., Eng, C. et al, (2020). A deep learning system for differential diagnosis of skin diseases. *Nat Med, 26*, 900–908. DOI 10.1038/s41591-020-0842-3

López-Sabater, M.B., Fiestas-García, I.M., López-Abadal, A., (2019). El patito feo. *Medicina general y de familia*, *8(3)*, 113-115. DOI 10.24038/mgyf.2019.024

Mayo Clinic, (2019). *Cáncer*. https://www.mayoclinic.org/es-es/diseases-conditions/cancer/symptoms-causes/syc-20370588#:~:text=El%20c%C3%A1ncer%20se%20refiere%20a,destruir%20el%20tejido%20corporal%20normal

MedicineNet, (2002, mayo 15). *Skin – What is it?* https://www.medicinenet.com/script/main/art.asp?articlekey=19340

Mendonca T, Ferreira PM, Marques JS, Marcal AR, Rozeira J, (2013). PH2 - a dermoscopic image database for research and benchmarking. *Annu Int Conf IEEE Eng Med Biol Soc*. DOI 10.1109/EMBC.2013.6610779

Mihajlova, I., (2019). *Everything You Ever Wanted to Know About Computer Vision*. https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e

Moreira, G., (2018). CRISP-DM and what I did wrong. https://blog.magrathealabs.com/crisp-dm-and-what-i-did-wrong-70c4e7e8656

Mumtazimah, M., Md S. Y. M., Muhammad H., (2014). *A review on OpenCV.* DOI 10.13140/RG.2.1.2269.8721

Na8, (2018, noviembre 29). *¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador*. https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

Narkhede, S., (2018).

National Cancer Institute, (2021). *What is cancer?* https://www.cancer.gov/about-cancer/understanding/what-is-cancer#:~:text=Cancer%20is%20a%20disease%20caused,are%20also%20called%20genetic%20changes

Nelson, J., (2020). *You Might Be Resizing Your Images Incorrectly*. https://blog.roboflow.com/you-might-be-resizing-your-images-incorrectly/

NumPy, (2021). *NumPy*. https://numpy.org/

OpenCV, (2020, diciembre 21). *Image Thresholding*. https://docs.opencv.org/4.5.1/d7/d4d/tutorial_py_thresholding.html

OpenCV, (2021, abril 2). Arithmetic Operations on Images. https://docs.opencv.org/4.5.2/d0/d86/tutorial_py_image_arithmetics.html

OpenCV, (2021, agosto 13). *Canny Edge Detection.* https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html

Pan, S.J., Yang, Q., (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering, 22*, 1345 – 1359. DOI 10.1109/TKDE.2009.191

Pandas, (2021). *Pandas*. https://pandas.pydata.org/

Parmar, R., (2018). *Common Loss functions in machine learning.* https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23

Patel, P., (2018, marzo 8). *Why Python is the most popular language used for Machine Learning*. https://medium.com/@UdacityINDIA/why-use-python-for-machine-learning-e4b0b4457a77

Peverelli, R., de Feniks, R., (2017). *SkinVision: Leading mobile solution to monitor, track and understand skin health*. https://www.digitalinsuranceagenda.com/featured-insurtechs/skinvision-leading-mobile-solution-to-monitor-track-and-understand-skin-health/#:~:text=SkinVision%20is%20based%20on%20a,uploaded%20to%20the%20users%20smartphone

Phillips, M., Greenhalgh, J., Marsden, H., Palamaras, I., (2019). Detection of malignant melanoma using artificial intelligence: an observational study of diagnostic accuracy. *Dermatol Pract Concept, 10(1):e2020011*. DOI 10.5826/dpc.1001a11

Pozzebon, S., (2014). *This App Can Detect Skin Cancer In 7 Out Of 10 Cases: Here's How It Works.* https://www.businessinsider.com/how-skinvision-app-works-2014-

11#:~:text=SkinVision%20is%20based%20on%20a,is%20uploaded%20to%20your%20smartphone

