ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DETECCIÓN DE TUMORES CUTÁNEOS MALIGNOS Y BENIGNOS UTILIZANDO UNA RED NEURONAL CONVOLUCIONAL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

JONATHAN ISRAEL MOROCHO JIMÉNEZ

jonathanmorochoj@gmail.com

DIRECTOR: MSc. HENRY PATRICIO PAZ ARIAS

henry.paz@epn.edu.ec

Quito, enero de 2019

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Jonathan Israel Morocho Jiménez, bajo mi supervisión.

_____

MSc. Henry Patricio Paz Arias

DIRECTOR DEL PROYECTO

DECLARACIÓN DE AUTORÍA

Yo, Jonathan Israel Morocho Jiménez, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Jonathan Israel Morocho Jiménez

DEDICATORIA

Dedico este trabajo a mi familia, en especial a mi madre y a mi hermano, los cuales fueron, son y serán un gran pilar en mi vida.

AGRADECIMIENTO

A mi madre y a mi hermano, por haber sido un soporte todos estos años para la culminación de mis estudios.

A mi director de tesis, por la paciencia y los consejos para que el presente trabajo se haya completado con éxito.

A los pocos profesores que realmente se enfocaron en ser una fuente de inspiración para que sus alumnos sean mejores personas y profesionales.

A mis compañeros de la carrera con quienes compartí éxitos académicos y deportivos.

A Diana Luna, por todo ese apoyo y amor incondicional de tantos años.

Gracias.

ÍNDICE

RESUMEN

El presente trabajo se centra en el uso de la arquitectura de la red neuronal convolucional Inception v3 y la técnica de transferencia de aprendizaje para crear un modelo que pueda clasificar imágenes dermatoscópicas de melanoma (tumor cutáneo maligno) y nevus (tumor cutáneo benigno) con una alta exactitud. Al carecer de una gran cantidad de datos para el entrenamiento, se hizo uso del método de aumento de datos fuera de línea para proveer de más información al clasificador en la ejecución de las tareas de entrenamiento y validación. Para el entrenamiento de la última capa totalmente conectada se usó el algoritmo de descenso de gradiente por mini lotes, logrando una exactitud del 89,63%. Finalmente se creó una aplicación web que sirve un recurso basado en la arquitectura REST para usar el modelo generado a través de una aplicación móvil.

Palabras clave: Transferencia de aprendizaje, Aumento de datos, Inception v3, Redes neuronales convolucionales, Tumores cutáneos, Aprendizaje profundo.

ABSTRACT

The present work focuses on the use of the Inception v3 convolutional neural network architecture and the transfer learning to create a model that can classify dermoscopic images of melanoma (malignant cutaneous tumor) and nevus (benign cutaneous tumor) with a high accuracy. Lacking a large amount of data for training, the offline data augmentation method was used to provide more information to the classifier in the execution of training and validation tasks. For the training of the last fully connected layer, the mini-batch gradient descent algorithm was used, achieving an accuracy of 89,63%. Finally, a web application was created that serves a resource based on the REST architecture to use the generated model by a mobile application.

Keywords: Transfer learning, Data augmentation, Inception v3, Convolutional neural networks, Cutaneous tumors, Deep learning.

1. INTRODUCCIÓN

1.1. Antecedentes

Según el último informe de SOLCA (Sociedad de Lucha Contra el Cáncer del Ecuador) sobre la epidemiología del cáncer en la ciudad de Quito, correspondiente al período 2011-2013, existe un incremento de casos de melanoma, situándose tasas de incidencia de 37.5 casos por cada 100 000 mujeres y 40.7 casos por cada 100 000 hombres [1]. Si este tipo de cáncer se detecta en su primera etapa, se logra tener una tasa de supervivencia cercana al 97% [2].

Las aplicaciones móviles han sido de gran ayuda en el campo médico [3]. Se ha desarrollado algunas aplicaciones de este tipo para encontrar casos de melanoma. Una de ellas es SkinVision que permite al usuario la evaluación de sus lunares y las condiciones de su piel para determinar el riesgo de cáncer de piel [4] mediante el uso de un algoritmo de procesamiento de imágenes propietario basado en análisis fractal de imágenes, el cual tiene una exactitud de 81% [5]. Otra aplicación móvil llamada Mole Investigator permite al usuario determinar la probabilidad de que un lunar sea melanoma, mediante un procesamiento previo de imágenes para extraer características como asimetría, borde, color; y posteriormente realizando la clasificación usando el modelo de regresión random forest, alcanzando una exactitud de 79.8% [6].

En el presente trabajo se propone superar estos porcentajes de exactitud mediante el uso de una red neuronal convolucional como alternativa al algoritmo propietario de procesamiento de imágenes de SkinVision y al modelo de regresión random forest de la aplicación Mole Investigator. Para esto se toma en cuenta investigaciones similares con este tipo de redes [7]. La razón de usar estos dos trabajos de referencia es que, al momento de revisar el estado del arte, eran de los pocos que trataban el problema de la clasificación de imágenes de melanoma con algoritmos diferentes a las redes neuronales convolucionales. Además, se enfocaban en que sus modelos se usen mediante dispositivos móviles, lo que permitía tener un punto de partida para verificar los posibles problemas y soluciones que tuvieron los autores de esos trabajos y que pudiesen usarse en el desarrollo de este trabajo.

1.2. Objetivos

1.2.1. Objetivo General

Detectar el melanoma (tumor cutáneo maligno) y nevus (tumor cutáneo benigno) con un nivel de exactitud superior al 81% en imágenes dermatoscópicas de pacientes mediante el uso de una red neuronal convolucional.

1.2.2. Objetivos Específicos

- Obtener un conjunto de imágenes dermatoscópicas de melanoma (tumor cutáneo maligno) y nevus (tumor cutáneo benigno) que sea útil para el entrenamiento, validación y pruebas de la red neuronal.
- Proponer una arquitectura basada en la red neuronal convolucional Inception v3 usando la librería para machine learning TensorFlow.
- Entrenar, validar y probar la red neuronal convolucional propuesta.
- Construir un servicio web para recibir imágenes dermatoscópicas y clasificarlas usando el modelo entrenado de la red neuronal convolucional propuesta.
- Construir el prototipo de una aplicación móvil para la plataforma Android, la cual permita capturar una imagen y enviarla al servicio web creado para su clasificación usando el modelo entrenado.

1.3. Alcance

El proyecto está orientado a crear una red neuronal convolucional que clasifique imágenes dermatoscópicas de melanoma o nevus. El modelo entrenado se utilizará mediante un servicio web, para lo cual se creará una aplicación móvil que consuma dicho servicio mediante el envío de una imagen capturada usando la cámara fotográfica que suelen integrar los teléfonos inteligentes. Esta aplicación móvil mostrará al usuario el resultado de la clasificación en forma de porcentaje.

1.4. Herramientas

Se usarán herramientas enfocadas en los diferentes problemas que abarca este proyecto. Se requiere de un lenguaje de programación fácil de aprender para crear la red neuronal convolucional. Este mismo lenguaje se usará en una aplicación web, por lo que se usará un framework adecuado para este propósito. Se requiere de una plataforma robusta de

hardware que permita ejecutar ciertas tareas necesarias para obtener un modelo de clasificación con un alto nivel de exactitud.

Adicionalmente, se usará una plataforma en la nube la cual permita alojar la aplicación web y ejecutarla posteriormente. Finalmente, se usará un framework para crear la aplicación móvil.

1.4.1. Python

Python es un lenguaje interpretado de alto nivel, multiparadigma, simple y consistente [8]. Este lenguaje será usado junto a las herramientas TensorFlow, Keras y Flask para la creación de los scripts que ejecutarán funcionalidades específicas.

1.4.2. TensorFlow

Es una biblioteca de software de código abierto para machine learning y cálculo numérico de alto rendimiento. Su arquitectura permite una fácil implementación de computación en una variedad de plataformas usando CPU y GPU [9]. Usando esta biblioteca, se construirá la arquitectura de la red neuronal convolucional y los scripts necesarios para el entrenamiento, validación y pruebas del modelo.

1.4.3.Keras

Es una API de alto nivel escrita en Python y que funciona por encima de otros frameworks como TensorFlow, CNTK, o Theano [10]. Provee de herramientas de procesamiento de imágenes, las cuales se usarán en el conjunto de imágenes de entrenamiento, validación y pruebas [11].

1.4.4. Flask

Es un framework web usado para crear aplicaciones web basadas en el lenguaje Python, el cual no requiere librerías o herramientas adicionales [12]. Permite crear servicios web basados en la arquitectura REST.

1.4.5. Docker

Es una plataforma de código abierto que permite desplegar aplicaciones en contenedores de software. Aislando recursos del kernel de Linux, Docker empaqueta en estos contenedores librerías, herramientas del sistema, código, entre otros, para la ejecución de aplicaciones [13]. Se usará Docker para crear una imagen y un contenedor para el despliegue de la aplicación web.

1.4.6. Ionic

Es un framework basado en Angular para la creación de aplicaciones híbridas para móviles y tablets [14]. El conjunto de tecnologías que usa está basado en HTML, CSS y JavaScript [15]. Se usará Ionic 2 para la creación de la aplicación móvil que permita capturar al usuario una imagen con la cámara integrada para posteriormente enviarla a un servicio web. La plataforma móvil mínima será Android 4.4 y versiones superiores.

1.4.7. TypeScript

Es un lenguaje de programación de código abierto mantenido y desarrollado por Microsoft. Provee un superconjunto de JavaScript, agregando tipos estáticos y funciones de programación orientada a objetos. El código de TypeScript se transpila [1] a código JavaScript nativo. Se usará este lenguaje en el desarrollo de la aplicación móvil con Ionic 2, debido a su soporte por parte de este framework.

1.4.8. Amazon Web Services

Es una plataforma de servicios en la nube que permite usar potencia de cómputo para permitir el desarrollo de infraestructura informática de las empresas en la nube [16]. Provee un servicio llamado Amazon EC2, el cual entrega servidores virtuales con capacidad de cómputo variable [17]. Uno de los tipos de instancias de EC2 es la instancia P2, la cual proporciona capacidad informática paralela con tecnología GPU [18]. Se usará una

[1] Transpilar: del inglés transpiler, significa convertir el código fuente de un lenguaje de programación a otro.

instancia P2 para el preprocesamiento de imágenes y para la ejecución de las tareas de entrenamiento, validación y test de la red neuronal.

Otro tipo de instancia es T2, la cual proporciona un nivel de desempeño de CPU alto cuando la carga de trabajo lo exija [19]. Se usará una instancia T2 de Amazon EC2 para el despliegue y ejecución del contenedor de Docker que contendrá la aplicación web que proveerá el servicio web para la clasificación de imágenes.

2. METODOLOGÍA

En esta sección se realiza un análisis del tipo de datos con los que se creará el modelo de clasificación, se describen los pasos de construcción de la red neuronal convolucional, los pasos de entrenamiento, validación y pruebas, así como el servicio web y la aplicación móvil para usuarios finales con los cuales podrán clasificar imágenes de melanoma o nevus usando el modelo generado la red neuronal convolucional.

2.1. Obtención del conjunto de imágenes

Se debe obtener un conjunto de imágenes que garantice la calidad de la fuente de las imágenes etiquetadas como melanoma y nevus para su uso en la construcción del modelo que permita clasificar este tipo de imágenes. Las siguientes definiciones médicas dentro del campo de la dermatología proporcionarán una guía para determinar la calidad de las fuentes de las imágenes a usarse en este trabajo [20]:

Melanocitos: producen el pigmento llamado melanina, el cual provoca que la piel tenga un color más oscuro. Este pigmento sirve para proteger las capas más profundas de la piel contra ciertos efectos nocivos del sol. Estas son las células que se pueden convertir en melanoma, cuando comienzan a crecer de forma descontrolada.

Nevus: conocido vulgarmente como lunar, es un tumor benigno de la piel que se origina a partir de los melanocitos. Dependiendo del tipo, puede aumentar el riesgo de melanoma con el paso del tiempo. Tomando en cuenta el tipo y la etapa de crecimiento, los médicos suelen tener dificultades para distinguir un nevus de un melanoma. Por precaución este tipo de nevus suele eliminarse mediante biopsia.

Melanoma: es un cáncer que se origina en los melanocitos debido a mutaciones genéticas causadas por la exposición a los rayos ultravioleta [21]. Suele conocerse como melanoma maligno o melanoma cutáneo. El melanoma puede ocurrir en cualquier parte de la piel, siendo lo común su aparición en los lugares de la piel que son expuestos al sol. De no ser detectado a tiempo puede atravesar las capas profundas de la piel causando metástasis.

Lesión melanocítica: lesión provocada por el cambio en el crecimiento de los melanocitos debido a factores genéticos y ambientales [22].

Dermatoscopía: es una técnica de diagnóstico no invasiva la cual mejora la exactitud en el diagnóstico de lesiones pigmentadas en comparación con el examen a simple vista. Se utiliza un aparato llamado dermatoscopio, el cual consta de una lente de aumento y un

sistema de luz polarizada que se proyecta de forma constante y uniforme sobre la lesión a analizar. Este sistema de luz elimina la distorsión originada por la reflexión y refracción de la piel, lo que permite analizar ciertas estructuras dermatoscópicas que no se pueden observar a simple vista [23].

Los médicos suelen tener problemas al clasificar el tipo de tumor si encuentran un tumor en etapa temprana de crecimiento, realizando en ese caso un proceso de biopsia para remover el tumor y descartar la aparición de melanoma luego de los análisis clínicos necesarios [24]. Para conseguir un modelo de clasificación confiable se debe conseguir imágenes de fuentes que garanticen la correcta clasificación de imágenes dermatoscópicas. El conjunto de imágenes a usarse es “ISIC Archive”, siendo tomado de centros médicos y creado por dermatólogos con experiencia. El conjunto de imágenes “ISIC Archive” contiene imágenes clínicas de la piel e imágenes dermatoscópicas. Dado que los médicos utilizan imágenes dermatoscópicas obtenidas a través de un dermatoscopio como técnica no invasiva para los diagnósticos clínicos [25], se usará este tipo de imágenes para construir el clasificador.

2.1.1. ISIC Archive

ISIC (International Skin Imaging Collaboration) es una asociación académica e industrial creada para facilitar el uso de imágenes digitales de la piel para ayudar a disminuir la mortalidad relacionada por el melanoma. La falta de estándares para la obtención de imágenes dermatoscópicas de calidad para su uso por parte de personas dedicadas a la enseñanza médica o para el desarrollo y prueba de sistemas de diagnóstico automatizado es la razón de ser de esta asociación. Ha desarrollado y expandido un archivo de acceso público de código abierto llamado “ISIC Archive”, el cual contiene imágenes clínicas y dermatoscópicas de lesiones cutáneas (Figura 1), así como la propuesta de creación de categorías de estándares para imágenes digitales de la piel. Las categorías propuestas son [26]:

- **Calidad**: contar con imágenes que cumplan parámetros como resolución espacial (la capacidad de distinguir estructuras de tamaño pequeño), exactitud de color (obtención del color más parecido al real), claridad de enfoque, entre otros.
- **Privacidad**: asegurar la privacidad de los datos de los pacientes, tanto de las imágenes dermatoscópicas como la información de salud personal asociada con dichas imágenes.

- **Interoperabilidad**: estándares para asegurar que las imágenes y sus metadatos asociados puedan compartirse sin problemas entre sistemas.

Para crear este conjunto de imágenes de lesiones cutáneas, ISIC ha recurrido a imágenes obtenidas por la comunidad internacional de dermatólogos que muestran interés en el desarrollo de estándares para la imagenología[2] cutánea.

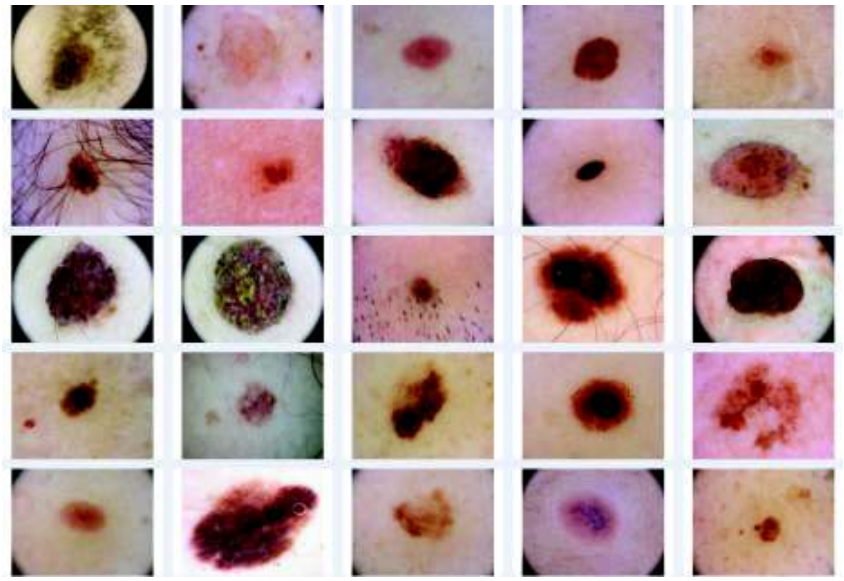

Figura 1: ISIC Archive

ISIC Archive contiene a su vez otros conjuntos de imágenes que han sido entregados a ISIC para su uso. Para el presente trabajo se usaron los siguientes conjuntos:

- HAM10000
- MSK-1
- MSK-2
- MSK-3
- MSK-4
- SONIC

[2] Imagenología: conjunto de procedimientos y técnicas para obtener imágenes del cuerpo humano para su uso en el trabajo científico o clínico.

