Servicio Web para la clasificación de cáncer de piel usando redes neuronales profundas

Endika Iglesias Garcia
Master universitario en Bioinformática y Bioestadística
TFM-Estadística y Bioinformática 25

Nombre Director/Consultor
Edwin Santiago Alférez Baquero
Nombre Profesor/a responsable de la asignatura
Jose Antonio Morán Moreno

Fecha entrega
05-06-2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	Servicio Web para la clasificación de cáncer de piel usando redes neuronales profundas.
Nombre del autor:	Endika Iglesias Garcia
Nombre del consultor/a:	Edwin Santiago Alférez Baquero
Nombre del PRA:	Jose Antonio Morán Moreno
Fecha de entrega (mm/aaaa):	05/2018
Titulación:	Master universitario en Bioinformática y Bioestadística
Área del Trabajo Final:	TFM-Estadística y Bioinformática 25
Idioma del trabajo:	Castellano
Palabras clave:	Servicio web, red neuronal, cancer de piel

Resumen del Trabajo (máximo 250 palabras):

El carcinoma de piel es el cáncer más común y su diagnóstico se realiza principalmente por un análisis visual. Inicialmente se realiza un análisis dermoscópico, una biopsia y un examen histopatológico. La clasificación automatizada del carcinoma de piel mediante imágenes es una tarea difícil debido a la variabilidad en la apariencia. Por ello, las redes neuronales convolucionales profundas tienen un potencial para este tipo de tareas generales y altamente variables.

Para lograrlo, se desarrollará una plataforma web accesible, donde los usuarios enviarán imágenes de áreas cutáneas y se analizarán e identificaran las posibles zonas cancerígenas encontradas, ya sean benignas o malignas. Este análisis lo realizará una red neuronal convolucional profunda. Se usará el conjunto de datos ISIC[4] que contiene imágenes clasificadas en benignas y malignas, también contiene otras clasificaciónes como el tipo de lesión que presenta la imágen.

Se realizará inicialmente un desarrollo de un modelo propio neuronal utilizando capas convolucionales. También se usaran redes pre-entrenadas como InceptionV3, VGG16 y VGG19. Se realizará un análisis preliminar, donde se podrá observar la precisión de cada una de estas redes neuronales. A continuación se evaluará la precisión de cada una de ellas para seleccionar la adecuada para realizar fine tunning y utilizar en el servicio web.

Abstract (in English, 250 words or less):

Skin cancer, the most common human malignancy, is primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by dermoscopic analysis, a biopsy and histopathological examination. Automated classification of skin lesions using images is a challenging task due to the variability in appearance. Convolutional Neural Networks have a potential for such general and highly variable tasks.

For this purpose, an accessible web platform will be developed, where users will send images of skin areas and the possible carcinogenic zones found will be analyzed and identified. This analysis will be performed by a deep convolutional neural network. We will use the ISIC[4] dataset containing images classified as benign and malignant will be used, as well as other classifications such as the type of lesion presented in the image.

Initially, a development of an own neuronal model using convolutional layers will be performed. Pre-trained networks such as InceptionV3, VGG16 and VGG19 will also be used. A preliminary analysis will be carried out, where the accuracy of each of these neural networks can be observed. The accuracy of each of these will then be evaluated to select the appropriate one for fine tuning and use in the web service.

Índice

Lista de figuras

Listado de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

El cáncer de piel es una enfermedad muy común, se diagnostica principalmente de forma visual, comenzando con un examen clínico inicial y seguido de un análisis dermatoscópico, una biopsia y un examen histopatológico.

Según el **Centro para el Control y la Prevención de Enfermedades** en el año 2014 en Estados Unidos se diagnósticaron **melanoma cutáneo** a **76.665 personas** de las cuales **9.324 personas** murieron [14]. Y según la **Organización Mundial de la Salud** las **personas** que utilizan las camas de **bronceado** antes de los 30 años son **2,5 veces** más propensas a desarrollar **carcinoma de células escamosas** y **1,5 veces** más propensas a desarrollar **carcinoma de células basales** [15]

Las redes neuronales convolucionales profundas (CNN) muestran un gran potencial para realizar tareas generales y altamente variables. Estudios como el de **Chang, Hao** publicado en 2017 (Skin cancer reorganization and classification with deep neural network) [3] demuestran que la clasificación de las lesiones cutáneas usando una red de neuronas convolucionales (CNN), entrenada de extremo a extremo a partir de imágenes directamente, utilizando solo píxeles y etiquetas de enfermedades como datos de entrada. La CNN logra un rendimiento a la par con todos los expertos probados en ambas tareas, lo que nos demuestra que una inteligencia artificial es capaz de clasificar el cáncer de piel con un nivel de competencia comparable al de los dermatólogos [2].

En este trabajo se **desarrollará** un **servicio web** público, donde los usuarios podrán enviar por medio de un sencillo formulario **imágenes cutáneas** para ser diagnosticadas por un modelo neuronal profundo. De esta forma, se pretende **facilitar el análisis** y mejorar su **detección a tiempo**.

Se utilizará la **base de datos de imágenes** ISIC [4] para **entrenar**, **probar** y **evaluar** las **neuronas**. Actualmente esta base de datos contiene:

8 conjuntos de datos diferentes, con un total de **13.786 imágenes** (50GB comprimidos).

- **MSK-1**, 1.100 imágenes, contiene lesiones melanocíticas tanto benignas como malignas. Casi todos los diagnósticos fueron confirmados por informes de histopatología; el resto consiste en lesiones benignas confirmadas por el seguimiento clínico. Las imágenes no fueron tomadas con cámaras digitales modernas.
- **MSK-2**, 1.535 imágenes, contiene lesiones cutáneas melanocíticas y no melanocíticas confirmadas por biopsia. Este conjunto de datos incluye más de 500 melanomas. Muchas de estas imágenes tienen variantes polarizadas y de contacto.
- **MSK-3**, 225 imágenes, contiene lesiones surtidas, en su mayoría nevus y carcinomas de células basales. Estas imágenes se encontraron basadas en una búsqueda no filtrada para ninguna patología particular. Todos los diagnósticos confirmados por histopatología.
- **MSK-4**, 947 imágenes, contiene imágenes encontradas en base a una búsqueda de pacientes con antecedentes personales, diagnóstico clínico o diagnóstico diferencial de melanoma. Todos los diagnósticos confirmados por histopatología.
- **MSK-5**, 111 imágenes, contiene queratosis seborreicas obtenidas de pacientes durante una visita clínica. Estas lesiones no se biopsiaron y se determinó que eran queratosis seborreicas por acuerdo de tres expertos.
- **SONIC**, 9.251 imágenes, contiene lunares en niños. Lesiones melanocíticas benignas de una población pediátrica. La naturaleza benigna de las lesiones se basa en la evaluación clínica.
- **UDA-1**, 557 imágenes, contiene lunares y melanomas. Lesiones melanocíticas fueron confirmadas por biopsia. Ambas lesiones malignas y benignas están incluidas.
- **UDA-2**, 60 imágenes, contiene lunares y melanomas. Lesiones melanocíticas fueron confirmadas por biopsia. Ambas lesiones malignas y benignas están incluidas.