Puentes, N., (2021). *Qué es Computer Vision*. https://www.nebulova.es/blog/computer-vision#:~:text=El%20t%C3%A9rmino%20Computer%20Vision%20(Visi%C3%B3n,contenido%20de%20las%20im%C3%A1genes%20digitales

Pypi (2020). *Split-folders 0.4.3*. https://pypi.org/project/split-folders/

Rath, S. R., (2019). *An Introduction to k-Nearest Neighbors in Machine Learning.* https://debuggercafe.com/an-introduction-to-k-nearest-neighbors-in-machine-learning/

Rosebrock, A., (2015). *OpenCV Gamma Correction.* https://www.pyimagesearch.com/2015/10/05/opencv-gamma-correction/

Saha, S., (2018, diciembre 15). *A Comprehensive Guide to Convolutional Neural Networks.* https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Sanghvi, K., (2020). *Image Classification Techniques*. https://medium.com/analytics-vidhya/image-classification-techniques-83fd87011cac

Simon, S., (2020). *How to Spot Skin Cancer*. https://www.cancer.org/latest-news/how-to-spot-skin-cancer.html

SkinVision, (2021). *SkinVision*. https://www.skinvision.com/

Solawetz, J., (2020, septiembre 4). *The Train, Validation, Test Split and Why You Need It.* https://blog.roboflow.com/train-test-split/

Tadakaluru, S., (2020). *Introduction to Image Pre-processing | What is Image Pre-processing?* https://www.mygreatlearning.com/blog/introduction-to-image-pre-processing/

Tarasenko, A. O., Yakimov, Y. V., Solovyov, V. N., (2019). *Convolutional neural networks for image classification.* http://elibrary.kdpu.edu.ua/xmlui/handle/123456789/3682

TensorFlow, (2021). *Clasificación de imágenes*. https://www.tensorflow.org/tutorials/images/classification

Tschandl, P., Sinz, C., Kittler, H., (2018). Domain-specific classification-pretrained fully convolutional network encoders for skin lesion segmentation. *Computers in Biology and Medicine*. DOI 10.1016/j.compbiomed.2018.11.010

Udrea, J., Mitra, G.D., Costea, D., Noels, E.C., Wakkee, M., Siegel, D.M., de Carvalho, T.M., Nijsten, T.E.C., (2019). Accuracy of a smartphone application for triage of skin lesions based on machine learning algorithms. *Journal of The European Academy of Dermatology and Venereology, 34*, 648-655. DOI 10.1111/jdv.15935

Van der Walt, S., Schönberger, J. L., Nuñez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., (2014). *scikit-image: Image processing in Python*. DOI 10.7287/peerj.preprints.336v2

Wang, S. C., (2003). Artificial Neural Network. *Interdisciplinary Computing in Java Programming. The Springer International Series in Engineering and Computer Science*, *743*. DOI 10.1007/978-1-4615-0377-4_5

Watson, H. A., Tribe, R. M., Shennan, A. H., (2019). The role of medical smartphone apps in clinical decision-support: A literature review. *Artificial Intelligence In Medicine, 100*. DOI 10.1016/j.artmed.2019.101707

Wei, Y., Zhang, Y., Huang, J., Yang, Q., (2018). Transfer Learning via Learning to Transfer. *Proceedings of the 35th International Conference on Machine Learning, 80*, 5085-5094.