- UDA-1
- UDA-2

Dentro de ISIC Archive, se pudo filtrar estos conjuntos de datos de acuerdo con los siguientes criterios:

Atributos de diagnóstico

- **Diagnóstico de lesión**: melanoma y nevus.

Atributos clínicos

- **Tipo de diagnóstico**: histopatológico.
- **Lesión melanocítica:** si

Atributos tecnológicos

- **Tipo de imagen**: dermatoscópica.

2.2. Métodos médicos de diagnóstico de lesiones melanocíticas

Existen algunos métodos médicos para realizar el diagnóstico diferencial del melanoma en base a las características de las lesiones cutáneas. En estos métodos, se puede realizar un diagnóstico de melanoma si se cumplen algunas o todas las reglas presentes en cada uno de los métodos (depende del método usado). El diagnóstico de nevus se basa en el no cumplimiento de las reglas de estos métodos. Se revisa a continuación el método ABCD, el método de los 7 puntos y el método de Menzies, los cuales han logrado una alta repercusión en la literatura médica [27]. Sin embargo, el diagnóstico final suele realizarse mediante la biopsia de la lesión y la obtención del resultado histopatológico.

2.2.1. Método ABCD

Es un conjunto de señales de advertencia que permiten ayudar a detectar cambios en posibles lesiones de melanoma. Un melanoma usualmente inicia con un cambio de forma, color o textura. Estas señales son [28]:

- **Asimetría**: una sección no es igual a otra, si se divide la lesión con dos ejes. Se valora asimetría respecto al color, estructura o forma.

- **Borde**: los bordes son irregulares o no se pueden definir con claridad. Se divide a la lesión en 8 segmentos y se realizan las comparaciones.
- **Color**: el color varía en diferentes zonas de la lesión y puede incluir tonos de color negro, marrón claro, marrón oscuro, rojo, blanco o azul.
- **Diámetro (Estructuras dermatoscópicas)**: hay cambios en el tamaño, el melanoma usualmente supera los 6 mm. Se comprueba la existencia de estructuras dermatoscópicas como retículos pigmentados, áreas desestructuradas, glóbulos, ramificaciones lineales y puntos.

Adicionalmente se agrega (si el diagnóstico médico lo permite) una señal adicional llamada evolución, para comparar los cambios en una lesión de la piel en un lapso determinado de tiempo.

2.2.2. Método de los 7 puntos

En este método se realiza una ponderación de los criterios dermatoscópicos. Si la puntuación es mayor a 3, se diagnosticará una lesión como melanoma con una sensibilidad del 95% y una especificidad del 75% [29]. Los criterios dermatoscópicos son:

- **Retículo pigmentado atípico**: retículo de color gris, negro o marrón.
- **Velo azul-blanquecino**: área de color azul cubierta por una capa blanca.
- **Patrón vascular atípico**: vasos lineales irregulares o vasos puntiformes.
- **Proyecciones irregulares**: proyecciones de la lesión de forma irregular en las partes externas de la lesión.
- **Puntos o globos irregulares**: tienen tamaños diferentes y se distribuyen de manera irregular en la lesión.
- **Manchas pigmentarias irregulares**: áreas irregulares de color gris, negro o marrón.
- **Estructuras de regresión**: abarcan áreas de color azul-gris y áreas de despigmentación.

2.2.3. Método de Menzies

Este método se basa en 11 criterios dermatoscópicos (Tabla 1), los cuales a su vez se dividen en dos criterios negativos, los cuales no deben presentarse en la lesión. También

se evalúan nueve criterios positivos de los cuales al menos uno debe presentarse en la lesión para diagnosticar una lesión melanocítica como melanoma [30].

Criterios negativos (ninguno debe estar presente)	**Criterios positivos (al menos uno debe estar presente)**
Simetría de patrones dermatoscópicos	Velo azul-blanquecino
Monocromía	Despigmentación pseudocicatricial
	Múltiples colores
	Retículo pigmentado prominente
	Pseudópodos
	Proyecciones radiales
	Múltiples puntos marrones
	Puntos/Glóbulos periféricos de color negro
	Puntos múltiples de color azul-gris

Tabla 1: Criterios dermatoscópicos del método de Menzies

2.2.4. Análisis de los métodos médicos

Como se puede comprobar en las reglas de cada método, existen patrones comunes que se repiten en los tres métodos, como son la presencia de elementos asimétricos, crecimientos anormales dentro y fuera de la lesión; y crecimientos de tamaño variable dentro de la lesión. Pero el patrón más repetido en estos métodos es la presencia de diferentes colores en diferentes estructuras dermatoscópicas, por lo que este patrón es el que menos debe cambiar al momento de aplicar cualquier técnica de preprocesamiento de imágenes.

2.3. Preprocesamiento de imágenes

El preprocesamiento de imágenes permite adaptar y mejorar las imágenes de origen a los parámetros necesarios para su uso, en este caso, de una red neuronal convolucional. De acuerdo con lo analizado en la sección 2.2.4, podemos aplicar algunas técnicas de preprocesamiento de imágenes para aumentar la cantidad de datos para ejecutar las tareas de creación del clasificador y reducir su error de generalización.

2.3.1. Aumento de datos

Se usa la técnica del aumento de los datos para aumentar el número de muestras con las que entrenaremos al clasificador y así compensar el costo de recolectar datos adicionales. Luego de aplicar las técnicas de preprocesamiento de imágenes, obtendremos nuevas imágenes acordes a las características representativas de cada clase [31]. Para el aumento de datos, tenemos dos opciones que son [32]:

Aumento fuera de línea: los datos se aumentan antes de ser usados. Se aplican técnicas de preprocesamiento de imágenes y se aumenta el conocimiento antes de iniciar las etapas de entrenamiento, validación y pruebas. Se debe usar cuando el conjunto de datos es pequeño.

Aumento en línea: el aumento se hace en el momento de ejecutar las etapas de entrenamiento, validación y pruebas. Mediante el uso de mini lotes, se realizan las trasformaciones en tiempo de ejecución. Se debe usar cuando se tiene un conjunto de datos muy grande, puesto que la lectura de tanta información puede llegar a crear una latencia notoria que perjudique la creación del modelo en términos de tiempo.

Para este proyecto se usará el aumento fuera de línea debido a que el conjunto de datos es pequeño, además de que la cantidad de imágenes de melanoma es baja con respecto a la cantidad de imágenes de nevus, por lo que se requiere igualar la cantidad de datos de estas dos clases de imágenes.

2.3.2. Técnicas de preprocesamiento de imágenes a aplicarse

Se aplicarán 4 técnicas de preprocesamiento de imágenes al conjunto de imágenes final, las cuales son [33]:

Rotación: al rotar las imágenes en un cierto número de grados se pueden generar variantes tanto en asimetría como en borde, puesto que, al ser los melanomas irregulares en estas dos características, mediante el uso de la técnica anterior se puede proveer de más información a la red neuronal convolucional para mejorar el porcentaje de clasificación.

En la operación de rotación se realiza una transformación geométrica para convertir la posición (x_1, y_1) de cada elemento de una imagen de entrada en una posición (x_2, y_2) de un elemento de una imagen de salida al rotarla mediante un ángulo θ sobre un origen O.

Si el valor de salida se encuentra fuera de los límites de la imagen, se ignora. La manera en la que se realiza una rotación es la siguiente:

$$x_2 = \cos(\theta)(x_1 - x_0) - \sin(\theta)\,(y_1 - y_0) + x_0$$

$$y_2 = \sin(\theta)(x_1 - x_0) + \cos(\theta)\,(y_1 - y_0) + y_0$$

donde (x_0, y_0) son las coordenadas del centro u origen de la imagen de entrada y θ es un ángulo con un valor positivo (en el sentido de las manecillas del reloj).

Traslación: al mover las imágenes dentro de los valores de alto y ancho que las contienen, se logra que la red neuronal pueda clasificar imágenes en las cuales la lesión no se encuentre precisamente en el centro de la imagen.

En la operación de traslación se realiza una transformación geométrica para convertir la posición (x_1, y_1) de cada elemento de una imagen de entrada en una posición (x_2, y_2) de un elemento de una imagen de salida al desplazar cada elemento una distancia (β_x, β_y) definida previamente. Si el elemento desplazado se encuentra fuera de los límites de la imagen, se ignorará. Para rellenar las secciones vacías se usa el método de interpolación por el vecino más cercano. La operación de traslación es la siguiente:

$$x_2 = x_1 + \beta_x$$

$$y_2 = y_1 + \beta_y$$

Zoom (enfocar): se usa para proveer información a la red neuronal sobre imágenes que contienen lesiones más cerca o más lejos de lo que usualmente se aprecia en las imágenes dermatoscópicas. Cuando se realiza zoom a una imagen, podemos reducir o ampliar el tamaño de una imagen. La reducción del tamaño de una imagen de entrada se realiza mediante la sustitución de un grupo de píxeles por uno solo elegido de manera aleatoria dentro de ese grupo. La ampliación del tamaño de una imagen de entrada se realiza mediante una replicación de píxeles. En ambos casos podemos convertir la posición (x_1, y_1) de cada elemento de una imagen de entrada en una posición (x_2, y_2) de un elemento de una imagen de salida al escalar cada elemento en un factor α. La operación es la siguiente:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & 1/\alpha \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

Dado que la red neuronal convolucional a usar tiene una entrada fija, una vez realizado el zoom a la imagen, esta debe modificarse para adaptarse al tamaño de entrada de esa red.

Escala: se usa para proveer a la red neuronal imágenes que se adapten al tamaño de la capa de entrada. La operación de escala implica cambiar las dimensiones de la imagen. Se realiza una transformación geométrica para convertir todos los píxeles de una imagen de entrada (A) de una imagen de entrada en un nuevo conjunto de píxeles (B) de una imagen de salida al escalar cada elemento un factor s definido previamente. Si el factor s es mayor que 1, implica un aumento en las dimensiones de la imagen original, conteniendo un número mayor de píxeles. Si el factor s es menor que 1, de la forma $\frac{1}{n}$, donde n es un entero, implica una disminución en las dimensiones de la imagen original, reteniendo información cada n filas y columnas de la imagen original. Se calcula el nuevo ancho (x) y alto (y) de la imagen de salida, multiplicando estas dimensiones por el factor de escala s. La operación de escala es la siguiente (al final se debe agregar un paso de interpolación):

$$x_2 = sx_1$$

$$y_2 = sy_1$$

$$Algoritmo\ de\ interpolación$$

2.4. Redes Neuronales Artificiales

Las Redes Neuronales Artificiales son un modelo computacional basado en una gran cantidad de unidades de cómputo simple que se encuentran interconectadas, permitiendo crear sistemas que aprenden, destacando en las áreas en las cuales la programación de soluciones de detección de características específicas es muy difícil por la complejidad, tiempo y esfuerzo que se emplearían. Una red neuronal artificial tiene su analogía con el cerebro de los mamíferos, dado que las neuronas almacenan el conocimiento basado en la experiencia. Este conocimiento se adquiere bajo un proceso de aprendizaje, en el cual se muestra a la red neuronal lo que se desea obtener de salida dada cierta entrada, hasta

que esta es capaz de acumular el suficiente conocimiento para inferir la salida correcta [34].

Como se indica en la Figura 2, las neuronas artificiales se pueden conectar unas a otras y disponerse creando una estructura conocida como arquitectura de red neuronal. Estas neuronas se suelen ubicar en capas. Al juntar varias capas se logra conformar una red neuronal artificial. Estas capas son las que permiten resolver problemas no lineales.

Una red neuronal artificial feed-forward de tres capas tiene la cualidad de funcionar como aproximador universal, esto es, dado una cantidad suficiente de neuronas en las capas ocultas, una red neuronal multicapa puede aproximar cualquier función continua.

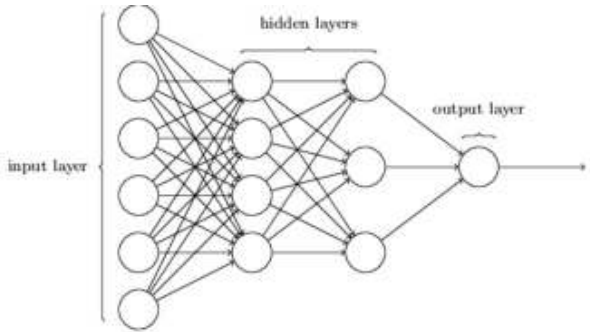

Figura 2: Red Neuronal Artificial con dos capas ocultas. Fuente: Neural Networks and Deep Learning

Este resultado proviene del teorema de aproximación universal [35]. Sin embargo, este teorema no señala la manera de alcanzar la exactitud deseada, ni cómo ajustar ciertos parámetros como el número de capas ocultas o el número de neuronas en las capas ocultas. Estos valores suelen determinarse usando un procedimiento de prueba y error.

2.5. Redes Neuronales Convolucionales

Una red neuronal convolucional funciona detectando información sencilla en sus primeras capas [36], para posteriormente combinar esta información en las capas profundas de la red [37]. El origen de las redes neuronales convolucionales radica en su similitud con la corteza visual primaria de los cerebros biológicos [38]. En un cerebro biológico, existen neuronas especializadas en la detección de patrones simples agrupadas en campos receptivos. La activación por parte de un estímulo a esta corteza visual se propaga a través de estos campos receptivos, especializados en tareas simples como detección de líneas y

bordes, y posteriormente a los campos de tareas complejas como rotación del estímulo visual, en las cuales las neuronas necesarias se activan.

La estructura de una red neuronal convolucional típica consta de los siguientes tipos de capas [39] y algoritmos [40]:

Capa convolucional: la convolución es una operación en la cual se recibe una imagen de entrada y se aplica posteriormente un filtro o "kernel" para obtener un mapa de características. La convolución provee a su vez la reducción de los parámetros de la imagen de entrada debido al menor tamaño del filtro aplicado comparado con la entrada original. En la Figura 3 se puede apreciar una operación de convolución:

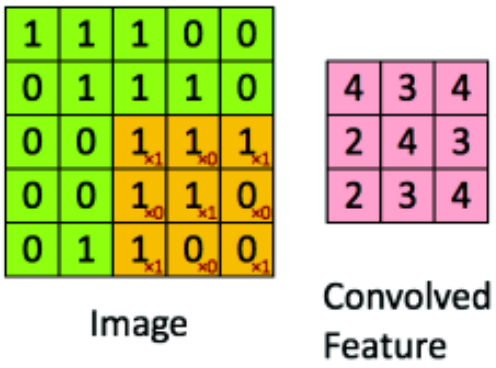

Figura 3: Operación de convolución. Fuente: The Data Science Blog

El tamaño del mapa de características se maneja por tres parámetros que son:

- **Profundidad**: se refiere al número de filtros aplicados en la misma capa convolucional.
- **Paso o "Stride":** es el número de píxeles con los que deslizamos el filtro o kernel sobre la matriz de la imagen de entrada. Por ejemplo, si nos movemos 2 píxeles a la vez, el paso o "stride" tiene un valor de 2.
- **Zero-padding**: significa rellenar la matriz de la imagen de entrada con ceros alrededor del borde de esta, de tal manera que no existan errores de cálculo al aplicar el filtro a los píxeles que forman el borde.