Todas las imágenes descritas anteriormente, se pueden clasificar en **benigas** (12.668 imágenes) y **malignas** (1.084 imágenes)

Para completar el diagnostico se intentarán detectar las siguientes lesiones [18]:

- La **queratosis actínica** (AK, actinic keratosis), es un parche precanceroso de piel gruesa, escamosa o con costra. Por lo general, se forman cuando la piel se daña por la radiación ultravioleta del sol o las cámaras de bronceado interiores. (Número total de muestras **2 imágenes**)

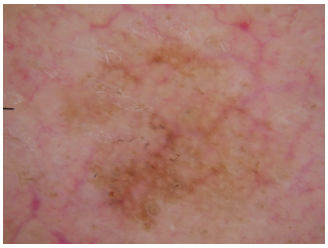

Figura 1. Ejemplo de queratosis actínica

- El **angiofibroma nasofaríngе** (angiofibroma or fibrous papule), es un tumor vascular benigno pero localmente agresivo que crece en la parte posterior de la cavidad nasal. Con mayor frecuencia afecta a varones adolescentes y puede crecer en fisuras del cráneo y puede extenderse a estructuras adyacentes. (Número total de muestras **1 imagen**)

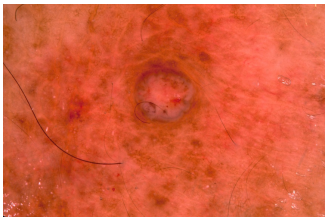

Figura 2. Ejemplo de angiofibroma nasofaríngе

- Los **angiomas** (angioma), son tumores benignos derivados de células de las paredes de vasos vasculares o linfáticos o derivados de células de los tejidos que rodean estos vasos. Son frecuentes a medida que los pacientes envejecen, pero pueden ser un indicador de problemas sistémicos como la enfermedad hepática. No están comúnmente asociados con malignidad. (Número total de muestras **15 imágenes**)

Figura 3. Ejemplo de angioma

- La **proliferación melanocítica atípica** (atypical melanocytic proliferation), se trata de la formación atípica de lunares, presentes en la epidermis, la capa más externa de la piel y es benigno. (Número total de muestras **13 imágenes**)

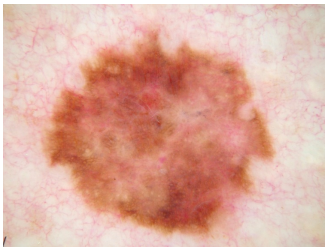

Figura 4. Ejemplo de proliferación melanocítica atípica

- El **carcinoma de células basales** (basal cell carcinoma), es la forma más frecuente de todos los cánceres (maligno). Surgen en las células basales de la piel, que recubren la capa más profunda de la epidermis. A menudo se ven como llagas abiertas, machas rojas, crecimientos rosados, protuberancias brillantes o cicatrices y generlamente son causados por una combinación de exposición solar acumulativa e intensa y ocasionalmente. (Número total de muestras **33 imágenes**)

Figura 5. Ejemplo de carciroma de células basales

- La **dermatofibroma**, es un tumor benigno muy frecuente que afecta a la piel. Aparece en adultos a partir de los 30 años, se localiza preferentemente en las extremidades inferiores y es más frecuente en mujeres. Generalmente no provoca nigún tipo de síntomas y presenta un tamaño pequeño, de menos de un centimetro de diámetro. (Número total de muestras **7 imágenes**)

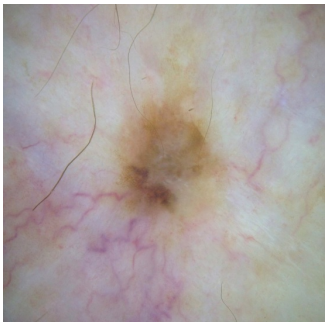

Figura 6. Ejemplo de dermatofibroma

- El **lentigo NOS**, es una pequeña mancha pigmentada en la piel con un borde claramente definido. Es una hiperplasia inofensiva (benigna). (Número total de muestras **71 imágenes**)

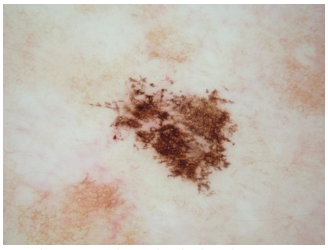

Figura 7. Ejemplo de lentigo NOS

- El **lentigo simple** (lentigo simplex), es la forma más común de lentigo. Una sola lesión o múltiples lesiones (benignas) pueden estar presentes en el nacimiento o más comúnmente se desarrollan en la primera infancia. No es inducido por la exposición al sol, y no está asociado con ninguna enfermedad o condición médica. (Número total de muestras **27 imágenes**)

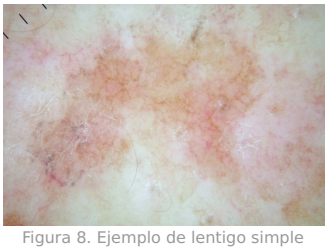

Figura 8. Ejemplo de lentigo simple

- La **queratosis liquenoide** (lichenoid keratosis), suele ser una mácula pequeña, solitaria e inflamada o una placa pigmentada delgada. También se describen múltipes queratosis liquenoide eruptivas en sitios expuestos al sol. Su color varía desde un marrón rojizo inicial a un marrón/púrpura grisáceo, es benigna. (Número total de muestras **1 imagen**)

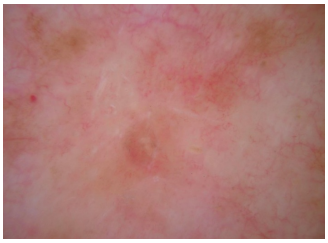

Figura 9. Ejemplo de queratosis liquenoide

- El **melanoma**, es el nombre genérico de los tumores melánicos o pigmentados. Es una grave variedad de cáncer de piel, causante de la mayoría de las muertes relacionadas con el cáncer de piel. Se trata de un tumor maligno generalmente cutáneo. (Número total de muestras **1019 imágenes**)

Figura 10. Ejemplo de melanoma

- Un **nevus** o nevo, es una proliferación de distintos tipos de células en la piel. Así, puede haber sebáceos, apocrinos (de las glándulas apócrinas de la piel), etc. Los más característicos son los nevus melanocíticos, que son proliferaciones de células pigmentadas llamadas “células névicas”, son benignas. (Número total de muestras **11861 imágenes**)

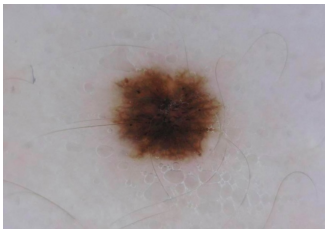

Figura 11. Ejemplo de nevus

- Una **cicatriz** (scar), es un área de tejido fibroso que reemplaza la piel normal después de una lesión (son benignas). Las cicatrices son el resultado del proceso biológico de reparación de heridas en la piel. (Número total de muestras **1 imagen**)

Figura 12. Ejemplo de cicatriz

- La **queratosis seborreica** (seborrheic keratosis), también conocida como verruga seborreica, papiloma de células basales o verruga senil. Es un tumor no canceroso(benigno) que se origina en las células de la capa externa de la piel. (Número total de muestras **419 imágenes**)

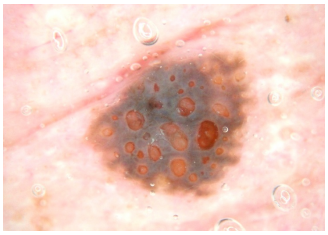

Figura 13. Ejemplo de queratosis seborreica

- El **lentigo solar** (solar lentigo), es una mancha en la piel asociada al envejecimiento y la exposición a la radiación ultravioleta del sol (benigna). (Número total de muestras **57 imágenes**)

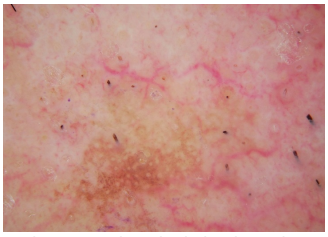

Figura 14. Ejemplo de lentigo solar

- El **carciroma de células escamosas** (squamous cell carcinoma), también conocidos como carcinoma epidermoide (maligno), son varios tipos de diferentes cánceres que resultan de células escamosas. (Número total de muestras **29 imágenes**)

Figura 15. Ejemplo de carciroma de células escamosas

Indicar que dada la escasez de imágenes disponibles por algunos tipos de lesión, se agruparán de la siguiente manera.