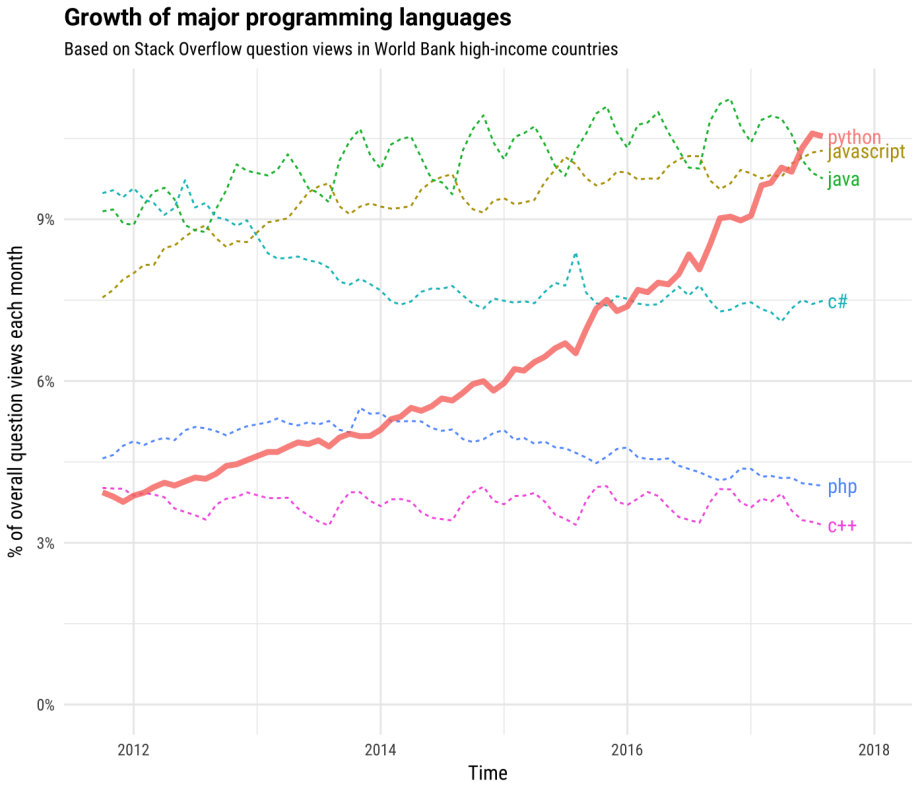
Zhang, N., Cai, Y., Wang, Y., Tian, Y., Wang, X., Badami, B., (2019). Skin Cancer Diagnosis Based on Optimized Convolutional Neural Network. *Artificial Intelligence In Medicine.* DOI 10.1016/j.artmed.2019.101756

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q., (2020). A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE, 109*, 43 – 76. DOI 10.1109/JPROC.2020.3004555

Anexos

Anexo I. Python

Python es un lenguaje de programación de alto nivel, multipropósito y con tipado dinámico, creado por Guido Van Rossum como proyecto paralelo (González, 2014). Y, como se puede observar en la **¡Error! No se encuentra el origen de la referencia.**, se trata de uno de los lenguajes más utilizado en la actualidad en el desarrollo de software.



Su popularidad se debe principalmente a la sencillez y claridad que presenta, pudiendo escribir scripts muy cercanos al lenguaje natural. Se puede utilizar en cualquier plataforma sin necesidad de instalación de librerías externas.

Pero el activo más importante que presenta este lenguaje es la enorme comunidad que lo conforma. Es una herramienta *open source* para la que cualquiera puede desarrollar librerías y documentación útiles para el aprendizaje y desarrollo de herramientas (Fernández, 2012).

Pero **¿por qué usar Python para desarrollar *machine learning*?**

El ingeniero en este sector tiene que lidiar con tareas complejas para la comprensión y análisis de los datos. En situaciones complejas conceptualmente, una implementación

rápida puede ayudar a validar una idea. Es por ello por lo que una sintáxis sencilla y clara y las enormes posibilidades para obtener código externo son muy útiles para el desarrollo de prototipos y modelos (Patel, 2018).

Librerías para CV

En esta sección se van a enumerar algunas de las diferentes herramientas dedicadas a *computer vision* que proporciona la comunidad Python y que se han utilizado para el desarrollo de este proyecto.