Luego de cada operación de convolución se suele aplicar una operación adicional llamada ReLU (Rectified Linear Unit). Esta es una operación no lineal y está dada por:

$$Output = \max(0, Input)$$

ReLU se aplica en cada píxel y reemplaza todos los píxeles con valores negativos por valores iguales a 0 [41]. El propósito de ReLU es introducir la no linealidad en la red neuronal convolucional, debido a que las operaciones de convolución son operaciones lineales. Investigaciones sobre funciones de activación indican que ReLU es mucho más rápida para el entrenamiento de redes profundas [42]. El algoritmo para que una capa convolucional pueda generar un volumen de salida a partir de un volumen de entrada es el siguiente:

Dado un volumen de entrada de tamaño $W_1 \times H_1 \times D_1$ donde W es el ancho, H es el alto y D es la profundidad; usando los hiperparámetros K (número de filtros), F (tamaño del campo receptivo o filtro), S (valor del stride) y P (valor del zero padding), se produce un volumen de salida de tamaño $W_2 \times H_2 \times D_2$ donde:

$$W_2 = \frac{(W_1 - F + 2P)}{S + 1}$$

$$H_2 = \frac{(H_1 - F + 2P)}{S + 1}$$

$$D_2 = K$$

Capa de pooling: se coloca luego de la capa convolucional. Su uso es para reducir las dimensiones de alto y ancho de su entrada para la siguiente capa convolucional. La operación que se usa en este proyecto es max-pooling. Esta operación divide a la entrada en rectángulos o cuadrados según sea el caso y, de acuerdo con cada región generada, crea una salida en la cual se incluye el máximo valor de cada región, como se puede apreciar en la Figura 4:

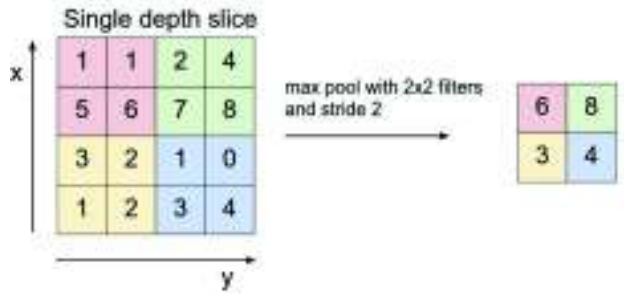

Figura 4: Operación max-pooling. Fuente: CS231 Stanford University Course

El algoritmo para que una capa de pooling pueda generar un volumen de salida a partir de un volumen de entrada es el siguiente:

Dado un volumen de entrada de tamaño $W_1 \times H_1 \times D_1$ donde W es el ancho, H es el alto y D es la profundidad; usando los hiperparámetros F (tamaño del campo receptivo o filtro) y S (valor del stride), se produce un volumen de salida de tamaño $W_2 \times H_2 \times D_2$ donde:

$$W_2 = \frac{(W_1 - F)}{S + 1}$$

$$H_2 = \frac{(H_1 - F)}{S + 1}$$

$$D_2 = D_1$$

Capa totalmente conectada: al final de todas las capas de la red, se coloca esta capa, la cual tendrá tantas neuronas como número de clases a predecirse.

Finalmente, en la Figura 5 se puede apreciar una estructura mínima de una red neuronal convolucional con las capas descritas anteriormente:

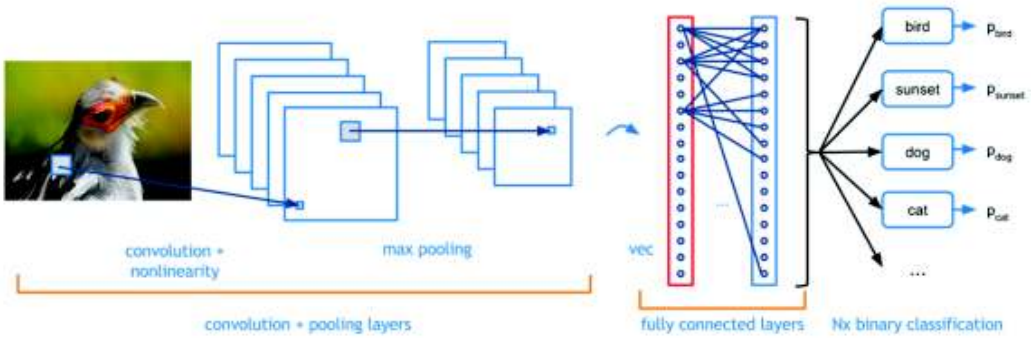

Figura 5: Red Neuronal Convolucional. Fuente: Adit Deshpande

En esta última capa se usa la función softmax, la cual "comprime" los elementos de un vector en valores en el rango [0, 1]. Esta función permite representar las salidas de la red neuronal convolucional como una distribución de probabilidad [43]. La función softmax se define como:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

para todo $j = 1,2, \ldots K$. La función softmax normaliza un vector z de valores reales a un vector $\sigma(z)$ donde todos sus componentes suman 1, convirtiéndolo en un vector de probabilidades.

2.6. Arquitectura Inception v3

La arquitectura de red neuronal GoogLeNet V3, también conocida como Inception v3 (Figura 6), se creó en base a la arquitectura GoogLeNet, la cual fue la ganadora del concurso ILSVRC 2014 con un 6.67% de error en Top-5. El concurso ILSVRC (ImageNet Large Scale Visual Recognition Challenge) evalúa los algoritmos para la detección de objetos y la clasificación de imágenes a gran escala. Una motivación de alto nivel de este concurso es permitir a los investigadores comparar el progreso en la detección en una gran variedad de objetos [44]. Inception v3 alcanzó un porcentaje de error de 17.2% en Top-1 y de 3,58% en Top-5 [45]. Esta red fue entrenada con el conjunto de imágenes ImageNet del 2012, el cual contiene imágenes etiquetadas en 1000 clases. Los autores utilizaron 1.2 millones de imágenes para el entrenamiento, 50 000 para validación y 100 000 para pruebas del conjunto de imágenes ImageNet [46].

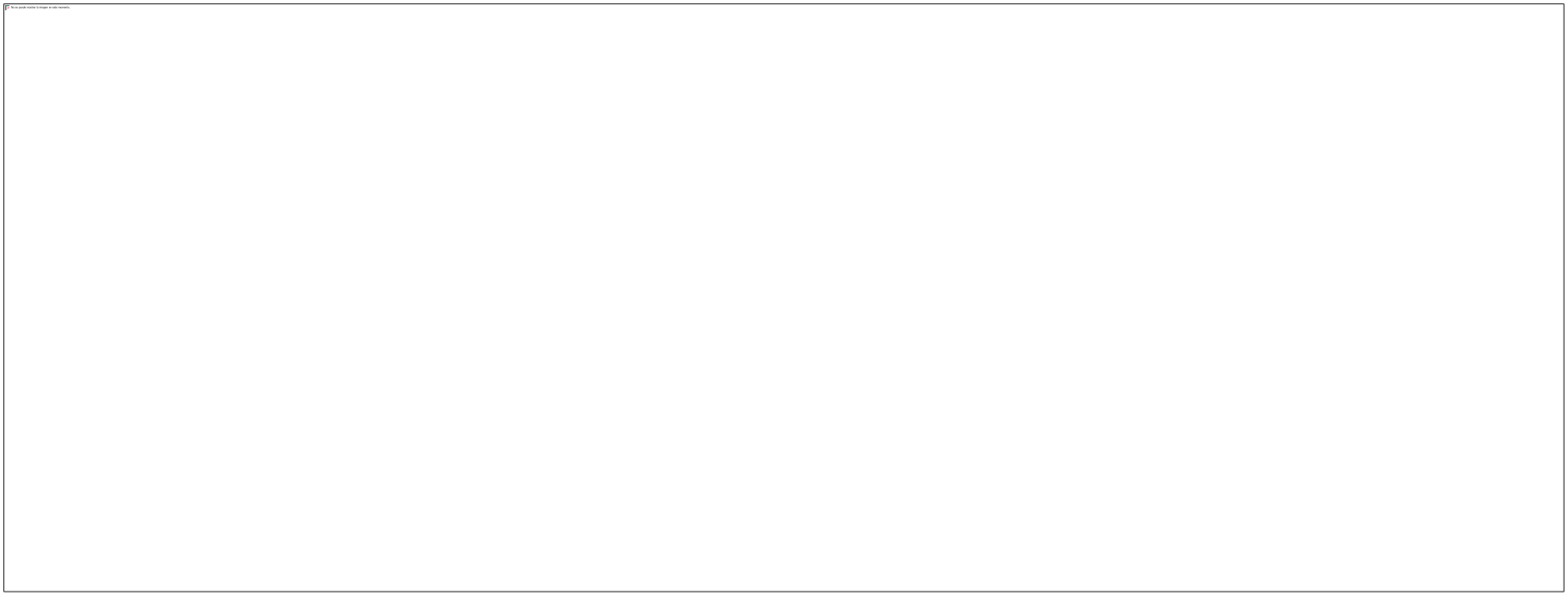

Figura 6: Arquitectura Inception-v3. Fuente: Initialized Capital

Esta red neuronal convolucional será la que se use en el presente trabajo como base para plantear una arquitectura derivada de la arquitectura original enfocada al uso de la técnica de transferencia de aprendizaje (sección 2.7). Con esta técnica, se reutilizarán los pesos de un modelo previamente entrenado y solo se entrenará la última capa totalmente conectada, adaptada a las clases de melanoma y nevus. Dado que la red Inception v3 es una evolución de las versiones 1 y 2, se indica a continuación las características de las 3 versiones y las mejoras aportadas en cada una de ellas.

Inception v1

Para crear un clasificador de objetos, los autores de Inception partieron de las siguientes premisas:

- Las partes relevantes de una imagen pueden llegar a tener una variación de tamaño relativamente grande. Por ejemplo, si queremos detectar un vehículo en una imagen, este objeto puede ocupar toda la imagen o parte de ella.
- Dada esta variación de tamaño, seleccionar el tamaño correcto del filtro o kernel para una operación de convolución se convierte en una tarea compleja. Se requiere usar un filtro grande para la información distribuida de manera global y se requiere usar un filtro pequeño para la información distribuida de manera local.
- Las redes neuronales profundas (con muchas capas) son propensas al fenómeno de sobreajuste u overfitting. Además, la actualización de los pesos de las neuronas en el proceso de entrenamiento se convierte en un problema.
- El apilar capas y operaciones de convolución se vuelve computacionalmente costoso.

Las arquitecturas Inception usan un módulo llamado “Inception” que paraleliza ciertas operaciones de convolución para finalmente concatenar los resultados de cada una de esas operaciones en una nueva matriz. La versión inicial presentada por los autores se llama “Naive Inception Module”, como se muestra en la Figura 7:

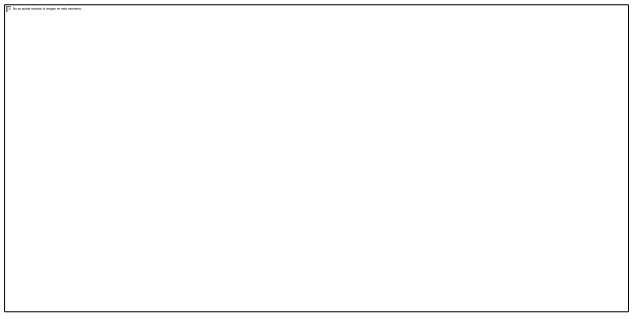

Figura 7: Naive Inception Module

En este módulo se realizan operaciones de convolución con tres tamaños diferentes de filtros (1x1, 3x3 y 5x5).

Adicionalmente se realiza una operación de max pooling. Las salidas de estas cuatro operaciones se concatenan, creando una nueva matriz para alojar los valores de las salidas y se envían al siguiente módulo Inception. Como todas estas operaciones son costosas computacionalmente, los autores limitan la dimensionalidad de las entradas al agregar operaciones de convolución de 1x1 antes de las convoluciones restantes, como se muestra en la Figura 8. Las convoluciones de tamaño 1x1 son muy útiles para reducir la profundidad del volumen de entrada. Por ejemplo, si tuviésemos una entrada de tamaño 200x200x40, siendo 40 la profundidad del volumen; al aplicar 5 filtros de convolución de tamaño 1x1x40, podemos reducir la profundidad del volumen de la entrada a 5 puesto que, al aplicar los 5 filtros, la salida es de tamaño 200x200x5. Posteriormente se pueden aplicar filtros más grandes dentro del módulo con entradas relativamente pequeñas [47].

Esta forma de organizar las operaciones permite que estos filtros grandes cubran un gran campo receptivo de la entrada para extraer su información. Finalmente, se obtiene un beneficio al disminuir el tamaño espacial de la entrada y el sobreajuste.

Figura 8: Módulo Inception con reducción de dimensionalidad

La reducción de la dimensionalidad es la que permite disminuir el número de operaciones a realizarse. Esto soluciona la aparente contradicción de un mayor uso de recursos computacionales al agregar más operaciones de convolución a las ya existentes.

Siendo una red bastante profunda, está sujeta al problema de desvanecimiento del gradiente. Para evitar este inconveniente, se introdujeron dos clasificadores auxiliares en esta versión de la arquitectura. Básicamente se aplica softmax a las salidas de dos de los

módulos de inicio señalados en la Figura 9 y se calcula una pérdida auxiliar sobre las mismas etiquetas. La función de pérdida total es una suma ponderada de las pérdidas auxiliares y la pérdida real al final de la red.

$$Total\ Loss = Real\ Loss + 0.3(Aux\ Loss\ 1) + 0.3(Aux\ Loss\ 2)$$

El valor utilizado en el artículo de los autores es de 0.3 para cada pérdida auxiliar. Estas pérdidas auxiliares son útiles para el proceso de entrenamiento, puesto que evitan el problema antes descrito en las secciones medias de la red neuronal. Los valores de las pérdidas se ignoran al momento de inferir un nuevo dato.

Figura 9: Inception v1 con pérdidas auxiliares. Fuente: Caylin Hickey

Inception v2

En un nuevo trabajo, los autores propusieron una serie de mejoras que aumentaron la exactitud del modelo y disminuyeron el costo computacional de las operaciones dentro de la red. Se basaron en [45]:

- Reducir el cuello de botella representacional (bottleneck). En una red neuronal convolucional típica, el cuello de botella se refiere a la última capa de la red antes de la capa totalmente conectada. En el caso de la arquitectura Inception, el cuello de botella representacional se refiere a la salida de cada módulo Inception. Dado que reducir demasiado la dimensionalidad puede llevar a la pérdida de información, el reducir el cuello de botella mediante el uso de convoluciones que no alteren drásticamente las dimensiones de las entradas de los módulos Inception evita este problema.
- Factorización de operaciones de convolución para que sean más eficientes en términos de complejidad computacional. La factorización implica separar una operación de convolución en varias.

Las soluciones propuestas por los autores fueron:

Factorizar la convolución de 5x5 en dos operaciones de convolución de 3x3 para aumentar la velocidad de la ejecución de los cálculos requeridos. Una operación de convolución de 5x5 es 2.78 veces más costosa en tiempo que una operación de convolución de 3x3. En lugar de tener una única convolución de 5x5, se apilan dos convoluciones de 3x3, mejorando el rendimiento. El módulo Inception correspondiente se puede apreciar en la Figura 10:

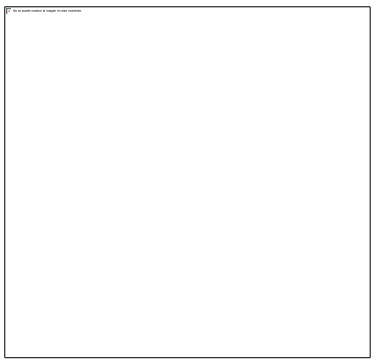

Figura 10: Módulo Inception A con convoluciones de 3x3 apiladas

Adicionalmente, los autores factorizan las convoluciones del tamaño de filtro nxn a una combinación de convoluciones 1xn y nx1. Para el caso de la convolución 3x3, esto equivale a realizar una convolución de 1x3 y posteriormente realizar una convolución de 3x1. Se encontró que esta combinación de convoluciones es un 33% más económico computacionalmente hablando, que la convolución de 3x3. La Figura 11 ilustra lo mencionado:

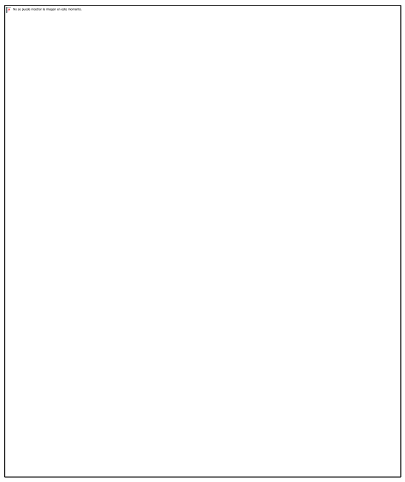

Figura 11: Módulo Inception B con factorización de convoluciones nxn

También hubo cambios con respecto a los filtros, en donde los filtros en cada módulo Inception se expandieron a lo ancho para evitar el problema del cuello de botella representacional. Un filtro grande a lo ancho permite obtener más información al abarcar un mayor campo receptivo. Un filtro grande en profundidad provoca una reducción excesiva de la dimensionalidad y por consecuente una pérdida de información. El módulo con filtros ampliados se muestra en la Figura 12:

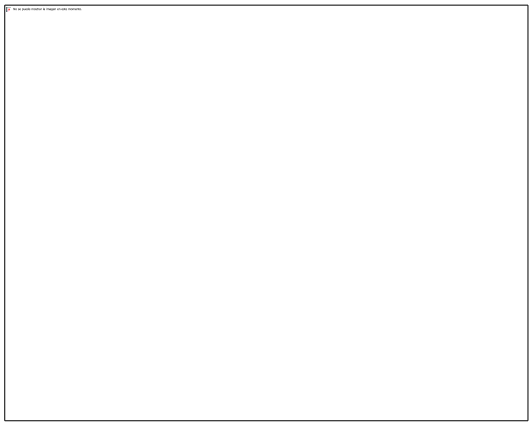

Figura 12: Módulo Inception C con filtros amplios

Los cambios anteriores derivaron en la creación de tres tipos diferentes de módulos Inception. La arquitectura correspondiente a Inception v2 es la siguiente (Tabla 2):

Type	Patch size/stride or remarks	Input size
conv	3x3/2	299x299x3
conv	3x3/1	149x149x32
conv padded	3x3/1	147x147x32
pool	3x3/2	147x147x64
conv	3x3/1	73x73x64
conv	3x3/2	71x71x80
conv	3x3/1	35x35x192
3xInception	Módulo Inception A	35x35x288
5xInception	Módulo Inception B	17x17x768
2xInception	Módulo Inception C	8x8x1280
pool	8x8	8x8x2048
linear	logits	1x1x2048

softmax	classifier	1x1x1000

Tabla 2: Esquema de la arquitectura Inception v2

Inception v3

Esta arquitectura parte de Inception v2 bajo las siguientes premisas:

- Los autores notaron que los clasificadores auxiliares no contribuyen mucho hasta casi el final del proceso de entrenamiento, cuando el problema de la fuga de gradiente iniciaba.
- Se buscó maneras de mejorar la arquitectura Inception v2 para evitar un cambio drástico en los tres tipos de módulos Inception que se utilizaban.