Lesión	Nº imágenes
Melanoma	1019
Nevus	11861
Otros	676

Tabla 1. resumen de agrupación de clasificaciones

De todo el conjunto de datos, se utilizaran **10.400 imágenes para entrenar** las neuronas, **3.341 para validar** y **150 para evaluar** (en el código fuente se definirán todos los conjuntos de datos)

1.2 Objetivos del Trabajo

Objetivo general:

- Desarrollar un servicio Web para la clasificación de cáncer de piel usando redes neuronales profundas.

Objetivos específicos:

- Preprocesar las imágenes del conjunto de datos ISIC [4], redimensionar a un tamaño fijo (512px de alto y ancho). Posteriormente, separar en dos conjuntos, uno para entrenar compuesto por 10.400 imágenes, otro para probar compuesto por 3.341 imágenes y otro para evaluar la red neuronal compuesto por 150 imágenes. Estos conjuntos serán utilizados para todos los modelos neuronales que se analizarán.
- Por cada modelo neuronal que se analizará, se desarrollaran las pruebas suficientes para obtener los valores necesarios y evaluación del mismo.
- Determinar con cual de todos los modelos neuronales se obtiene el mejor resultado, y luego realizar un proceso de fine tuning.
- Diseñar y desarrollar un servicio web, donde se podrá hacer uso de la red neuronal seleccionada para analizar imágenes cutáneas.

1.3 Enfoque y método a seguir

El aprendizaje automático o **Machine Learning** es el subcampo de las ciencias de la computación y una rama de la **inteligencia artificial** cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender.

De esta, sale a su vez una subrama, las **redes neuronales**. Estos son un modelo computacional basado en un gran conjunto de unidades neuronales simples (**neuronas artificiales**). Cada unidad neuronal está conectada con muchas otras y los enlaces entre ellas pueden incrementar o inhibir el estado de activación de las neuronas adyacentes.

Estas **redes neuronales** contienen tres grandes paradigmas, **aprendizaje supervisado**, el **aprendizaje no supervisado** y el **aprendizaje por refuerzo**. Nos centraremos en el aprendizaje supervisado. Como método de ajuste usaremos la retropropagación (back propagation), que consiste en utilizar la regla de la cadena para el cálculo del gradiente. Ése método emplea un ciclo de propagación de dos fases:

1. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas siguientes de la red, hasta generar una salida.

2. Esta salida se compara con la salida deseada y se calcula el error.

Esta técnica de **retropropagación** se utiliza concretamente en las redes neuronales convolucionales.

Las **redes neuronales convolucionales**, son un tipo de red de multiples capas, pero debido a que su aplicación es realizada mediante tensores multidimensionales, son muy efectivas para tareas de visión artificial, como en la clasificación y segmentación de imágenes.

En este proyecto, inicialmente se pre-procesarán el conjunto de datos ISIC [4]. Se puede observar que las imágenes no tienen el mismo tamaño, por lo que se redimensionarán evitando la pérdida de datos en el proceso. A continuación se definirán cúales son las imágenes que corresponderán al conjunto de datos de entrenamiento y cuales al de evaluación.

Siguiendo las buenas prácticas de desarrollo de software, se definirán pequeños **objetivos** o tareas de 4 a 7 días.

Por ejemplo, las pruebas de las diferentes arquitecturas de redes neuronales profundas que se implementarán, se realizarán en pequeñas tareas de 4 y 2 días. En este periodo se obtendrá los datos necesarios para la evaluación de los diferentes algoritmos.

Evaluación de los diferentes modelos neuronales

Las redes neuronales se evaluaran en función de los siguientes valores obtenidos:

- **Exactitud (accuracy)**, es el porcentaje de precisión/exactitud en las predicciones realizadas por el modelo neuronal. Una precisión baja es síntoma de un mal aprendizaje, por lo que será motivo de descarte.
- **Pérdida (loss)**, porcentaje de pérdida que hace referencia a la optimización del algoritmo. Dicho porcentaje no interfiere con el nivel de precisión, pero como consecuencia de un nivel alto de pérdida obtendremos un algoritmo poco optimizado y probablemente consuma más recursos de los necesarios, será lento en la predicción.
- **Tiempo de respuesta**, se obtendrá una media del tiempo de respuesta de la red neuronal profunda. Se tendrá en cuenta dicho valor. El tiempo de respuesta consiste en el tiempo que tarda en dar una respuesta la red neuronal al predecir la respuesta.
- **Recursos del sistema**, este valor solamente se tendrá en cuenta si existe el caso de indecisión entre dos modelos y se realizará fine tuning a dos de ellos. Se evaluará los requerimientos de cada modelo neuronal para tomar una decisión. Principalmente se tendrá en cuenta el consumo de CPU.

Diseño, desarrollo y publicación del servicio web

Se incluirá en la misma fase todo lo relacionado con el servicio web. Esto es debido a que consistirá en un formulario sencillo y 3 páginas visibles:

- **Página principal**, contará con información y descripción del servicio junto con el correspondiente formulario.
- **Página de resultados**, mostrará de forma clara el resultado tras el análisis realizado por la red neuronal profunda.
- **Página de privacidad**, mostrará la información obtenida por parte del usuario y el uso que se realiza.

Además el formulario pedirá únicamente el **sexo** del usuario, la **edad**, la **imagen** a analizar y su **localización** geográfica.

Nos comprometemos a proteger la privacidad y datos personales del usuario. Utilizamos sus datos personales estrictamente dentro de los límites legales establecidos por las leyes aplicables sobre protección de datos. Toda la información se almacena por razones puramente técnicas y de forma anónima.