OpenCV

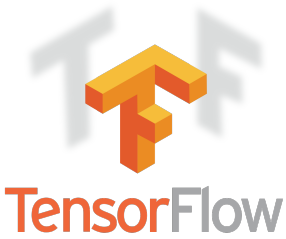
Librería específica de CV desarrollada por Intel en 1999. Fue diseñada con un fuerte enfoque en aplicaciones en tiempo real y su principal objetivo era ser eficiente computacionalmente (Brahmbhatt, 2013).

OpenCV está ganando rápidamente popularidad, gracias a que dispone de una cantidad enorme de módulos, documentación y algoritmos optimizados. Proporciona una infraestructura para este tipo de aplicaciones y su filosofía se basa en la difusión y aportación constante. Permite a los investigadores poner en marcha rápidamente demos o proyectos de investigación, aprovechando la gran colección de algoritmos que ya están disponibles (Mumtazimah, 2015).

https://docs.opencv.org/master/index.html

TensorFlow

Esta es la librería más popular en el desarrollo de *Deep Learning*. Desarrollada por Google y que se convirtió en *open source* en 2015. Y aunque su función principal es el desarrollo de redes neuronales, tiene una versatilidad enorme en diferentes dominios del aprendizaje automático.



Se basa en modelos matemáticos que son construidos mediante capas o neuronas. Estos modelos pueden abarcar un dominio muy simple o complejo, dependiendo de la arquitectura de capas que se utilice. Además, proporciona una documentación muy completa con ejemplos y tutoriales que permiten un aprendizaje rápido y práctico.

https://www.tensorflow.org/api_docs

Keras

Keras es una librería destinada al desarrollo de redes neuronales. Es código abierto y fue escrita en Python por François Chollet. Se trata de una interfaz de gráficos de flujo, como son las redes neuronales, en lugar de ser un *framework*. Esto significa que se ha de ejecutar sobre otros *frameworks* base como son Microsoft Cognitive Toolkit, Theano o Tensorflow, cuya combinación es la más habitual.

Esta interfaz proporciona abstracciones intuitivas de alto nivel que permiten generar algoritmos con código más sencillo independientemente del *framework* sobre el que se implemente. Proporciona un API de alto nivel con gran cantidad de ejemplos. Esta documentación unida a la proporcionada por TensorFlow son un gran complemento en el aprendizaje y desarrollo de *machine learning*.

https://keras.io/api/

Scikits-learn

Se trata de una librería desarrollada por un equipo activo de desarrolladores e incluye algoritmos para el procesamiento de imágenes. Es básicamente una caja de herramientas de procesamiento de imágenes para *SciPy*. Proporciona una API para Python bien documentada (Van der Walt et al, 2014).

Incluye varios algoritmos de clasificación, regresión y *clustering*. Está diseñada para operar con librerías numéricas como NumPy. Es muy útil para el análisis de resultados y el modelado estadístico.

https://scikit-learn.org/stable/modules/classes.html

Anexo II. Código del proyecto

En esta sección se recogen los diferentes scripts que se han utilizado para el desarrollo del proyecto.

Homogeneización de los datos

- https://drive.google.com/file/d/1CSw_yahJvUNDO-sGUrxDgaCgTlJZbbv1/view?usp=sharing

Pre-procesado

- https://colab.research.google.com/drive/185kb-HJ6cPUfHwvJ_qS_ZcuXnO3w8ZtO?usp=sharing

Escoger modelo pre-entrenado

- https://colab.research.google.com/drive/1-GXJETLE-EXI6U_Adb6gv2RIUqDWR1LQ?usp=sharing

Desarrollo de los modelos

MobileNet

- https://colab.research.google.com/drive/1oed67GAGbVkbJJxMDgPxeSMPmhzprz-g?usp=sharing

Xception

- https://colab.research.google.com/drive/1OHS8ZpNsuBsLqtjsX0qmR16nzkA1F4rf?usp=sharing

ResNet101V2

- https://colab.research.google.com/drive/1GsBViJYIqA0E4AOxqoRKg7PkG5x4scp9?usp=sharing