Finalmente decidieron utilizar solo un clasificador auxiliar cerca del final de la red. Además, se agregaron técnicas como el uso del optimizador RMSProp para la etapa de entrenamiento, convoluciones factorizadas de 7x7, BatchNorm para el clasificador auxiliar y Label Smooting. Esta arquitectura contiene 23 millones de parámetros y se basa en las dos arquitecturas descritas anteriormente.

2.7. Transferencia de aprendizaje

El éxito en la aplicación de redes neuronales convolucionales radica en que la clasificación de imágenes se basa en los siguientes factores:

- Existencia de una gran cantidad de datos (cientos de miles o millones).
- Gran capacidad de cálculo.

Existe inconvenientes cuando no se cumple al menos uno de estos factores, lo que impide usar una red neuronal convolucional de manera eficiente. La técnica de transferencia de aprendizaje permite aprovechar el conocimiento, de una red previamente entrenada bajo un dominio, en otro dominio. Esto funciona debido a que las capas inferiores de una red neuronal convolucional se especializan en la detección de características genéricas, por ejemplo, la detección de bordes o colores, las cuales son útiles para muchas tareas. Por otra parte, las capas superiores se especializan en detectar características más específicas de las clases del dominio presente en el conjunto de datos original [48] (Figura 13):

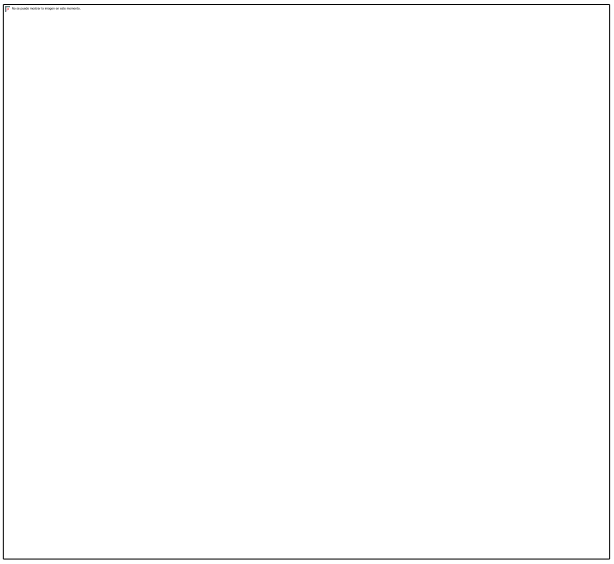

Figura 13: Transferencia de aprendizaje. Fuente: ResearchGate

Para dar una definición matemática de transferencia de aprendizaje, se revisarán las definiciones matemáticas de dominio y tarea. Un dominio $\mathcal{D}$ consiste en dos componentes [49]: un espacio de representaciones de todas las muestras $\mathcal{X}$ y una distribución de probabilidad marginal $P(X)$ donde $X = \{x_1, \dots, x_n\} \in \mathcal{X}$.

Para este proyecto, $\mathcal{X}$ es el espacio de todas las representaciones de las imágenes de melanoma y nevus, x_i es el i-ésimo vector de características correspondiente a alguna imagen de melanoma o nevus y X es el conjunto de imágenes de melanoma y nevus usadas para el entrenamiento.

Dado un dominio específico $\mathcal{D} = \{\mathcal{X}, P(X)\}$, una tarea $\mathcal{T}$ consiste en un espacio de etiquetas $\mathcal{Y}$ y una distribución de probabilidad condicional $P(Y|X)$ (formando la tupla $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$) el cual puede ser aprendido de los datos de entrenamiento, los cuales consisten en pares $(x_i \in X, y_i \in \mathcal{Y})$. Para este proyecto, $\mathcal{Y}$ es el conjunto de etiquetas $\{melanoma, nevus\}$ y y_i es cualquiera de estas etiquetas.

Dado un dominio de origen $\mathcal{D}_S$ y una tarea de origen $\mathcal{T}_S$, así como un dominio de destino $\mathcal{D}_T$ y una tarea de destino $\mathcal{T}_T$, el objetivo de la transferencia de aprendizaje es permitir aprender la distribución de probabilidad condicional de destino $P(Y_T|X_T)$ en el dominio de destino $\mathcal{D}_T$ con la información obtenida del dominio de origen $\mathcal{D}_S$ y la tarea de origen $\mathcal{T}_S$, donde $\mathcal{D}_S \neq \mathcal{D}_T$ o $\mathcal{T}_S \neq \mathcal{T}_T$. Para este proyecto, significa que entre un conjunto de imágenes de origen y un conjunto de imágenes de destino, las características son diferentes entre

los dos conjuntos, por ejemplo, son objetos diferentes, o sus distribuciones de probabilidad marginales son diferentes.

Dados los dominios de origen y destino $\mathcal{D}_S$ y $\mathcal{D}_T$, donde $\mathcal{D} = \{\mathcal{X}, P(X)\}$; y las tareas de origen y destino $\mathcal{T}_S$ y $\mathcal{T}_T$, donde $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$, las condiciones de origen y destino pueden variar de cuatro maneras:

- $\mathcal{X}_S \neq \mathcal{X}_T$: los espacios de características del dominio de origen y destino son diferentes.
- $P(X_S) \neq P(X_T)$: las distribuciones de probabilidad marginal de los dominios de origen y destino son diferentes.
- $\mathcal{Y}_S \neq \mathcal{Y}_T$: los espacios de etiquetas entre dos tareas son diferentes.
- $P(Y_S|X_S) \neq P(Y_T|X_T)$: las distribuciones de probabilidad condicional de las tareas de origen y destino son diferentes.

2.7.1. Estrategias para transferencia de aprendizaje

Debido a la diferencia de características entre las capas inferiores y superiores de una red neuronal convolucional, y tomando en cuenta el conjunto de datos de la red previamente entrenada y la similitud entre los datos de entrenamiento originales y los nuevos datos de entrenamiento, surgen cuatro escenarios [50] sobre los cuales se puede decidir una estrategia a usar (Tabla 3):

Tamaño del nuevo conjunto de datos	**Similitud con el conjunto de datos de la red entrenada**	**Estrategia**
Pequeño	Similar	Entrenar la capa totalmente conectada
Pequeño	Diferente	Remover capas superiores y entrenar capa totalmente conectada
Grande	Diferente	Entrenar desde cero, estableciendo los pesos a los mismos de la red entrenada.
Grande	Similar	Entrenar desde cero, estableciendo los pesos a los mismos de la red entrenada.

Tabla 3: Escenarios y estrategias para aplicar transferencia de aprendizaje

2.8. Diseño de la red neuronal convolucional

En esta sección se describe el diseño final de la arquitectura a ser usada, basada en la arquitectura Inception v3.

2.8.1. Tratamiento de datos

Para crear el conjunto de imágenes base del presente trabajo, tomando en cuenta los filtros de la sección 2.1.1, se obtuvo imágenes de melanoma y nevus mediante el uso de la API y los parámetros que provee ISIC, logrando un total de 1 019 imágenes etiquetadas como melanoma y 1 944 imágenes etiquetadas como nevus, dando un total de 2 963 imágenes en total.

Se utilizará el algoritmo de descenso de gradiente por mini lotes en el entrenamiento de la última capa totalmente conectada. El uso de este tipo de descenso de gradiente con lotes de 32 hasta 512 muestras es suficiente para minimizar de manera óptima la función de error [51]. Dado que se usarán tamaños de lote de 32, 64, 128 y 256 muestras para entrenamiento y validación, el número de imágenes a generar será un múltiplo de estos 4 valores. Esto permitirá que, en todas las iteraciones, se use un lote completo de acuerdo con el tamaño del lote definido.

De acuerdo al conjunto de imágenes obtenido de la API de ISIC y a las técnicas de preprocesamiento de imágenes que se indican en la sección 2.3.2; usando el lenguaje Python y el framework Keras, mediante la clase “Image Data Generator”, se creará un script capaz de leer las imágenes del conjunto de imágenes de melanoma y nevus, y se aplicarán todas estas técnicas para generar 25 600 imágenes por clase (múltiplo de 32, 64, 128 y 256). Esto implica utilizar la técnica de aumento de datos y partir de un conjunto de 2 963 imágenes para generar un nuevo conjunto de 51 200 imágenes (25 600 imágenes de melanoma y 25 600 imágenes de nevus) lo suficientemente grande como para poder ser utilizado sobre la última capa de la red neuronal convolucional Inception v3.

Este número específico de imágenes se debe a que se necesita un número alto de imágenes para la tarea de entrenamiento de la última capa totalmente conectada. Las características de la instancia P2 con el nombre “p2.xlarge” de Amazon EC2 permiten paralelizar la generación de las imágenes. Las características que tiene esta instancia son un GPU NVIDIA K80 con 4 992 núcleos CUDA, 4 virtual CPUs y 61 GiB (1 GiB equivale a 2^{30} bytes) de memoria RAM.

Se aplicarán los siguientes parámetros para cada técnica:

- **Rotación**: se rotará la imagen de 0 a 360 grados de manera aleatoria.
- **Traslación**: se desplazará la imagen en un porcentaje aleatorio de 0% a 20% a la izquierda y de 0% a 20% a la derecha del total del ancho de la imagen. Los mismos porcentajes se aplicarán al total de la altura de la imagen para desplazarla hacia arriba o hacia abajo.
- **Zoom**: se escalará la imagen hacia adentro y hacia afuera en un rango de 0% a 20% aplicado a la relación entre el ancho y el alto de la imagen.
- **Escala**: se redimensionará cada imagen (siendo sus dimensiones originales variables) para que coincida con los valores de alto y ancho de la capa de entrada de la red Inception v3. Para la altura y anchura se usan 299 píxeles en cada dimensión.

Un ejemplo de una imagen con aumento de datos se puede apreciar en la Figura 14:

Figura 14: Imagen y sus variaciones con aumento de datos

En esta etapa se realiza una normalización sobre las imágenes. Existe una técnica usada en Inception v3 conocida como normalización por lotes, en la cual se mejora el rendimiento y la estabilidad en el entrenamiento [52]. Por cada canal, se calcula la media y la desviación estándar asociada, tras lo cual se resta la media en cada píxel y se divide para la desviación estándar. Esto garantiza que cada parámetro de entrada tenga una distribución de datos similar y permite que la red neuronal aprenda más rápidamente.

2.8.2. Obtención de Vectores de características

Puesto que sólo entrenaremos la última capa totalmente conectada, debemos extraer los vectores de características de cada imagen del conjunto de imágenes final. Sin embargo, como las imágenes se reutilizarán varias veces en el entrenamiento, calcular los vectores de características cada vez que se utilicen lleva una considerable cantidad de tiempo, por lo que es necesario almacenar en el disco duro esta información para su posterior uso, con la ventaja del ahorro de tiempo [53].

Por cada imagen del conjunto de imágenes final:

- Se creará un identificador único basado en el nombre del archivo de imagen para crear el archivo que contendrá el vector de características.
- Se calculará el vector de características (2 048 elementos), realizando las diferentes operaciones en la red Inception v3 desde la capa de entrada hasta la capa antes de la última capa totalmente conectada.
- Se guardará el vector de características en el archivo creado para tal fin.

Para las tareas posteriores, simplemente se leerá cada archivo con el correspondiente vector de características y se utilizará dicha información para cualquiera de las tareas de entrenamiento, validación y pruebas.

En la capa de entrada se usa una imagen con tres canales RGB. El modelo cuenta con filtros para cada una de las capas, los cuales se convolucionan con la capa correspondiente y la salida será la concatenación de los resultados de las convoluciones.

El modelo previamente entrenado con la arquitectura Inception v3 proporciona las siguientes características [54]:

Bordes y colores: El modelo previamente entrenado usa filtros de bordes orientados en diferentes direcciones. Incluyen información de colores específicos.

Texturas: En el caso del modelo de Inception v3, los filtros de texturas se enfocan en detectar algunos patrones continuos muy simples como puntos, líneas y colores.

Patrones: A diferencia de las texturas, las capas más profundas se enfocan en patrones más complejos y específicos de las muestras de las clases con las cuales se entrenó el modelo, basados en características pequeñas tales como ruedas, ojos, etc.

Partes: Estos filtros permiten detectar secciones específicas de las clases tales como brazos, ropa, pelo, etc.

Objetos: Estos filtros permiten encontrar objetos específicos de las muestras de las clases, tales como perros, personas, etc.

Una muestra de las características usadas por Inception v3 se puede apreciar en la Figura 15:

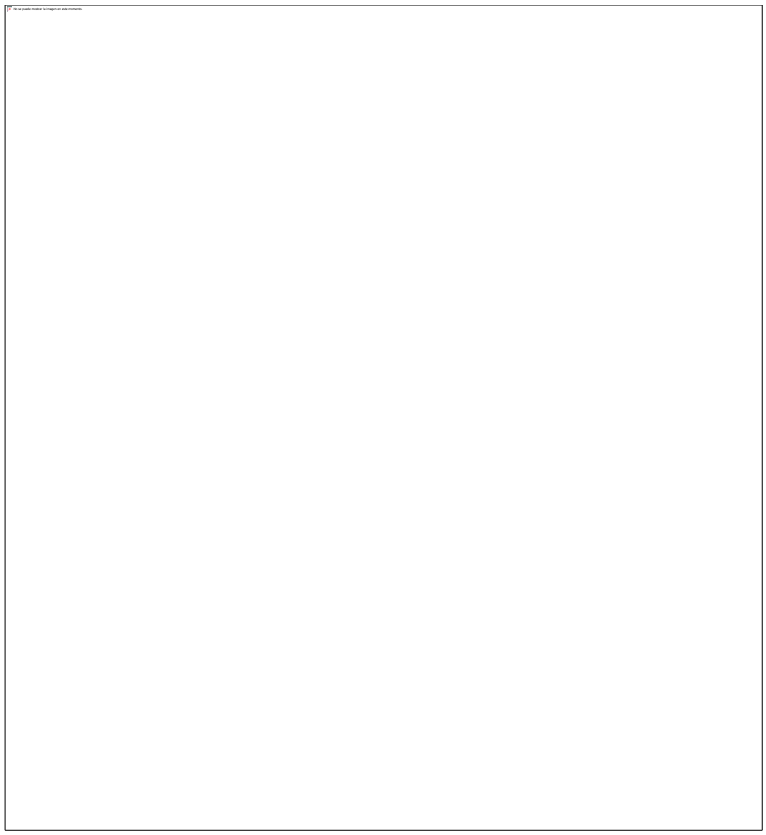

Figura 15: Visualización de características de Inception v3. Fuente: Distill

Los filtros de la capa antes de la última capa totalmente conectada, aplicados a una imagen, generan los siguientes resultados (Figura 16):

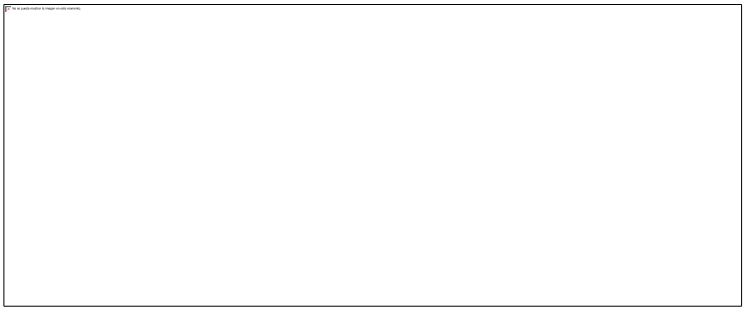

Figura 16: a) Imagen original y b) salida de los filtros de la última capa convolucional

2.8.3. Caso de transferencia de aprendizaje a usarse

El conjunto de imágenes ImageNet no contiene imágenes etiquetadas como melanoma o nevus y el número de imágenes obtenidas luego del preprocesamiento y aumento de datos (51 200 imágenes), no se acerca al número de imágenes usadas de ImageNet para entrenar la red Inception-v3, el cual es de 1.2 millones. Teniendo en cuenta estos precedentes, el caso a utilizar tomando en cuenta la sección 2.7.1 es el caso cuando el tamaño del nuevo conjunto de datos es pequeño y es diferente al conjunto de datos de la red previamente entrenada. Se removerá el último módulo Inception C de la arquitectura.