1.4 Planificación del Trabajo

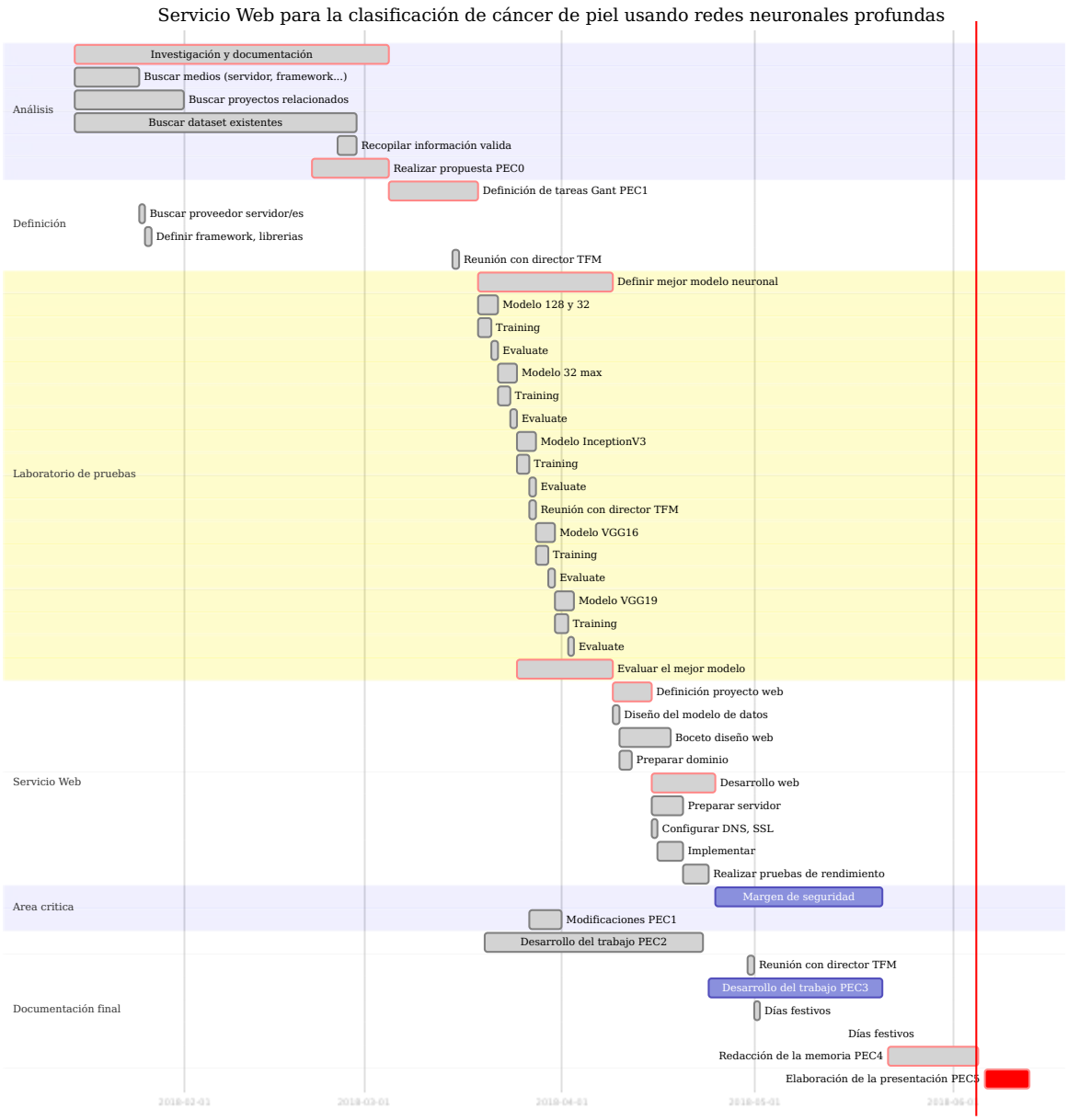

Figura 16. Diagrama de Gantt

1.4.1 Análisis (35 días, 94 horas aproximadamente)

- [Crítica] Investigación y documentación
  1. Buscar medios (servidor, framework...)
  2. Buscar proyectos relacionados
  3. Buscar dataset existentes
  4. Recopilar información valida
  5. Realizar propuesta

Esta sección contiene una tarea crítica con diferentes hitos. Se trata del proceso de tareas previas a la redacción de la propuesta del TFM. Donde se busca información y documentación relacionada con el proyecto a desarrollar.

1.4.2 Definición (10 días, 27 horas aproximadamente.)

- Definición de tareas Gant
- Buscar proveedor servidor/es
- Definir framework, librerias
- Reunión con director TFM

En esta tarea, una vez recopilada la información se procederá a investigar sobre las herramientas y los servicios externos que se usarán para llevar a cabo el trabajo final. En esta fase se define el plan de trabajo, el lenguage de progración y los diferentes marcos de trabajo (frameworks) concretamente:

- **Keras** [12] para la implementación de las Convnets.
- **Django** [11] marco de trabajo para desarrollar el servicio web.
- **Docker** [10] para la implementación de los diferentes desplieges del servicio (entorno desarrollo, entorno pruebas y entorno producción)
- **Amazon** [6] para el despliege de los servidores de desarrollo y pruebas.
- **DigitalOcean** [5] para el despliege del servidor de producción.
- **Namecheap** [7] servicio para el registro del dominio.
- **Let's Encrypt** [8] servicio para la obtención de certificados SSL.
- **PostgreSQL** [9] gestor de base de datos.

1.4.3 Laboratorio de pruebas

En la sección de laboratorio de pruebas (13 días), se definen los procesos de trabajo de cada uno de las diferentes redes neuronales profundas a desarrollar. Esta sección se detallará en otros apartados y anexos donde se reflejará los resultados de cada prueba.

1.4.4 Servicio Web (12 días, 32 horas en total)

- [Crítica] Definición proyecto web (5 días)

    1. Diseño del modelo de datos
    2. Boceto diseño web
    3. Preparar dominio

- [Crítica] Desarrollo web (7 días)

    1. Preparar servidor
    2. Configurar DNS, SSL
    3. Implementar
    4. Realizar pruebas de rendimiento

En esta sección se definen dos tareas criticas, **definición proyecto web** y **desarrollo web**.

Definición proyecto web:

- Nombre del dominio web.
- Estructura de la base de datos, número de tablas, campos y su relación.
- Estructura del proyecto, modelo, vista, controlador. Definición de las clases, funcionalidades.
- Documento de privacidad, redactar documento de privacidad de usuario y los datos almacenados.
- Almacenamiento de datos, securizar accesos.
- Diseño web.

Desarrollo web:

- Desarrollar area frontal de la web (frontend).
- Crear y configurar la base de datos.
- Desarrollar area interna de la web (backend).
- Crear y configurar servidor.
- Configurar resto de servicios DNS, SSL, Nginx.
- Realizar pruebas de estrés al servidor para evaluar rendimiento.

Hay un margen denominado **Area crítica** (15 días, 40 horas aproximadamente.) el cual se utilizará para realizar las tareas que se han ido retrasando en el tiempo, por problemas técnicos, falta de recursos y planificaciones erróneas.

Por último, la sección **Documentación final** (40 días, 108 horas aproximadamente) refleja las dos últimas tareas previas a la presentación del TFM:

- Redacción de la memoria.
- Elaboración de la presentación.

1.5 Gestión de riesgos

En esta sección evaluamos diferentes tipos de riesgos y como manejarlos:

1. Problemas en la definición del trabajo a realizar

La definición de las tareas que se ejecutarán para el desarrollo de este proyecto es un punto muy importante antes de ponerse a desarrollar. La incorrecta definición del trabajo puede desencadenar un exceso del tiempo empleado en tareas que no se definieron o previeron.

Gravedad Alta: se trata de un riesgo muy alto ya que supone la no finalización correcta del trabajo.

Solución: disponer de un tiempo extra para imprevistos o problemas. Con el fin de mitificar este riesgo se tendrá un análisis previo recopilando fuentes y documentación necesaria para asegurarnos de tener todos los conocimientos necesarios para afrontar imprevistos.

2. Recursos hardware insuficientes.

Las redes neuronales profundas necesitan de un gran volumen de información y como consecuencia un gran número de recursos como GPU, RAM y disco. La no disponibilidad o la limitación de estos recursos desencadenará en la imposibilidad de entrenar correctamente la red neuronal profunda.

Gravedad Media: se trata de un problema medio debido a la alta gama y disponibilidad de recursos que existen online. Se realizaría un coste mayor de lo esperado.

Solución: realizar pruebas, simulando la carga de datos que vamos a usar, esto ayudará a tener una previsión acertada y sin sobrecostes.

3. Compromiso con el proyecto

Se realizarán tareas definidas durnate el tiempo estipulado. La disponibilidad para ejecutar cada una de las tareas definidas es crucial para no tener retrasos. Por otro lado, las malas prácticas tanto de trabajo como de código pueden agravar la situación.

Gravedad Alta: Retrasarnos en la finalización de las tareas puede acarrear en graves modificaciones en el diagrama de Gantt.

Solución: La organización y definición de los tiempos de trabajo y aplicar las buenas prácticas en el código es crucial. Desarrollando test unitarios en nuestro software minimiza el riesgo.

1.6 Gestión de gastos

A continuación se presentan la previsión de gastos de cada servicio a usar.

- Amazon Web Service (AWS)

AMI: Deep Learning AMI (Ubuntu) Version 5.0 - **ami-0ebac377**

Instancia: p2.xlarge GPU compute, 4CPUs, RAM 61Gigas.

Precio: $0.972 por hora

Tiempo estimado de uso: 130 horas estimación calculada, partiendo de un maximo de 18 horas por red neuronal profunda entrenada.