La transferibilidad de las características aprendidas de una red disminuye a medida que aumenta la distancia entre las tareas de origen y de destino, pero es preferible realizar la transferencia de aprendizaje entre tareas distantes que iniciar un proceso de entrenamiento desde cero con características o pesos aleatorios, dado que incluso pueden llevar a un mejor resultado en la clasificación de elementos de la tarea de destino [55]. Es de esperar que esta transferibilidad de características se manifieste en los resultados del entrenamiento, ya que se usará las generalizaciones de las primeras capas para la clasificación.

2.8.4. Arquitectura

Se debe construir inicialmente una arquitectura basada en Inception v3, este paso se realizará con TensorFlow, el cual provee una potente API para realizar esta tarea. En el repositorio de código del proyecto de Inception v3 alojado en GitHub, se puede encontrar el modelo previamente entrenado en formato ".pb" para su descarga. Esta extensión hace referencia a "Protocol Buffers", el cual es un mecanismo extensible para serializar datos estructurados.

La arquitectura propuesta se basa en la Tabla 2, adaptada a las características de Inception v3. Los cambios realizados a la arquitectura son eliminar el clasificador auxiliar ya que no se entrenará la red desde cero. Además, se elimina el último módulo Inception C y se cambia la capa totalmente conectada. Dado que no se va a entrenar toda la red neuronal convolucional desde cero y que se usará la técnica de transferencia de aprendizaje, se mantiene la arquitectura de Inception v3 hasta antes de la última capa totalmente conectada. Esto implica que no se usarán técnicas como regularización ni se cambiarán los algoritmos de las capas de pooling.

Tomando en cuenta los módulos Inception de la arquitectura Inception v3 y que solo se entrenará la última capa totalmente conectada debido al uso de transferencia de aprendizaje, las características de la arquitectura final son las siguientes (Tabla 4):

Type	Patch size/stride or remarks	Input size
conv	3x3/2	299x299x3
conv	3x3/1	149x149x32
conv padded	3x3/1	147x147x32
pool	3x3/2	147x147x64
conv	3x3/1	73x73x64
conv	3x3/2	71x71x80
conv	3x3/1	35x35x192
3xInception	Módulo Inception A	35x35x288
5xInception	Módulo Inception B	17x17x768
1xInception	Módulo Inception C	8x8x1280
pool	8x8	8x8x2048
linear	logits	1x1x2048
softmax	classifier	1x1x2

Tabla 4: Características de la arquitectura propuesta

El proceso para construir la arquitectura propuesta es el siguiente:

- Construir la arquitectura Inception v3, excepto la última capa totalmente conectada, el clasificador auxiliar y el último módulo Inception C.
- Cambiar la capa totalmente conectada para adecuarla a las 2 clases con las que se trabajará: melanoma y nevus.
- Cargar los pesos del modelo Inception v3 previamente entrenado en las secciones correspondientes.

En la Figura 17 se puede apreciar la arquitectura Inception v3 modificada para adaptarse a las clases a utilizarse (melanoma y nevus).

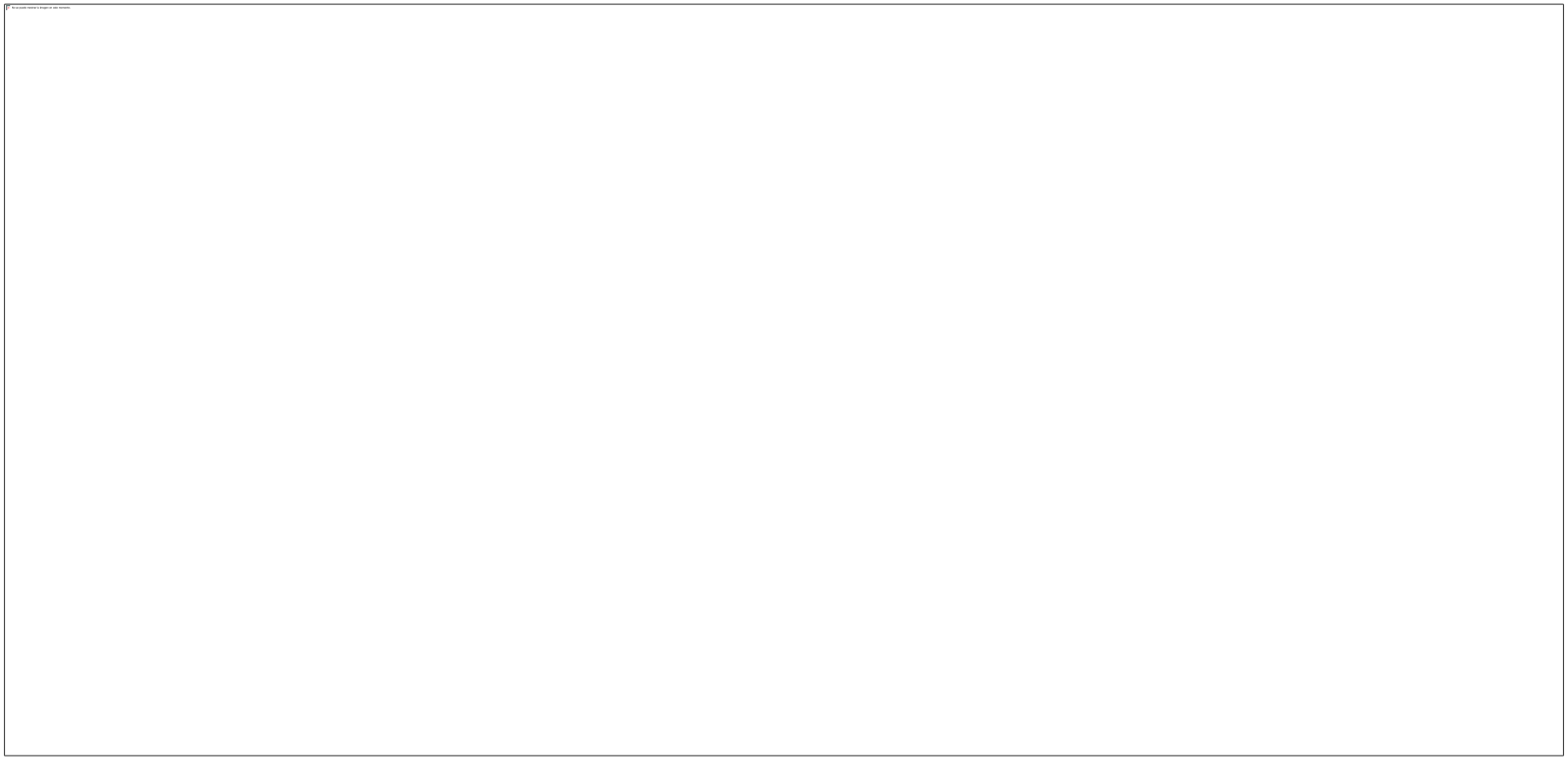

Figura 17: Arquitectura propuesta en base a la arquitectura Inception

2.8.5. Backpropagation

Este método de propagación de errores hacia atrás, conocido como backpropagation, es un método de aprendizaje supervisado para calcular de manera iterativa los errores de los pesos de las neuronas de una red neuronal artificial teniendo en cuenta la clasificación real y la clasificación predicha por el modelo. Este método consta de dos partes diferenciadas que se ejecutan de manera iterativa por cada elemento de los datos de entrenamiento [56].

La derivada de la función de error con respecto al peso θ se calcula utilizando la regla de la cadena de la siguiente manera:

$$\frac{\partial L(z,y)}{\partial \theta} = \frac{\partial L(z,y)}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial \theta}$$

siendo $L(z,y)$ la función de error, a la salida de la neurona del peso actual y z la entrada total de la neurona.

2.8.6. Descenso de gradiente

El descenso de gradiente es un algoritmo de optimización que trabaja de manera iterativa para encontrar el mínimo de una función. Dado que la función de error se usa para controlar el error en las predicciones de un modelo, el objetivo de usar el descenso de gradiente es minimizar la función de error para aumentar la exactitud del modelo mientras se ajustan los pesos del modelo. Usualmente se alcanza un mínimo local que es lo suficientemente bueno como el mínimo global de la función de error [57].

Si tenemos un conjunto de entrenamiento (x_i, y_i) para todo $i = 1, \dots, n$; siendo x_i una muestra y y_i su predicción real; y un conjunto de predicciones dadas por el modelo o hipótesis $h_\theta(x_i)$, además de una función de error $J(\theta) = \sum_{i=1}^{m} L(h_\theta(x_i), y_i)$, la regla de actualización para el descenso de gradiente se expresa con una tasa de aprendizaje $\alpha \epsilon \mathbb{R}$ y la derivada de función de error J respecto al peso θ_j:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

La Figura 18 muestra los pasos de un peso hasta su valor óptimo que minimiza la función de error:

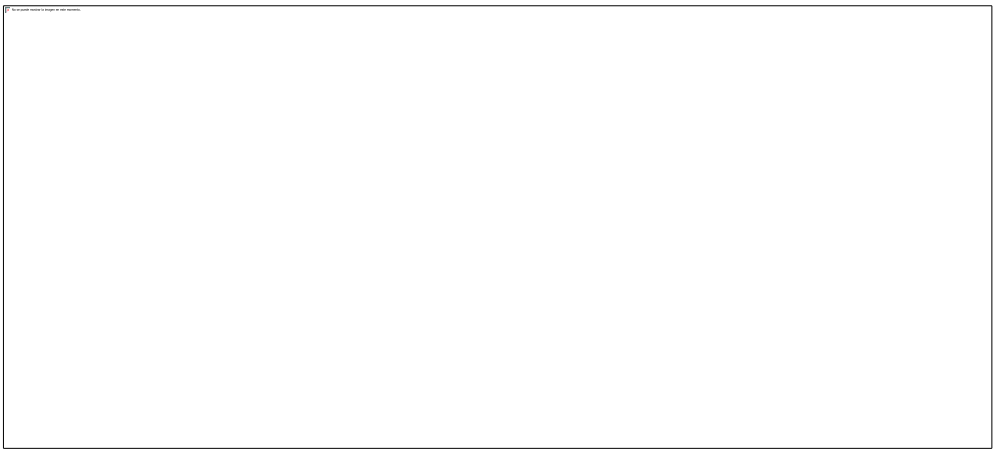

Figura 18: Descenso de gradiente. Fuente: Stanford CS 229 ML Course

Existen tres tipos de descenso de gradiente que difieren principalmente en la cantidad de muestras que utilizan [58], los cuales son el descenso de gradiente por lotes, estocástico y por mini lotes. Se usará este último tipo de descenso de gradiente para el entrenamiento, en el cual se divide el conjunto de muestras de entrenamiento en pequeños lotes y se realiza una actualización de los pesos por cada uno de esos lotes formados por muestras aleatorias.

El usar mini lotes en el algoritmo del descenso de gradiente permite encontrar un mínimo local tan bueno como el mínimo global sin usar demasiado tiempo y recursos computacionales [59].

2.8.7. Cross Entropy

La función de error cross-entropy se utiliza para cuantificar la diferencia entre dos distribuciones de probabilidad. Por lo general, la distribución de probabilidad con las predicciones reales se expresa en términos de una distribución única [60].

La ecuación de cross-entropy $L(z, y)$ para la clasificación binaria se define por:

$$L(z, y) = -[y \log(z) + (1 - y) \log(1 - z)]$$

donde y es la predicción real y z es la predicción del modelo.

2.9. Entrenamiento

Para ejecutar esta tarea primero se toma el nuevo conjunto de imágenes descrito en la sección 2.8.1 y se obtiene la siguiente cantidad de muestras:

$$\text{Conjunto de entrenamiento} = 70\% \text{ del total del conjunto de imágenes } (35\,840)$$

La tarea de entrenamiento se encarga de utilizar las muestras del conjunto de entrenamiento para determinar los pesos de la red que permitan ejecutar una tarea, en este caso de clasificación, con un alto nivel de exactitud [61].

En esta etapa se usará el algoritmo de backpropagation (2.8.5), descenso de gradiente por mini lotes (2.8.6) y cross-entropy como función de error (2.8.7). Teniendo en cuenta que el algoritmo de descenso de gradiente por mini lotes es un algoritmo iterativo, se usará el número de iteraciones como hiperparámetro del modelo, usando lotes de 32 muestras.

Existen algunos valores de hiperparámetros que se recomiendan usar en el entrenamiento [62]. Teniendo en cuenta que únicamente se entrenará la última capa totalmente conectada, se utilizarán los valores recomendados de los hiperparámetros, además de los valores originales usados en el entrenamiento de la red Inception v3 [63].

2.10. Validación

La evaluación de la exactitud de un modelo de clasificación en el conjunto de datos de entrenamiento se convierte en una calificación sesgada, ya que el modelo se ajusta precisamente a esos datos para intentar generalizar las características propias de ese conjunto y poder clasificar nuevas muestras del mismo dominio. Se usará la siguiente cantidad de muestras para esta tarea, tomando en cuenta el conjunto de imágenes de la sección 2.8.1:

$$\text{Conjunto de validación} = 15\% \text{ del total del conjunto de imágenes } (7\,680)$$

El conjunto de validación se utiliza para obtener una evaluación imparcial del ajuste del modelo en el conjunto de datos de entrenamiento mientras se ajustan los hiperparámetros de ese modelo [61] para alcanzar una mejor exactitud en la tarea de clasificación. Existen problemas que surgen en la tarea de entrenamiento, como el overfitting o sobreajuste [64] y el underfitting o subajuste [65] los cuales se pueden solventar ajustando los hiperparámetros.

Para identificar si existen estos problemas, se debe calcular el resultado de la función de error en la etapa del entrenamiento en cada iteración. Posteriormente se toma un lote del conjunto de imágenes de validación del mismo tamaño del lote que se usa en el entrenamiento. Usar un conjunto de datos de validación proporciona una evaluación imparcial del ajuste de un modelo en el conjunto de datos de entrenamiento mientras se ajustan los hiperparámetros del modelo.

Finalmente se calcula el resultado de la función de error en este lote de validación y se grafica junto al error del lote de entrenamiento respecto a cada iteración. Analizando las curvas de error de las tareas de entrenamiento y validación, se pueden realizar ajustes en los hiperparámetros de la tarea de entrenamiento de ser necesario.

2.11. Prueba

La tarea de prueba o test se refiere a utilizar datos de prueba para proporcionar una evaluación imparcial del ajuste del modelo final en el conjunto de datos de entrenamiento [66]. Esta tarea se la ejecuta cuando la tarea de entrenamiento haya finalizado y se disponga de dicho modelo final. Se utiliza este procedimiento para evaluar la generalización del clasificador y verificar su funcionamiento con nuevos datos nunca utilizados en las tareas de entrenamiento y validación. Tomando en cuenta el conjunto de muestras generadas en la sección 2.8.1, se toma la siguiente cantidad de muestras:

$$\text{Conjunto de pruebas} = 15\% \text{ del total del conjunto de imágenes } (7\,680)$$

2.12. Arquitectura del sistema

En esta sección se describe el uso del modelo de la red neuronal por parte de una aplicación web que ofrece un servicio web para la clasificación de imágenes.

Este servicio será consumido por la aplicación móvil. La arquitectura de todo el sistema que se construirá se muestra en la Figura 19:

Figura 19: Arquitectura del sistema

2.12.1. Creación de la aplicación web

El servicio web estará basado en la arquitectura REST (Representational State Transfer). La arquitectura REST [67], creada por Roy Fielding, define una serie de principios arquitectónicos en las cuales debe basarse el desarrollo de servicios web. La implementación correcta de un servicio web debe seguir estos 4 principios:

- Usar métodos HTTP de manera explícita
- No mantener estado en cada transacción realizada
- Exponer recursos como URIs
- Transferir información mediante un formato de intercambio de datos
- Usar una nomenclatura simple basada en jerarquías para los recursos

Estos principios nos permiten obtener flexibilidad al momento de exponer los recursos que deseamos ofrecer. Esta aplicación web se creará usando Flask, la cual constará de 3 partes:

- Aplicación: punto de inicio en donde se define una ruta con el nombre “evaluate”, la cual, usando el método POST, obtendrá la imagen enviada y devolverá el resultado

de la clasificación en formato JSON. La aplicación se inicia en el puerto 5000 por defecto.