Precio total estimado: $126,36

- Namecheap

Precio del dominio .net: 10,51€/año

- DigitalOcean

Descripción del servidor: RAM 8GB, 4vCPUs, disco SSD 160GB, transferencia de datos mensual 5T

Precio: $40/mensual

- Otros gastos

Precio Análisis: 20€/hora, se ha estimado un total de 94 horas de análisis.

Precio Diseño: 5€/hora, se ha estimado un total de 10 horas de diseño.

Precio Desarrollo: 18€/hora, se ha estimado un total de 22 horas de desarrollo.

- Resumen

Para el resto (librería Keras, Docker, Django, SSL y conjunto de datos ISIC) no se necesita licencias o pagos previos.

Actualmente el cambio de **$** a **€** es **0,806**. El gasto total estimado es **2.478,36€**

Servicio	Cantidad	Precio	Total
Amazon Web Service	1	$126,36	101,85€
Namecheap	1	10,51€/año	10,51€
DigitalOcean	1	$40/mensual	40€
Análisis	94 horas	20€/hora	1.880€
Diseño	10 horas	5€/hora	50€
Desarrollo	22 horas	18€/hora	396€
		Total	**2.478,36€**

Tabla 2. resumen de estimación de gastos

1.7 Breve sumario de productos obtenidos

Se obtendrán los siguientes productos:

- Se publicará un servicio web operativo, donde se podrá realizar análisis cutáneos online.
- Una memoria en la que se detallarán todos los procesos y resultados durante la fase de desarrollo de pruebas.
- Se incluirá una copia del proyecto alojado en el servidor donde se podrá reproducir todas las tareas(entrenamiento red neuronal, análisis)

2. Desarrollo de modelos neuronales

A continuación se detallarán los pasos seguidos para ejecutar las pruebas con los algoritmos pre-entrenados más utilizados **VGG16**, **VGG19** y **InceptionV3**. Además de estos, se desarrollará un modelo personalizado utilizando redes convolucionales. Finalmente se compararán los resultados.

Breve introducción a los modelos neuronales convolucionales

En este apartado se realiza una breve introducción a las redes neuronales convolucionales [19] también conocidas como **convnets**, un tipo de modelo de aprendizaje profundo muy utilizado en aplicaciones de reconocimiento visual.

Es importante destacar que un convnet toma como entrada los **tensores** de la siguiente forma `(image_height, image_width, image_channels)`, durante el desarrollo del proyecto vamos a trabajar con convnet donde tendran tendranla misma dimensión de tensor `Conv2D`.

Funcionamiento de una convnets

Una convolución funciona **deslizando** ventanas, en el caso del modelo Inception V3 estas tienen un tamaño de 3×3 sobre el mapa de entrada 3D. Por cada ubicación del mapa de entrada se va extrayendo un parche 3D de las características circundantes en el caso de Inception v3 tiene capas que extraen parches de 5×5, en niveles más profundo de la red convoluciona se extrae parches de 1×1. Hasta llegar a un vector de salida 1D, en el caso de Inception v3 es `(1, 1, 1000)`.

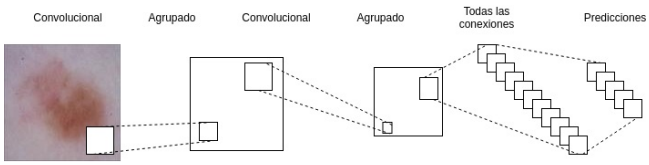

Figura 17. Ejemplo de funcionamiento de una convnet.

Debido a que relativamente hay pocas muestras de entrenamiento (Melanoma 1.019, ver Tabla 1.), el sobreajuste será la principal preocupación. Hay técnicas de sobreajuste como como `Dropout` y el aumento de datos.

Con esta técnica de **sobreajuste**, se puede mejorar el entrenamiento de un modelo. Con datos infinitos, el modelo estaría expuesto y nunca se sobreajustaría. El **aumento de datos** se basa en la generación de más datos para el entrenamiento a partir de muestras de entrenamiento existentes, este aumento de muestras se realiza a través de una serie de transformaciones aleatorias que generan imágenes de aspecto creíble. El objetivo de esta técnica es que en el entrenamiento, el modelo neuronal nunca vea exactamente la misma imagen dos veces. Esto ayuda a exponer el modelo a más aspectos de los datos y a generalizar mejor.

Veamos un ejemplo de aumento de datos usando `ImageDataGenerator`

```
datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

img = image.load_img('demo001.jpg', target_size=(32, 32))
x = image.img_to_array(img)
x = x.reshape((1, ) + x.shape)
batch = datagen.flow(x, batch_size=1)[0]
img = image.array_to_img(batch[0])
new_path = '{}gen_demo001.jpg'.format(path)
img.save(new_path)
```

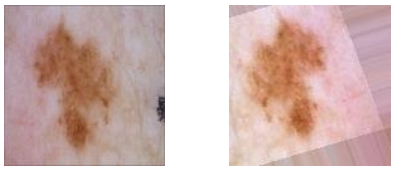

Figura 18. Ejemplo de figura original y figura transformada.

2.1 Modelo convolucional personalizado

En esta sección se presenta tres modelos neuronales, estos tres modelos es el resultado de todos los desarrollos realizados usando la librería Keras y las redes neuronales convolucionales. De todas las pruebas, nos quedamos con los modelos que se obtienen los mejores resultados. Antes de nada indicar que todos los scripts mencionados estarán en el repositorio de GitHub del TFM (https://github.com/endika/skin-cancer-TFM). Y como ya se indicó en el punto **1.1** el conjunto de datos que se utilizará **10.400 imágenes para entrenar** las neuronas, **3.341 para validar** y **150 para evaluar**.

2.1.1 Modelo 128

Preprocesamiento de las imágenes

En el siguiente modelo se han tomado 128x128 pixeles como tamaño de entrada para las imágenes. Se ha definido el anterior valor debido a la variedad de tamaños que se observan en las imágenes del conjunto de datos. El código fuente se encuentra en la ruta **dataset/raw/process.py**. El script genera también en un fichero con la información relevante que se va a usar para entrenar la neurona. En este caso, por cada una de las imágenes, se almacena el estado **benigno** o **maligno**. En el directorio **dataset/process/** se encuentra todo el contenido que se va usar para entrenar los modelos neuronales.

Descripción del modelo neuronal convolucional

A continuación vamos a proceder a realizar un resumen breve del funcionamiento del modelo neuronal:

- De **entrada** tenemos una capa **convolucional**, esta tiene un tensor 3D `(128, 128, 3)`.
- En las **capas ocultas**, contiene tres capas **convolucionales**, una vez procesada la información, agrupa lo máximo posible los datos y desecha aleatoriamente una fracción (25%) de las unidades. Este último paso ayuda a prevenir el sobreajuste de la neurona.

- ***La capa de salida***, aplana los datos que recibe y por medio del activador `sigmoid` retorna el resultado final.

A continuación podemos ver una muestra del código descrito anteriormente.

```
model = Sequential()
# First layer (input layer)
model.add(
    Conv2D(32, (3, 3), padding='valid', input_shape=(128, 128, 3))
)
model.add(Activation('relu'))

# Second layer
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Third layer
model.add(Conv2D(32, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Output layer
model.add(Flatten())
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(len(self.categories) + 1))
model.add(Activation('sigmoid'))

sgd = SGD(lr=1e-6, momentum=0.9)
model.compile(
    loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy']
)
```

Figura 19. fragmento del código modelo neuronal convolucional personalizado.

En la Figura 41 del Anexo se muestra el diagrama del flujo de trabajo de la red neuronal.

Resultados

A continuación, mostramos un resumen de los valores de **pérdida** y **exactitud** obtenidos en la validación del modelo neuronal con los valores del conjunto de datos seleccionados en el punto 1.1. También, añadiremos el resto de valores que se van a usar para comparar las redes neuronales.