- Controlador: clase en donde se definen operaciones para guardar las imágenes receptadas, así como validaciones de formatos y manejo de códigos HTTP de respuesta para diferentes escenarios.
- Servicio de clasificación: clase en donde se carga el modelo entrenado de la red neuronal, se inicia una sesión de TensorFlow y se clasifica una imagen.

El diagrama de clases es el que muestra la Figura 20 :

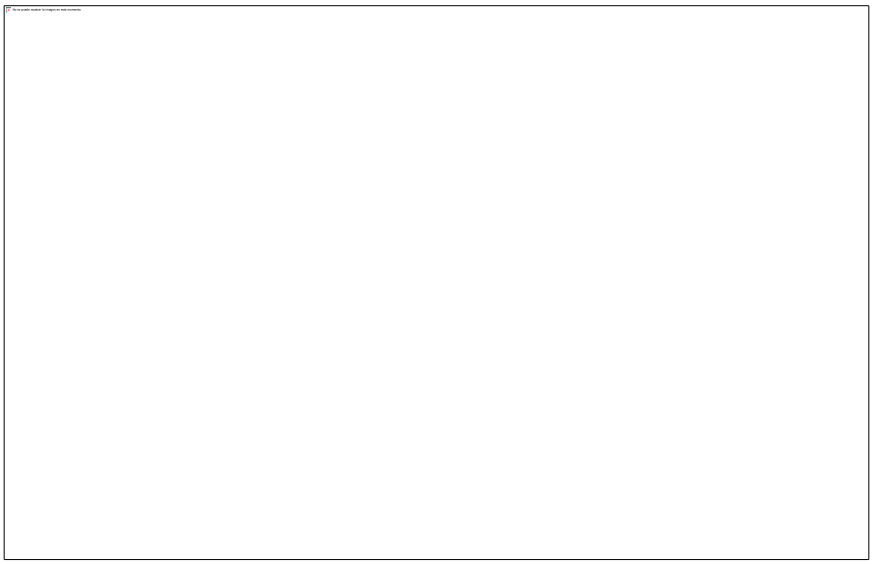

Figura 20: Diagrama de clases de la aplicación web

Se podrá acceder al servicio REST realizando una petición POST al recurso */evaluate*, ya que se enviará información al servidor y queremos obtener el resultado de ese envío.

2.12.2. Construcción de la aplicación móvil

La aplicación móvil se construirá con tres vistas: una para la configuración, otra para capturar la imagen a ser enviada al servicio REST y finalmente una vista para mostrar el resultado de la clasificación (Figura 21). Se usará herramientas como Ionic y TypeScript y se usará la versión 4.4 de Android como base mínima para la compatibilidad de la aplicación.

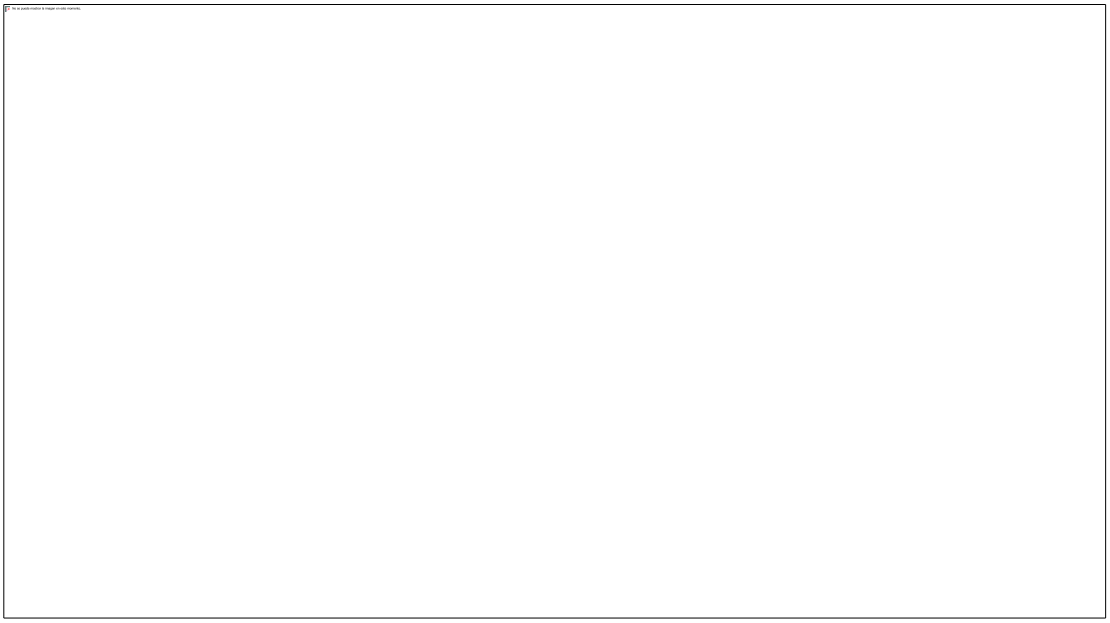

Figura 21: Vistas de la aplicación móvil

3. RESULTADOS Y DISCUSIÓN

En esta sección se muestran los resultados obtenidos luego de ejecutar las tareas de entrenamiento, validación y prueba. Se indican los gráficos de las funciones de error de las tareas de entrenamiento y validación y la curva de aprendizaje, además de las matrices de confusión [68] de estas dos tareas y de la tarea de prueba. Adicionalmente, se muestran los valores de la exactitud, sensibilidad y especificidad alcanzadas en estas tareas, en base a las siguientes definiciones y ecuaciones [69]:

Exactitud: es una relación entre las muestras clasificadas correctamente y el total de las muestras. Se define como:

$$Exactitud = \frac{VP + VN}{VP + FP + FN + VN}$$

donde VP son los verdaderos positivos, VN son los verdaderos negativos, FP son los falsos positivos y FN los falsos negativos.

Sensibilidad: mide la proporción de verdaderos positivos que se clasifican correctamente como tales. Se define como:

$$Sensibilidad = \frac{VP}{VP + FN}$$

donde VP son los verdaderos positivos y FN los falsos negativos. Este valor (mientras más alto es mejor) indicará que tan bueno es el modelo clasificando imágenes con la clase nevus.

Especificidad: mide la proporción de verdaderos negativos que se clasifican correctamente como tales. Se define como:

$$Especificidad = \frac{VN}{VN + FP}$$

donde VN son los verdaderos negativos y FP los falsos positivos. Este valor (mientras más alto es mejor) indicará que tan bueno es el modelo clasificando imágenes con la clase melanoma.

Finalmente, se calcula el coeficiente de correlación de Matthews (MCC), el cual se utiliza como una medida de la calidad de un clasificador binario. Es un coeficiente de correlación entre los valores reales de las clases de las muestras y sus predicciones [70]. El resultado de su cálculo devuelve un valor entre -1 y 1. El valor de -1 representa un error total entre la predicción de una muestra y su clase, un valor de cero representa una predicción aleatoria y un valor de +1 representa una predicción perfecta. Se calcula de la siguiente manera:

$$MCC = \frac{VP \times VN - FP \times FN}{\sqrt{(VP + FP)(VP + FN)(VN + FP)(VN + FN)}}$$

Donde VP son los verdaderos positivos, VN son los verdaderos negativos, FP son los falsos positivos y FN son los falsos negativos. Este valor es más informativo que otras medidas que se pueden calcular de la matriz de confusión como la precisión y el F1-score, ya que toma en cuenta las proporciones de los cuatro valores de la matriz de confusión [71].

3.1. Resultados

Se muestran los resultados de los modelos generados con dos escenarios que incluyen variaciones en los hiperparámetros de tasa de aprendizaje y tamaño del lote.

3.1.1. Primer escenario

Se inicia ajustando el valor de la tasa de aprendizaje, por lo que se evaluaron tres casos con tres valores diferentes de este hiperparámetro. Se usaron los siguientes hiperparámetros para el entrenamiento (Tabla 5):

Hiperparámetro	**Valor del hiperparámetro**
Imágenes de entrenamiento	35 840 imágenes (17 920 por clase)
Tasa de aprendizaje	0,01; 0,001 y 0,0001
Iteraciones	4480
Tamaño de lote	32 imágenes

Tabla 5: Hiperparámetros para entrenamiento del primer escenario

Para la tarea de validación se usaron los siguientes valores (Tabla 6):

Recurso	Valor
Imágenes de validación	7 680 imágenes (3 840 por clase)
Tamaño de lote	32 imágenes

Tabla 6: Recursos y valores utilizados para validación del primer escenario

Para la tarea de prueba se usaron los siguientes valores (Tabla 7):

Recurso	Valor
Imágenes de prueba	7 680 imágenes (3 840 por clase)

Tabla 7: Recursos y valores utilizados para prueba del primer escenario

Los resultados de la ejecución de las tres tareas son los siguientes:

Tasa de aprendizaje: 0,01

Figura 22: Matrices de confusión con tasa de aprendizaje igual a 0,01

Tarea	Exactitud	MCC	Sensibilidad (Nevus)	Especificidad (Melanoma)
Entrenamiento	0,8926	0,7896	0,8398	0,9455
Validación	0,8924	0,7882	0,8455	0,9390

Prueba	0,8961	0,7977	0,8353	0,9560

Tabla 8: Resultados de tareas con tasa de aprendizaje igual a 0,01

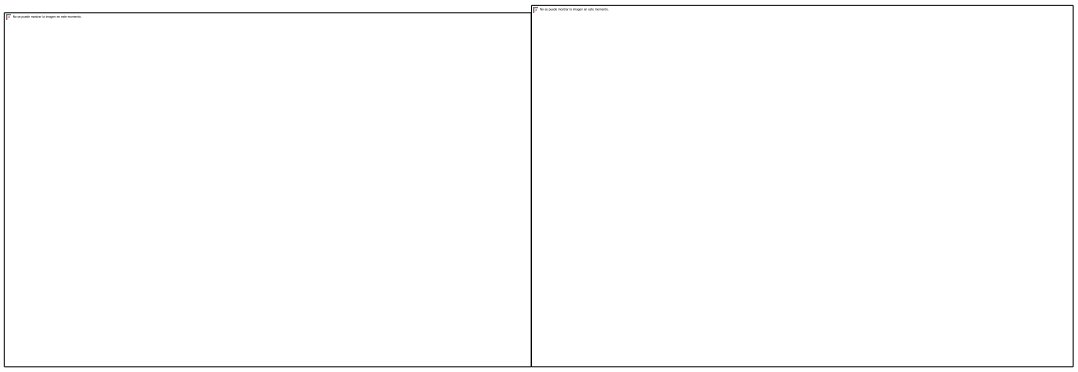

Figura 23: a) Curva de aprendizaje y b) curva de error con tasa de aprendizaje igual a 0,01

Tasa de aprendizaje: 0,001

Figura 24: Matrices de confusión con tasa de aprendizaje igual a 0,001

Tarea	Exactitud	MCC	Sensibilidad (Nevus)	Especificidad (Melanoma)
Entrenamiento	0,8883	0,7856	0,8136	0,9633
Validación	0,8921	0,7922	0,8211	0,9631
Prueba	0,8915	0,7896	0,8250	0,9570

Tabla 9: Resultados de tareas con tasa de aprendizaje igual a 0,001

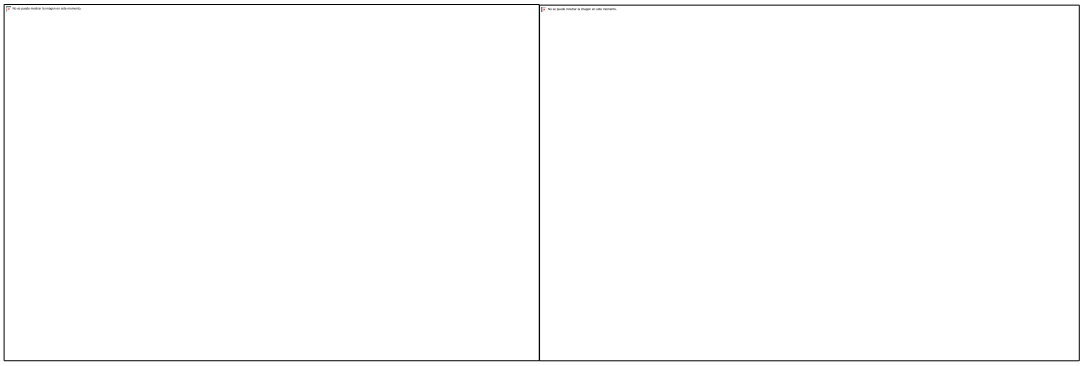

Figura 25: a) Curva de aprendizaje y b) curva de error con tasa de aprendizaje igual a 0,001

Tasa de aprendizaje: 0,0001

Figura 26: Matrices de confusión con tasa de aprendizaje igual a 0,0001

Tarea	Exactitud	MCC	Sensibilidad (Nevus)	Especificidad (Melanoma)
Entrenamiento	0,8784	0,7724	0,7785	0,9784
Validación	0,8833	0,7802	0,7902	0,9760
Prueba	0,8841	0,7800	0,7942	0,9718

Tabla 10: Resultados de tareas con tasa de aprendizaje igual a 0,0001

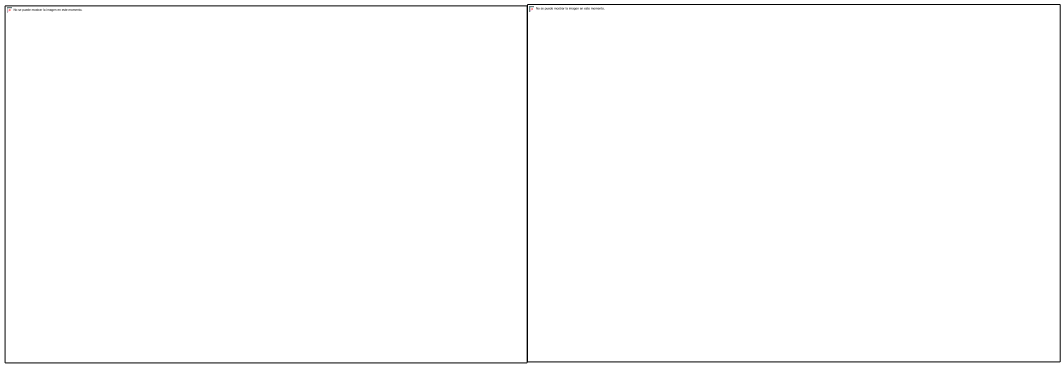

Figura 27: a) Curva de aprendizaje y b) curva de error con tasa de aprendizaje igual a 0,0001

En las matrices de confusión (Figura 22, Figura 24 y Figura 26, sección a) de entrenamiento tenemos un total de muestras de 143 360 en cada una. Esto se debe a que al utilizar lotes de 32 muestras para el entrenamiento y un número de iteraciones igual a 4 480, al final del entrenamiento se utilizan $4\,480 \times 32 = 143\,360$ muestras para la tarea de entrenamiento.

En las matrices de confusión de validación (Figura 22, Figura 24 y Figura 26, sección b) tenemos un total de muestras de 143 360 en cada una. De manera similar a la tarea de entrenamiento, esto se debe a que, al utilizar lotes de 32 muestras para la validación por cada iteración, y un número de iteraciones de entrenamiento igual a 4 480, al final del entrenamiento se utilizan $4480 \times 32 = 143\,360$ muestras para la tarea de validación.

En las matrices de confusión de pruebas (Figura 22, Figura 24 y Figura 26, sección c) tenemos un total de muestras de 7 678 en cada una, correspondiente al 15% del total de muestras del conjunto de imágenes.

En las curvas de aprendizaje (Figura 23, Figura 25 y Figura 27) se puede apreciar visualmente que la diferencia entre los valores de exactitud de las curvas de entrenamiento y validación se incrementa conforme disminuye la tasa de aprendizaje, incidiendo también en el aumento del error en las curvas de error en las figuras mencionadas.