Pérdida: 0.0296

Exactitud: 0.9997

Tamaño del modelo neuronal: 7,4M

Tiempo de respuesta (predicción): 1 segundo.

Recursos del sistema: 1 CPU y 2G de RAM. (Se tomará como referencia la máquina con menor recurso en la que la red neuronal a funcionado correctamente.)

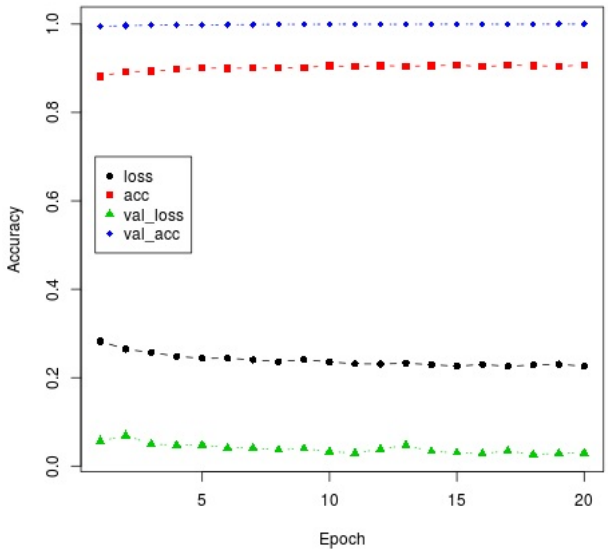

Figura 20. Gráfico muestra los valores loss y accuracy por cada epochs.

2.1.1 Modelo 32

Preprocesamiento de las imágenes

En el siguiente modelo vamos modificar el tensor, para ello se ha tomado 32×32 pixeles como tamaño de entrada para las imágenes. Se ha definido un tamaño más pequeño que el anterior para poder ajustar la neurona. El código fuente se encuentra en la ruta **dataset/raw/process_32.py**.

Descripción del modelo neuronal convolucional

Para este modelo no vamos a realizar resumen del funcionamiento. Es igual que el anterior, solamente modificamos el tensor `(32, 32, 3)` .

A continuación podemos ver una muestra del código.

```
model = Sequential()
# First layer (input layer)
model.add(Conv2D(32, (3, 3), padding='valid', input_shape=(32, 32, 3)))
model.add(Activation('relu'))

# Second layer
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Third layer
model.add(Conv2D(32, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Output layer
model.add(Flatten())
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(len(self.categories) + 1))
model.add(Activation('sigmoid'))

sgd = SGD(lr=1e-6, momentum=0.9)
model.compile(
    loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy']
)
```

Figura 21. fragmento del código modelo neuronal convolucional personalizado.

En este caso no mostraremos el diagrama de flujo de la red neuronal ya que es exactamente igual que la anterior (ver Figura 42.)

Resultados

Pérdida: 0.0435

Exactitud: 0.9996

Tamaño del modelo neuronal: 572K

Tiempo de respuesta: 1 segundo.

Recursos del sistema: 1 CPU y 2G de RAM.

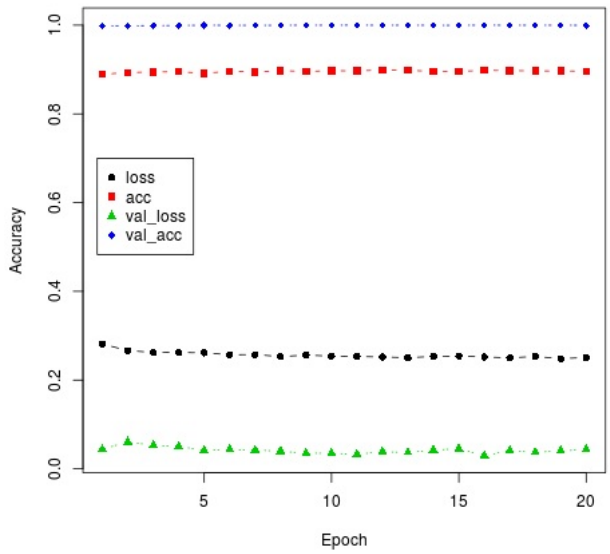

Figura 22. Gráfico muestra los valores loss y accuracy por cada epochs.

2.1.1 Modelo 32 max.

Preprocesamiento de las imágenes

Para el siguiente modelo no es necesario ningún tipo de transformación. Se usaran las imágenes del modelo anterior de 32 pixeles.

Descripción del modelo neuronal convolucional

A continuación vamos a proceder a describir cada una de las capas que interfieren en el modelo neuronal:

- De **entrada** tenemos una capa **convolucional**, esta tiene un tensor 3D `(32, 32, 3)`.
- En las **capas ocultas**, contiene cinco capas **convolucionales**, una vez procesada la información, agrupa lo máximo posible los datos y desecha aleatoriamente una fracción (25%) de las unidades.
- ***La capa de salida***, aplana los datos que recibe y por medio del activador `softmax` retorna el resultado final.

A continuación podemos ver una muestra del código.

```
model = Sequential()
# First layer (input layer)
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Second layer
model.add(Conv2D(32, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Third layer
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Fourth layer
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Output layer
model.add(Flatten())
model.add(Dense(3))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(len(self.categories) + 1))
model.add(Activation('softmax'))

sgd = SGD(lr=1e-6, momentum=0.9)
model.compile(
    loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy']
)
```

Figura 23. fragmento del código modelo neuronal convolucional personalizado.

En la Figura 43 del Anexo se muestra el diagrama del flujo de trabajo de la red neuronal.

Resultados

Pérdida: 0.2558

Exactitud: 0.9609

Tamaño del modelo neuronal: 328K

Tiempo de respuesta: 1 segundo.

Recursos del sistema: 1 CPU y 2G de RAM.

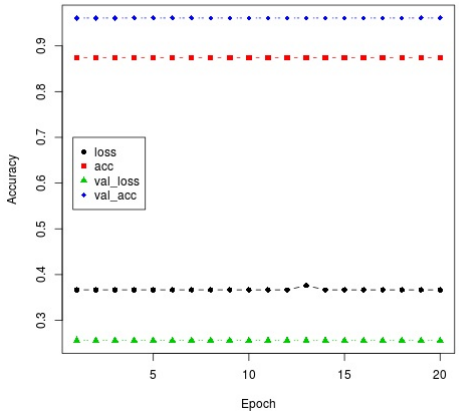

Figura 24. Gráfico muestra los valores loss y accuracy por cada epochs.

2.2 VGG16

Preprocesamiento de las imágenes

En este modelo tenemos que modificar las imágenes ya que esté modelo esta limitado a un tamaño de tensor, 224×224 píxeles.

Descripción del modelo neuronal convolucional

VGG16 se trata de otra red neuronal convolucional pre-entrenada y capacitada para el reconocimiento visual. Se usará esta red neuronal para intentar clasificar las imágenes del conjunto de datos.

Para comenzar modificamos la capa de salida para que esta solo retorne 3 posibles valores. A continuación podemos ver una muestra del código.

```
base_model = VGG16(
    weights="imagenet",
    include_top=False,
    input_tensor=Input(shape=(224, 224, 3))
)
x = base_model.output
x = Flatten(input_shape=base_model.output_shape[1:])(x
x = Dense(32, activation='relu')(x)
preds = Dense(3, activation='softmax')(x)
model = Model(input=base_model.input, output=preds)
opt = SGD(lr=.01, momentum=.9)
model.compile(
    optimizer=opt,
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Figura 25. fragmento del código VGG16

A continuación, se muestra el diagrama del flujo de trabajo de la red neuronal.

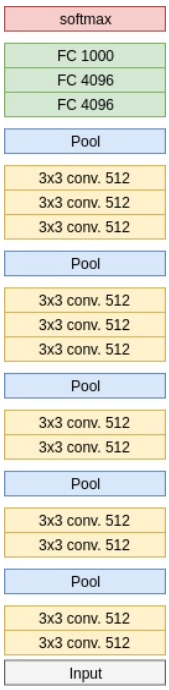

Figura 26. modelo neuronal VGG16

Resultados

Pérdida: 0.9769

Exactitud: 0.9394

Tamaño del modelo neuronal: 119M

Tiempo de respuesta: 4 segundos.

Recursos del sistema: 4 CPU y 8G de RAM.

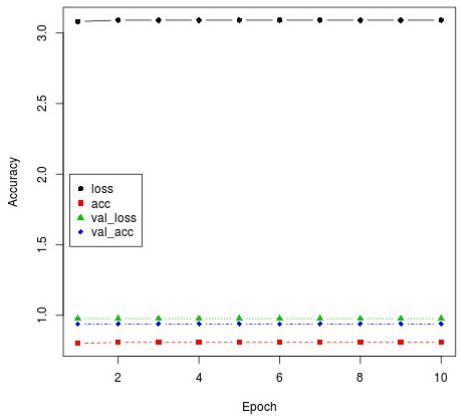

Figura 27. Gráfico muestra los valores loss y accuracy por cada epochs.

2.3 VGG19

Preprocesamiento de las imágenes

El tensor de entrada es el mismo que el de la versión anterior VGG16.

Descripción del modelo neuronal convolucional

VGG19 se trata de otra red neuronal convolucional pre-entrenada, es una versión mejorada de VGG16. Esta red neuronal se usará para intentar clasificar las imágenes del conjunto de datos.

Para comenzar modificamos la capa de salida para que esta solo retorne 3 posibles valores. A continuación podemos ver una muestra del código.

```
base_model = VGG19(
    weights="imagenet",
    include_top=False,
    input_tensor=Input(shape=(224, 224, 3))
)
x = base_model.output
x = Flatten(input_shape=base_model.output_shape[1:])(x
x = Dense(32, activation='relu')(x)
preds = Dense(3, activation='softmax')(x)
model = Model(input=base_model.input, output=preds)
opt = SGD(lr=.01, momentum=.9)
model.compile(
    optimizer=opt,
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Figura 28. fragmento del código VGG19

A continuación, se muestra el diagrama del flujo de trabajo de la red neuronal.

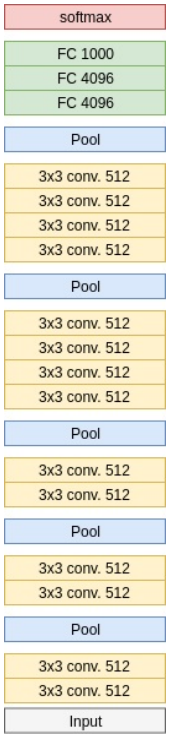

Figura 29. modelo neuronal VGG19

Resultados

Pérdida: 15.1758

Exactitud: 0.0585

Tamaño del modelo neuronal: 160M

Tiempo de respuesta: 4 segundos.

Recursos del sistema: 4 CPU y 8G de RAM.

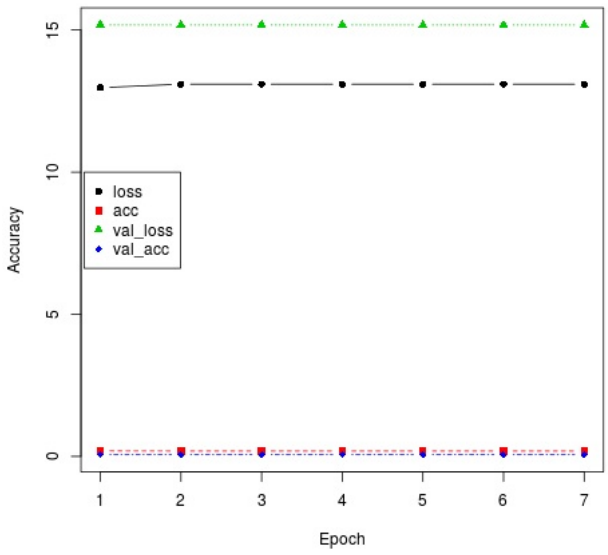

Figura 30. Gráfico muestra los valores loss y accuracy por cada epochs.

2.4 Inception V3

Preprocesamiento de las imágenes

En este caso se vuelve a modificar el tensor por lo que tenemos que transformar las imágenes a 299×299 píxeles.

Descripción del modelo neuronal convolucional

InceptionV3 se trata de una red neuronal convolucional pre-entrenada y capacitada para el reconocimiento visual. Se usará esta red neuronal para intentar clasificar las imágenes del conjunto de datos [21].