El resumen de las 3 tareas (Tabla 8,

Tabla 9 y Tabla 10), variando la tasa de aprendizaje, se muestra a continuación:

Tasa de aprendizaje	Exactitud prueba	MCC prueba	Exactitud validación	MCC validación
0,01	**0,8961**	**0,7977**	**0,8924**	**0,7882**

0,001	0,8915	0,7896	0,8921	0,7922
0,0001	0,8841	0,7800	0,8833	0,7802

Tabla 11: Resumen de resultados al variar la tasa de aprendizaje

Para seleccionar el mejor modelo de los tres, se toma en cuenta el mayor valor de MCC en prueba y en validación, así como el mayor valor de exactitud en prueba y en validación. Analizando los resultados de la Tabla 11, el modelo seleccionado es aquel que usa una tasa de aprendizaje de 0,01; puesto que es el que cumple el mayor número de criterios de selección.

3.1.2. Segundo escenario

Tomando en cuenta el primer escenario, se optó por usar una tasa de aprendizaje igual a 0,01 y variar el tamaño del lote de entrenamiento, por lo que se evaluaron tres casos con tres valores diferentes de este hiperparámetro. Se usaron los siguientes hiperparámetros para el entrenamiento (Tabla 12):

Hiperparámetro	**Valor del hiperparámetro**
Imágenes de entrenamiento	35 840 imágenes (17 920 por clase)
Tasa de aprendizaje	0,01
Iteraciones	4 480
Tamaño de lote	64, 128 y 256 imágenes

Tabla 12: Hiperparámetros para entrenamiento del segundo escenario

Para la tarea de validación se usaron los siguientes valores (Tabla 13):

Recurso	**Valor**
Imágenes de validación	7 680 imágenes (3 840 por clase)
Tamaño de lote	64, 128 y 256 imágenes

Tabla 13: Recursos y valores utilizados para validación del segundo escenario

Para la tarea de prueba se usaron los siguientes valores (Tabla 14):

Recurso	Valor
Imágenes de prueba	7 680 imágenes (3 840 por clase)

Tabla 14: Recursos y valores utilizados para prueba del segundo escenario

Los resultados de la ejecución de las tres tareas son los siguientes:

Lote: 64 imágenes

Figura 28: Matrices de confusión con tamaño de lote igual a 64

En la matriz de confusión de entrenamiento al usar un lote de 64 muestras (Figura 28, sección a), tenemos un total de muestras de 286 720. Esto se debe a que al utilizar lotes de 64 muestras para el entrenamiento y un número de iteraciones igual a 4 480, al final del entrenamiento se utilizan $4\,480 \times 64 = 286\,720$ muestras para esa tarea de entrenamiento. Debe tomarse en cuenta que cada lote de entrenamiento de 64 muestras proviene del número original de muestras de entrenamiento (35 840 imágenes, 70% del total de muestras).

En la matriz de confusión de validación al usar un lote de 64 muestras (Figura 28, sección b), tenemos un total de muestras de 286 720. De manera similar a la tarea de entrenamiento, esto se debe a que, al utilizar lotes de 64 muestras para la validación por cada iteración, y un número de iteraciones de entrenamiento igual a 4 480, al final del entrenamiento se utilizan $4\,480 \times 64 = 286\,720$ muestras para esa tarea de validación.

Debe tomarse en cuenta que cada lote de validación de 64 muestras proviene del número original de muestras de validación (7 680 imágenes, 15% del total de muestras).

En la matriz de confusión de pruebas (Figura 28, sección c), se usan 7 680 imágenes para la tarea de prueba (7 680 imágenes, 15% del total de muestras).

Tarea	Exactitud	MCC	Sensibilidad (Nevus)	Especificidad (Melanoma)
Entrenamiento	0,8945	0,7942	0,8373	0,9516
Validación	0,8955	0,7950	0,8464	0,9449
Prueba	0,8926	0,7870	0,8558	0,9288

Tabla 15: Resultados de tareas con tamaño de lote igual a 64

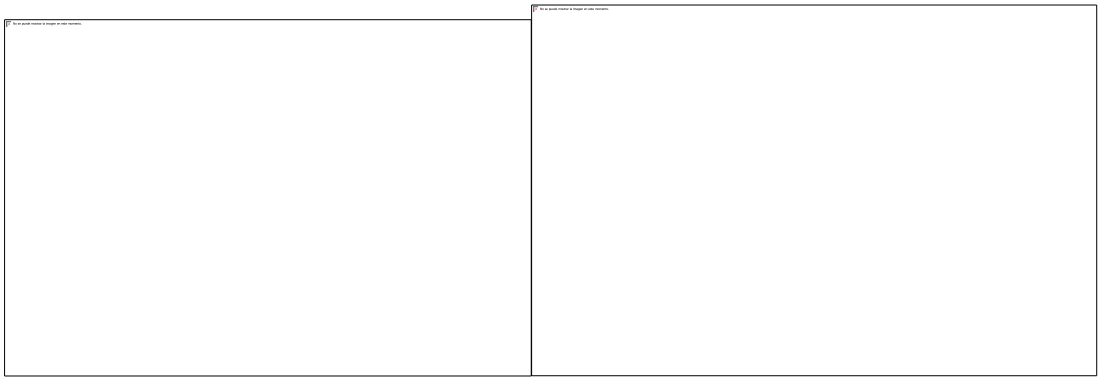

Figura 29: a) Curva de aprendizaje y b) curva de error con tamaño de lote igual a 64

Lote: 128 imágenes

Figura 30: Matrices de confusión con tamaño de lote igual a 128

En la matriz de confusión de entrenamiento al usar un lote de 128 muestras (Figura 30, sección a), tenemos un total de muestras de 573 440. Esto se debe a que al utilizar lotes de 128 muestras para el entrenamiento y un número de iteraciones igual a 4480, al final del entrenamiento se utilizan $4\,480 \times 128 = 573\,440$ muestras para esa tarea de entrenamiento. Debe tomarse en cuenta que cada lote de entrenamiento de 128 muestras proviene del número original de muestras de entrenamiento (35 840 imágenes, 70% del total de muestras).

En la matriz de confusión de validación al usar un lote de 128 muestras (Figura 30, sección b), tenemos un total de muestras de 573 440. De manera similar a la tarea de entrenamiento, esto se debe a que, al utilizar lotes de 128 muestras para la validación por cada iteración, y un número de iteraciones de entrenamiento igual a 4 480, al final del entrenamiento se utilizan $4\,480 \times 128 = 573\,440$ muestras para esa tarea de validación. Debe tomarse en cuenta que cada lote de validación de 64 muestras proviene del número original de muestras de validación (7 680 imágenes, 15% del total de muestras).

En la matriz de confusión de pruebas (Figura 30, sección c), se usan 7 680 imágenes para la tarea de prueba (7 680 imágenes, 15% del total de muestras).

Tarea	Exactitud	MCC	Sensibilidad (Nevus)	Especificidad (Melanoma)
Entrenamiento	0,8956	0,7967	0,8371	0,9541
Validación	0,8967	0,7974	0,8466	0,9468
Prueba	0,8910	0,7882	0,8282	0,9337

Tabla 16: Resultados de tareas con tamaño de lote igual a 128

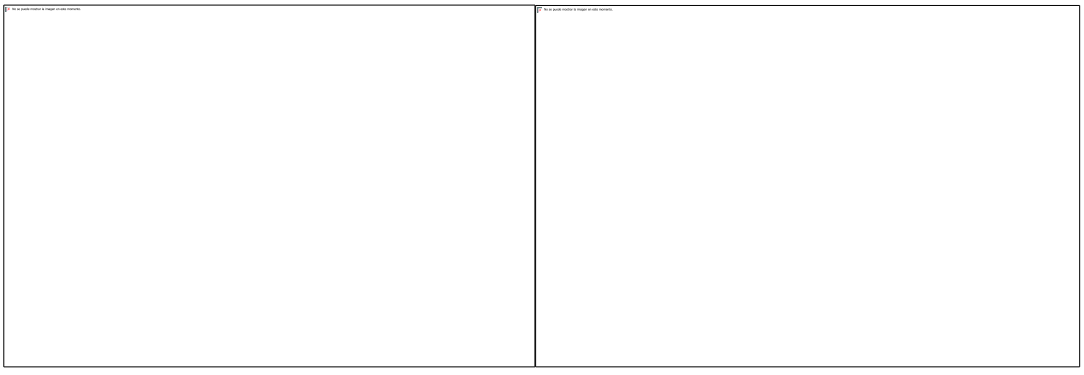

Figura 31: a) Curva de aprendizaje y b) curva de error con tamaño de lote igual a 128

Lote: 256 imágenes

Figura 32: Matrices de confusión con tamaño de lote igual a 256

En la matriz de confusión de entrenamiento al usar un lote de 256 muestras (Figura 32, sección a), tenemos un total de muestras de 1 146 880. Esto se debe a que al utilizar lotes de 128 muestras para el entrenamiento y un número de iteraciones igual a 4 480, al final del entrenamiento se utilizan $4\,480 \times 256 = 1\,146\,880$ muestras para esa tarea de entrenamiento. Debe tomarse en cuenta que cada lote de entrenamiento de 256 muestras proviene del número original de muestras de entrenamiento (35 840 imágenes, 70% del total de muestras).

En la matriz de confusión de validación al usar un lote de 256 muestras (Figura 32, sección b), tenemos un total de muestras de 1 146 880. De manera similar a la tarea de entrenamiento, esto se debe a que, al utilizar lotes de 256 muestras para la validación por cada iteración, y un número de iteraciones de entrenamiento igual a 4 480, al final del entrenamiento se utilizan $4480 \times 256 = 1\,146\,880$ muestras para esa tarea de validación. Debe tomarse en cuenta que cada lote de validación de 256 muestras proviene del número original de muestras de validación (7 680 imágenes, 15% del total de muestras).

En la matriz de confusión de pruebas (Figura 32, sección c), se usan 7 680 imágenes para la tarea de prueba (7 680 imágenes, 15% del total de muestras).

Tarea	Exactitud	MCC	Sensibilidad (Nevus)	Especificidad (Melanoma)
Entrenamiento	0,8965	0,7984	0,8379	0,9551

Validación	0,8967	0,7975	0,8453	0,9479
Prueba	0,8963	0,7974	0,8398	0,9521

Tabla 17: Resultados de tareas con tamaño de lote igual a 256

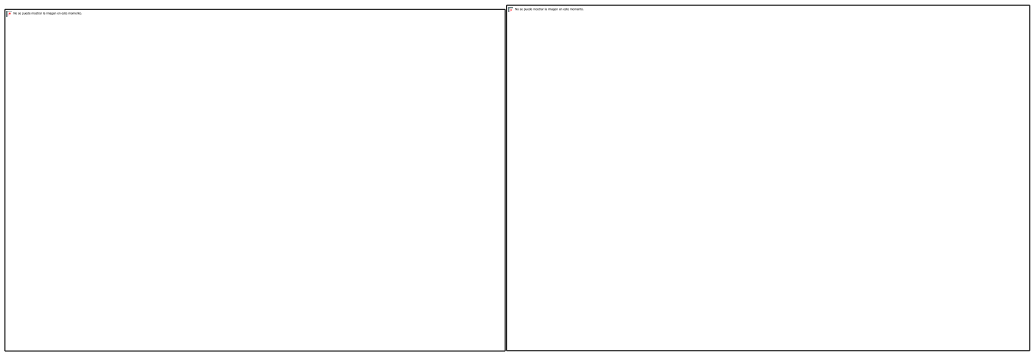

Figura 33: a) Curva de aprendizaje y b) curva de error con tamaño de lote igual a 256

En las curvas de aprendizaje (Figura 29, Figura 31 y Figura 33) se puede apreciar visualmente que la diferencia entre los valores de exactitud de las curvas de entrenamiento y validación disminuye conforme aumenta el tamaño del lote, incidiendo también en la disminución del error en las curvas de error en las figuras mencionadas.

El resumen de las 3 tareas (Tabla 15, Tabla 16 y Tabla 17), variando el tamaño del lote, se muestra a continuación:

Tamaño de lote	**Exactitud prueba**	**MCC prueba**	**Exactitud validación**	**MCC validación**
64	0,8926	0,7870	0,8955	0,7950
128	0,8910	0,7882	0,8967	0,7974
256	**0,8963**	**0,7974**	**0,8967**	**0,7975**

Tabla 18: Resumen de resultados al variar el tamaño del lote

Para seleccionar el mejor modelo de los tres, se toma en cuenta el mayor valor de MCC en prueba y en validación, así como el mayor valor de exactitud en prueba y en validación.

Analizando los resultados de la Tabla 18, el modelo seleccionado es aquel que usa un tamaño de lote igual a 256; puesto que es el que cumple el mayor número de criterios de selección.

3.2. Discusión

En todas las matrices de confusión, al analizar la sensititividad (nevus) y la especificidad (melanoma) se clasifica de una manera más exacta las imágenes de melanoma que nevus.

En las curvas de error de todos los casos se puede apreciar que, en las primeras iteraciones, la tasa de error de validación es menor a la de entrenamiento. Esto se debe a que el error de entrenamiento es el promedio de los errores sobre cada lote de muestras de entrenamiento. Dado que el modelo cambia en cada iteración, el error del entrenamiento de los primeros lotes de las primeras iteraciones es generalmente más alto que el error de los últimos lotes.

Tomando en cuenta las definiciones de melanoma, nevus y lesión melanocítica de la sección 2.1 y trabajos existentes de la relación entre melanoma y nevus [72] [73], cuando existen anomalías en el crecimiento de los melanocitos, en etapas tempranas es muy difícil determinar si ese crecimiento anormal de los melanocitos originará un melanoma. Sin embargo, si ese crecimiento da origen a las estructuras dermatológicas consistentes con melanoma, es más fácil realizar el diagnóstico para un dermatólogo con experiencia. El modelo del presente trabajo concuerda con lo mencionado anteriormente, ya que es mejor clasificando imágenes de melanoma que de nevus (Tabla 17).

Escenario	**Exactitud validación**	**MCC validación**	**Exactitud prueba**	**MCC prueba**
Primer escenario (LR=0,01 y lote=32)	0,8924	0,7882	0,8961	0,7977
Segundo escenario (LR=0,01 y lote=256)	0,8967	0,7975	**0,8963**	0,7974
T Maier et al. [5]	---	---	0,8100	---
A. Gadiraju, S. Keshwani, and E. Minkin [6]	---	---	0,7980	---

Tabla 19: Comparativa de los modelos generados y modelos de otros autores

El modelo final elegido en el presente trabajo es aquel que fue entrenado con una tasa de aprendizaje de 0,01 y un tamaño de lote de 256 muestras (Tabla 19). Si bien un tamaño de lote lo suficientemente grande permite evitar el ruido y acercar al modelo a un mínimo global, la ganancia no es tan alta como se esperaba, ya que, al usar una tasa de aprendizaje de 0,01 y un lote de 32 muestras, la exactitud de prueba es igual a 0,8961; mientras que al usar la misma tasa de aprendizaje y un lote de 256 muestras, la exactitud de prueba es igual a 0,8963; teniendo una ganancia de apenas 2×10^{-4} en la exactitud del clasificador. De la misma manera con la exactitud de validación de ambos escenarios, se tiene una ganancia de apenas $4{,}3 \times 10^{-3}$; representando una baja ganancia al usar un número de recursos computacionales altos al aumentar el tamaño del lote. El coeficiente de Matthews en prueba disminuye en 3×10^{-4}.

Finalmente, un dermatólogo con experiencia puede clasificar imágenes dermatoscópicas de melanoma con una exactitud de 89% a 90%, y de 74,5% a 77,7% la tasa de clasificaciones correctas con la que puede clasificar imágenes dermatoscópicas de nevus [74] [75]. El modelo seleccionado (Tabla 17) de todos los modelos generados en este trabajo puede clasificar imágenes dermatoscópicas de melanoma con una tasa de clasificación de 95,21%, y de 83,98% la tasa de clasificación de imágenes dermatoscópicas de nevus, teniendo una exactitud del 89,63%. Esto implica que el modelo iguala a dermatólogos con experiencia y supera a los modelos mencionados en la sección 1.1.

4. CONCLUSIONES Y RECOMENDACIONES

En esta sección se establecen las conclusiones y recomendaciones del proyecto.

4.1. Conclusiones

Se verificó que el modelo generado se ajusta a los niveles de clasificación de dermatólogos con experiencia, ya que ellos pueden clasificar imágenes dermatoscópicas de melanomas con un mayor nivel de exactitud que las imágenes dermatoscópicas de nevus. Además, el modelo generado supera en exactitud a los 2 modelos propuestos en el presente trabajo, logrando así el objetivo general propuesto.

Realizar un aumento de datos al conjunto de imágenes obtenido de ISIC Archive, usando varias técnicas de preprocesamiento de imágenes, permitió proporcionar información al modelo para que pueda clasificar imágenes de melanoma y nevus con diferentes variaciones respecto al conjunto de datos original. Sirvió adicionalmente para evitar el largo y costoso proceso que puede resultar de recolectar imágenes de lesiones junto con su diagnóstico histopatológico, además de la limpieza de datos que supone este proceso para garantizar el anonimato de los pacientes o cualquier dato que pueda identificarlos.

El hecho de ejecutar las tareas de entrenamiento, validación y prueba usando recursos computacionales en la nube resultó de gran utilidad, puesto que se lograron cumplir estas tareas incluida la tarea de aumento de datos sin necesidad de usar recursos computacionales de manera local, ahorrando el costo monetario en términos de adquisición de hardware y software, así como el costo energético asociado al uso de esos elementos.

Ejecutar una aplicación web basada en la arquitectura REST facilitó el uso del modelo generado en una aplicación móvil. Adicionalmente el uso del lenguaje Python permitió una integración fácil entre el código de TensorFlow (para usar el modelo generado) y el código de la aplicación web creada con Flask (para recibir imágenes y enviarlas al modelo).

El hecho de usar un tamaño de lote de entrenamiento y de validación alto permitió encontrar un mejor modelo comparado con el modelo generado por el uso de un tamaño de lote bajo; sin embargo, la ganancia en la exactitud fue muy pequeña como para justificar el tiempo y esfuerzo invertidos. Con eso se concluye que usar el algoritmo de descenso de gradiente por mini lotes con un tamaño de lote bajo es tan bueno como usarlo con un tamaño de lote alto, con la ventaja que, al usar un tamaño de lote bajo, el modelo convergerá más rápido a un mínimo local.

4.2. Recomendaciones

La técnica de aumento de datos ha resultado ser de gran ayuda para generar una gran cantidad de imágenes para compensar el bajo número de imágenes de melanoma respecto al número de imágenes de nevus. Se recomienda acceder a un hardware más potente que incluya un clúster de GPUs para realizar un entrenamiento de toda la red iniciando los pesos de la red neuronal convolucional Inception v3 con los valores del modelo pre entrenado con el conjunto de imágenes ImageNet.

Al eliminar un módulo Inception tipo C, se disminuyó el número de cálculos al evitar usar los pesos de esa capa que representaba la generalización de todas las clases del conjunto de imágenes ImageNet. Se recomienda hacer pruebas adicionales removiendo el resto de los módulos uno por uno, volver a entrenar la última capa totalmente conectada y verificar estos indicadores cada vez que se elimine un módulo. Este proceso puede permitir optimizar el modelo clasificador para evitar realizar operaciones de convolución innecesarias al clasificar, debido a que las capas iniciales contienen información de filtros generales que sirven para cualquier tipo de problemas de clasificación de imágenes.

La aplicación web fue diseñada para ser ejecutada bajo un único servidor, por lo que su disponibilidad de recursos de hardware es limitada. Existe una herramienta de TensorFlow llamada TensorFlow Server, la cual permite alojar el modelo generado y proporciona un acceso en forma de recurso REST. También ejecuta de manera abstracta la clasificación usando recursos como CPUs y GPUs para las tareas que lo requieran, de manera transparente para el programador. Se recomienda usar este servicio para proporcionar una mayor disponibilidad del servicio web creado.

Se recomienda utilizar la técnica de early stopping o la regularización del valor de los pesos sinápticos para obtener un modelo lo suficientemente bueno con un mínimo número de iteraciones. Se debe usar esta técnica con un tamaño de lote alto, para evitar los problemas de ruido inherentes al uso del descenso de gradiente por mini lotes. También se recomienda utilizar conjuntamente la validación cruzada, ya que es un método rápido de calcular y permite evaluar el rendimiento predictivo del modelo de clasificación.

Usar Ionic 2 con TypeScript permitió prototipar la aplicación móvil en muy poco tiempo. Además, una de sus características es usar el mismo código fuente para generar una aplicación compatible con iOS, por lo que se recomienda seguir usando esta herramienta.

Se recomienda revisar el estado del arte de arquitecturas de redes neuronales convolucionales y realizar comparaciones entre diferentes arquitecturas usando la técnica

de transferencia de aprendizaje, para verificar la mejor arquitectura que se pueda utilizar para resolver el problema de clasificación de imágenes dermatoscópicas de melanoma y nevus con una alta exactitud. Además, se recomienda realizar comparaciones entre diferentes algoritmos de clasificación enfocados en crear un clasificador binario.

Finalmente se recomienda utilizar datos adicionales a las imágenes como sexo, edad, historia familiar y personal de lesiones dermatológicas, entre otros, para construir un clasificador que considere estas variables médicas, puesto que en dermatología son también una fuente de datos a consultar (en caso de existir) antes de emitir un diagnóstico definitivo.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] SOLCA Núcleo de Quito, "Incidencia del Cáncer en Quito Período 2011-2013," Quito, 2017.

[2] American Cancer Society. (2016, Mayo) Survival Rates for Melanoma Skin Cancer, by Stage. [Online]. https://www.cancer.org/cancer/melanoma-skin-cancer/detection-diagnosis-staging/survival-rates-for-melanoma-skin-cancer-by-stage.html

[3] M. Kamel Boulos, A. Brewer, C. Karimkhani, D. Buller, and R. Dellavalle, "Mobile medical and health apps: state of the art, concerns, regulatory control and certification," *Online Journal of Public Health Informatics*, vol. 5, no. 3, pp. 2-3, 2014.

[4] SkinVision. About SkinVision. [Online]. https://skinvision.com/about-skin-cancer-melanoma-mobile-app

[5] T Maier et al., "Accuracy of a smartphone application using fractal image analysis of pigmented moles compared to clinical diagnosis and histological result," *Journal of the European Academy of Dermatology and Venereology*, vol. 29, pp. 663-667, Abril 2015.

[6] A. Gadiraju, S. Keshwani, and E. Minkin, "Mole Investigator: Detecting Cancerous Skin Moles Through Computer Vision," 2015. [Online]. https://www.seas.upenn.edu/~cse400/CSE400_2014_2015/reports/20_report.pdf

[7] A.P. Kassianos, J.D. Emery, P. Murchie, and F.M. Walter, "Smartphone applications for melanoma detection by community, patient and generalist clinician users: a review," *British Journal of Dermatology*, vol. 172, pp. 1507–1518, Junio 2015.

[8] M. Amayri and S. Ploix, *Machine Learning with Python/Scikit-Learn*. Grenoble, Francia, 2015.

[9] N. Shukla, "TensorFlow," in *Machine Learning with TensorFlow*.: Manning, 2018, pp. 21-22.

[10] F. Chollet, "Introduction to Keras," in *Deep Learning with Python*.: Manning Publications Company, 2017, pp. 61-62.

[11] F. Chollet. (2018) Keras: The Python Deep Learning library. [Online]. https://keras.io/#guiding-principles

[12] A. Ronacher. (2017) Flask. [Online]. http://flask.pocoo.org/

[13] J. Turnbull, "Introducing Docker," in *The Docker Book: Containerization is the new virtualization*, James Turnbull, Ed., 2014, pp. 7-16.

[14] I. Singh, *Ionic 2 Blueprints*. Mumbay: Packt Publishing, 2016, pp. 8-9.

[15] Ionic. (2018) Ionic View App. [Online]. https://ionicframework.com/pro/view

[16] Amazon Web Services. (2018) Cloud Computing con Amazon Web Services. [Online]. https://aws.amazon.com/es/what-is-aws/

[17] Amazon Web Services. (2018) Amazon EC2. [Online]. https://aws.amazon.com/es/ec2/

[18] Amazon Web Services. (2018) Instancias P2 de Amazon EC2. [Online]. https://aws.amazon.com/es/ec2/instance-types/p2/

[19] Amazon Web Services. (2018) Instancias T2 de Amazon EC2. [Online]. https://aws.amazon.com/es/ec2/instance-types/t2/

[20] American Cancer Society. (2018) ¿Qué es el cáncer de piel tipo melanoma? [Online]. https://www.cancer.org/es/cancer/cancer-de-piel-tipo-melanoma/acerca/que-es-melanoma.html

[21] M. Rastrelli et al., "Melanoma m1: Diagnosis and Therapy," *International Journal of Experimental and Clinical Pathophysiology and Drug Research*, vol. 28, pp. 273-285, May 2014.

[22] Iyatomi H. et al., "Classification of melanocytic skin lesions from non-melanocytic lesions," *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 5407-5410, 2010.

[23] Harald K. et al., "Standardization of terminology in dermoscopy/dermatoscopy: Results of the third consensus conference of the International Society of Dermoscopy," *Journal of the American Academy of Dermatology*, vol. 74, no. 6, pp. 1093-1106, 2016.

[24] S.J. Brown and C.M. Lawrence, "The management of skin malignancy: to what extent should we rely on clinical diagnosis?," *British Journal of Dermatology*, vol. 155, no. 1, pp. 100-103, May 2006.

[25] E. Errichetti and G. Stinco, "Dermoscopy in General Dermatology: A Practical Overview," *Dermatology and Therapy*, vol. 6, no. 4, pp. 471–507, December 2016.

[26] ISIC. (2017) ISIC Project. [Online]. https://isdis.net/isic-project

[27] C. Dolianitis, J. Kelly, R. Wolfe, and P. Simpson, "Comparative Performance of 4 Dermoscopic Algorithms by Nonexperts for the Diagnosis of Melanocytic Lesions," *Archives of Dermatology*, vol. 141, no. 8, pp. 1008–1014, August 2005.

[28] Stolz W. et al., "The ABCD rule of dermatoscopy. High prospective value in the diagnosis of doubtful melanocytic skin lesions," *European Journal of Dermatology*, vol. 4, pp. 521-527, April 1994.

[29] G. Argenziano et al., "Epiluminescence microscopy for the diagnosis of doubtful melanocytic skin lesions. Comparison of the ABCD rule of dermatoscopy and a new 7-point checklist based on pattern analysis," *Archives of Dermatology*, vol. 134, no. 12, pp. 1563-1570, 1998.

[30] Menzies S. W., Ingvar C., Crotty K. A., and McCarthy W. H., "Frequency and morphologic characteristics of invasive melanomas lacking specific surface microscopic features," *Archives of Dermatology*, vol. 132, no. 10, pp. 1178-1182, 1996.

[31] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard, "Adaptive data augmentation for image classification," *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3688-3692, September 2016.

[32] L. Perez and J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning," *Computing Research Repository*, vol. abs/1712.04621, pp. 2-7, 2017.

[33] J. Flusser, T. Suk, and B. Zitová, "2D Moment Invariants to Translation, Rotation, and Scaling," in *2D and 3D Image Analysis by Moments*. Prague, Czech Republic: John Wiley & Sons, 2016, pp. 63-123.

[34] G. Dreyfus, "Neural Networks: Definitions and Properties," in *Neural Networks, Methodology and Applications*.: Springer-Verlag Berlin Heidelberg, 2005, pp. 2-7.

[35] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314, 1989.

[36] H. Gao and S. Ji, "Efficient and Invariant Convolutional Neural Networks for Dense Prediction," Washington State University, 2017.

[37] N. Buduma, "Convolutional Neural Networks," in *Fundamentals of Deep Learning, Designing Next-Generation Machine Intelligence Algorithms*.: O'Reilly Media, 2017, pp. 85-90.

[38] I. Goodfellow, Y. Bengio, and A. Courville. (2016) Deep Learning. [Online]. http://www.deeplearningbook.org

[39] LISA Lab, "Convolutional Neural Networks," in *Deep Learning Tutorial*.: University of Montreal, 2015, pp. 54-58.

[40] W. Rawat and Z. Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review," *Neural Comput*, vol. 29, no. 9, pp. 2352-2449, September 2017.

[41] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," *ArXiv e-prints*, vol. 1803.08375, March 2018.

[42] G. Dahl, E. Sainath, and G. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8609-8613, 2013.

[43] J. Heaton, "Softmax Activation Function," in *Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks*, Tracy Heaton, Ed.: Heaton Research, 2015, p. 19.

[44] Stanford Vision Lab. (2015) ImageNet Large Scale Visual Recognition Challenge (ILSVRC). [Online]. http://www.image-net.org/challenges/LSVRC/

[45] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818-2826.

[46] O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211-252, 2015.

[47] C. Szegedy et al., "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-9, 2015.

[48] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436-444, May 2015.

[49] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1345–1359, 2010.

[50] M. Sewak, R. Karim., and P. Pujari, "Transfer Learning," in *Practical Convolutional Neural Networks*.: Packt Publishing, 2018.

[51] N. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. Tang, "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima," *Computing Research Repository*, vol. abs/1609.04836, 2016.

[52] S Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reduc-ing internal covariate shift," *ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. 37, pp. 448-456, 2015.

[53] A. Menshawy, Md. R. Karim, and G. Zaccone, "Bottlenecks," in *Deep Learning with TensorFlow*.: Packt Publishing, 2017.

[54] C. Olah, A. Mordvintsev, and L. Schubert, "Feature Visualization," *Distill*, no. 10.23915/distill.00007, November 2017.

[55] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing Systems*, vol. 2, 2014, pp. 3320--3328.

[56] R. Hetch-Nielsen, "Theory of the Backpropagation Neural Network," in *Neural Networks for Perception*, Harry Wechsler, Ed., 1992, pp. 65-93.

[57] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Proceedings of the IEEE," *Gradient-based learning applied to document recognition*, vol. 86, no. 11, pp. 2278 - 2324, November 1998.

[58] S. Ruder, "An overview of gradient descent optimization algorithms ," *Computing Research Repository*, vol. abs/1609.04747, 2016.

[59] D. Masters and C. Luschi, "Revisiting Small Batch Training for Deep Neural Networks," *Computer Vision and Pattern Recognition*, vol. arXiv:1804.07612, 2018.

[60] D. Ramos, J. Franco-Pedroso, A. Lozano-Diez, and J. Gonzalez-Rodriguez, "Deconstructing Cross-Entropy for Probabilistic Binary Classifiers," *Entropy*, vol. 20, no. 208, 2018.

[61] B. D. Ripley, *Pattern Recognition and Neural Networks*.: Cambridge University Press, January 1996, p. 354.

[62] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, and K. Müller, Eds.: Springer Berlin Heidelberg, 2012, pp. 437-478.

[63] Google. (2018) Advanced Guide to Inception v3 on Cloud TPU. [Online]. https://cloud.google.com/tpu/docs/inception-v3-advanced

[64] I. V. Tetko, D. J. Livingstone, and A. I. Luik, "Neural network studies. 1. Comparison of overfitting and overtraining," *Journal of Chemical Information and Computer Sciences*, vol. 35, no. 5, pp. 826–833, September 1995.

[65] W. S. Sarle, "Stopped Training and Other Remedies for Overfitting," *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics*, pp. 352-360, 1995.

[66] Ma. Kuhn and K. Johnson, "Over-Fitting and Model Tuning," in *Applied Predictive Modeling*.: Springer, 2013, pp. 67-69.

[67] L. Richardson, S. Ruby, and M. Amundsen, *RESTful Web Services*.: O'Reilly Media, 2013.

[68] G. James, D. Witten, T. Hastie, and R. Tibshirani, "Classification," in *An Introduction to Statistical Learning: with Applications in R*.: Springer Texts in Statistics, 2013, p. 145.

[69] T. Fawcett, "An introduction to ROC analysis," *Journal Pattern Recognition Letters - Special issue: ROC analysis in pattern recognition*, vol. 27, no. 8, pp. 861-874, June 2006.

[70] B.W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442-451, October 1975.

[71] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Mining*, vol. 10, no. 35, 2017.

[72] W. E. Damsky and M. Bosenberg, "Melanocytic nevi and melanoma: unraveling a complex relationship," *Oncogene*, vol. 36, no. 42, October 2017.

[73] D. Shitara et al., "Mutational status of nevus associated-melanomas," *British Journal of Dermatology*, vol. 173, no. 3, pp. 671-680, September 2015.

[74] A. Bhattacharya et al., "Precision Diagnosis Of Melanoma And Other Skin Lesions From Digital Images," *AMIA Summits on Translational Science Proceedings*, vol. 2017, no. 2017, pp. 220–226, 2017.

[75] H. A. Haenssle et al., "Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists," *Annals of Oncology*, vol. 29, no. 18, pp. 1836–1842, 2018.

6. ANEXOS

En el CD se encuentran adjuntos los anexos detallados a continuación:

ANEXO I Imágenes dermatoscópicas de melanoma y nevus usadas para el entrenamiento, validación y pruebas

ANEXO II Archivos con los vectores de características de las imágenes dermatoscópicas de melanoma y nevus usadas para el entrenamiento, validación y pruebas

ANEXO III Código fuente para ejecutar las tareas de entrenamiento, validación y pruebas

ANEXO IV Archivos con los valores de exactitud y error de las tareas de entrenamiento y validación de los dos escenarios

ANEXO V Archivo binario del modelo de clasificación

ANEXO VI Código fuente de la aplicación web

ANEXO VII Código fuente de la aplicación móvil

ANEXO VIII Manual para desplegar la aplicación web en AWS EC2

ANEXO IX Manual para usar la aplicación móvil