Para comenzar modificamos la capa de salida para que esta solo retorne 3 posibles valores. A continuación podemos ver una muestra del código.

```
base_model = InceptionV3(
    weights='imagenet',
    include_top=False,
    input_tensor=Input(shape=(299, 299, 3))
)
x = base_model.output
x = AveragePooling2D(pool_size=(8, 8))(x)
x = Dropout(.4)(x)
x = Flatten()(x)
predictions = Dense(
    3,
    init='glorot_uniform',
    W_regularizer=l2(.0005),
    activation='softmax'
)(x)

model = Model(input=base_model.input, output=predictions)

opt = SGD(lr=.01, momentum=.9)
model.compile(
    optimizer=opt,
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Figura 31. fragmento del código InceptionV3

A continuación, se muestra el diagrama del flujo de trabajo de la red neuronal.

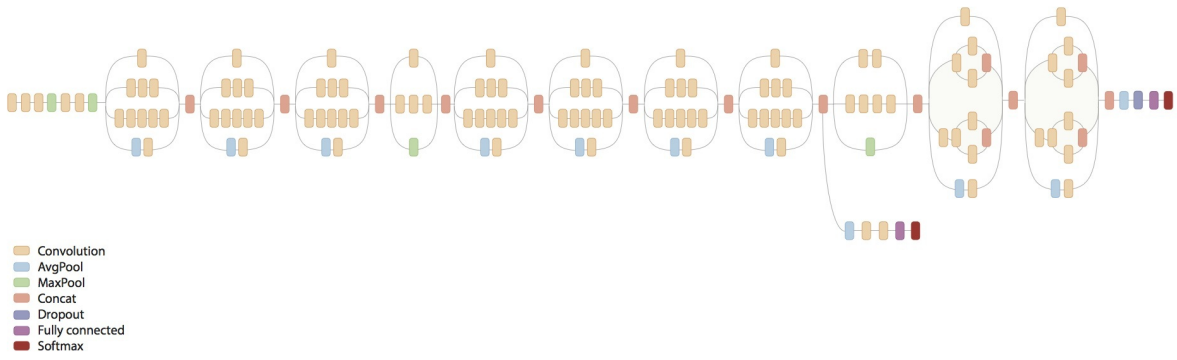

Figura 32. modelo neuronal InceptionV3, fuente Google IA Blog [21]

Resultados

Pérdida: 0.3076

Exactitud: 0.9353

Tamaño del modelo neuronal: 168M

Tiempo de respuesta: 4 segundos.

Recursos del sistema: 4 CPU y 8G de RAM.

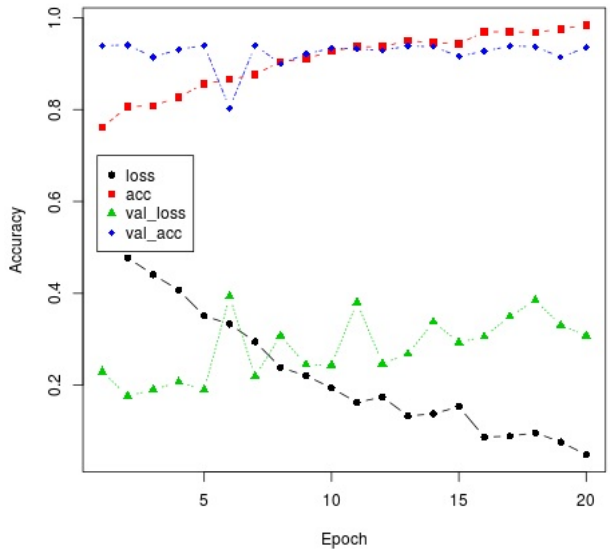

Figura 33. Gráfico muestra los valores loss y accuracy por cada epochs.

3. Análisis y comparación

En esta sección se muestran los resultados obtenidos por medio de los entrenamientos y las evaluaciones de los modelos de las redes neuronales.

- **Modelo personalizado 128**, con una precisión del **93,94%** no se obtienen buenos resultados. En el gŕafico se puede observar que tiene un sobre ajuste muy alto con lo cual no es suficiente como para obtener resultados fiables.
- **Modelo personalizado 32**, el cambio de usar un tamaño de imagen menor mejora la exactitud a **99,96%**. Pero volvemos a obtener un sobreajuste demasiado alto.
- **Modelo personalizado 32 max**, alterando las capas la red neuronal se consigue una exactitud de **96,09%**. Seguimos teniendo un sobreajuste alto.
- **VGG16**, con este algoritmo pre-entrenado se obtiene una exactitud de **93,94%**. Se observa un sobreajuste alto como en los casos anteriores.
- **VGG19**, con esta versión del algoritmo pre-entrenado se obtiene una exactitud de **5,85%**, con muy alto sobreajuste.
- **Inception V3**, con este modelo neuronal pre-entrenado se obtiene una exactitud de **93,53%**. Podemos observar en la 9 iteración que el modelo se ajusta correctamente.

En la siguiente tabla se agrupan los datos más relevantes obtenidos en la validación del modelo.

Red Neuronal	Pérdida (val_loss)	Exactitud (val_acc)	Recursos del sistema
Personalizado 128	0.0296	0.9997	1 CPU y 2G de RAM
Personalizado 32	0.0435	0.9996	1 CPU y 2G de RAM
Personalizado 32 max	0.2558	0.9609	1 CPU y 2G de RAM
VGG16	0.9769	0.9394	4 CPU y 8G de RAM
VGG19	15.1758	0.0585	4 CPU y 8G de RAM
Inception V3	0.3076	0.9353	4 CPU y 8G de RAM

Tabla 4. resumen resultados modelos neuronales

Se puede observar que con el algoritmo **Inception V3** se obtiene los mejores resultados. No se observa sobre ajuste y con el obtenemos un porcentaje de exactitud muy alto **93,53%**. Por esta razón la arquitectura de red seleccionada para el servicio web será Inception V3.

Lesiones cutáneas, análisis y evaluación

Partiendo del algoritmo anteriormente seleccionado, se pondrá a prueba para detectar lesiones cutáneas.

A continuación se muestra una tabla resumiendo el total de imágenes que hay en el conjunto de datos y la lesión a la que pertenece.

Nombre lesión	Número total de imágenes
queratosis actínica	2
angiofibroma nasofaríngе	1
angiomas	15
proliferación melanocítica atípica	13
carciroma de células basales	33
dermatofibroma	7
lentigo NOS	71
lentigo simple	27
queratosis liquenoide	1
melanoma	1.019
nevus	11.861
cicatriz	1
queratosis seborreica	419
lentigo solar	57
carciroma de células escamosas	29

Tabla 5. listado de lesiones cutáneas y numero total de imágenes.

Como se puede observar en la tabla, el numero de imágenes por cada tipo de lesión es muy bajo. Para probarlo se ha desarrollado un modelo neuronal usando **inception v3** disponible en el directorio **/reports/management/commands/training_i_diagnosis.py**. Por supuesto, hay muy pocos datos en algunas clases y además están desbalanceados.

Los resultados obtenidos son los esperados. La red neuronal no es capaz de clasificar con precisión cada uno de los tipos de lesión. Se logra mayor exactitud en la clasificación con la lesión cutánea **nevus** (lunares), por su mayor numero de imágenes. Pero aun así no lo suficiente.

Aumento de datos y fine-tuning

En el siguiente apartado se realizan técnicas de aumento de datos y fine tuning para mejorar el modelo neuronal `Inception V3`. En el apartado anterior se descartó el uso de este modelo neuronal para detectar lesiones cutáneas. Antes de descartarlo vamos a aplicar la técnica de aumento de datos usando `ImageDataGenerator`.

Y tal como se menciona en la Tabla 1 vamos a agrupar las lesiones en 3 clases **melanoma** (1.019 imagenes), **nevus** (11.861 imagenes) y otros (676 imagenes). Se aumentará el conjunto de datos de **otros** a **1.000** y los otros

dos los submuestramos a **1.000** imágenes. Usaremos en total **2.250** para entrenar, **750** para testear y el resto **10.880** para evaluar el modelo neuronal.

También vamos a aplicar `fine-tuning` para obtener el mejor resultado posible. El script completo se encuentra en **/reports/management/commands/training_i_mno.py**

A continuación se aplica el mismo método de aumento de datos y fine-tuning para entrenar un modelo neuronal que clasifique las imágenes **benign** y **malignant**. Al igual que el caso anterior sumuestramos a **1.000** así se tendrá **2.000** imágenes, de las cuales **1.500** se usaran para entrenar el modelo neuronal, **500** para validarlo y el resto de las imágenes **11.752** para evaluar.

El script completo se encuentra en **/reports/management/commands/training_i_bm.py**

A continuación se muestra una tabla con los resultados de ambos experimentos.

Red Neuronal	Pérdida (val_loss)	Exactitud (val_acc)	Recursos del sistema
Inception V3 Benign & Malignant	0.0950	0.9808	4 CPU y 8G de RAM
Inception V3 Melanoma, Nevus and Others	1.1079	0.7267	4 CPU y 8G de RAM

Tabla 6. resumen resultados de Inception v3

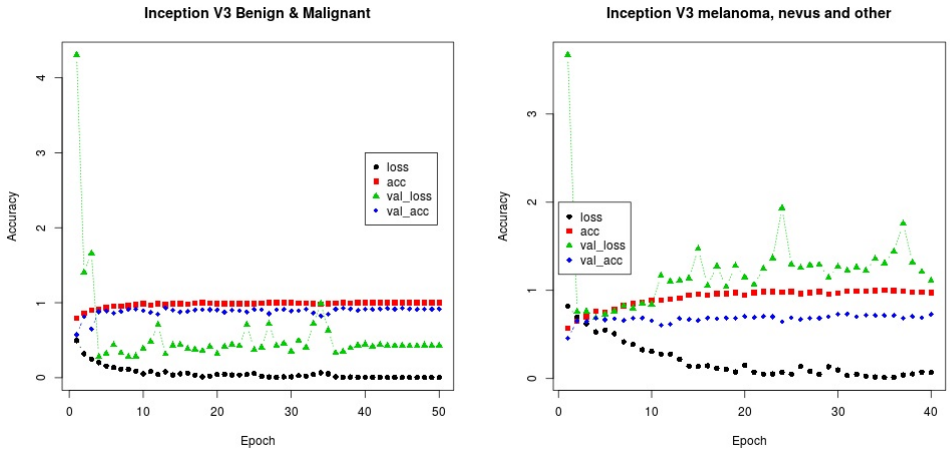

Figura 34. Gráfico muestra los valores loss y accuracy por cada epochs.

Según las evaluaciones de cada uno de los modelos, se obtiene mayor precisión despues de aplicar las técnicas de aumento de datos y fine-tuning

4. Servicio Web

En el siguiente apartado se mostrará la plataforma web desarrollada, y como utilizarla. Se mostraran los diferentes diagramas de la arquitectura del servicio web.

Contenedores Docker

El servicio contiene 3 contenedores Docker.

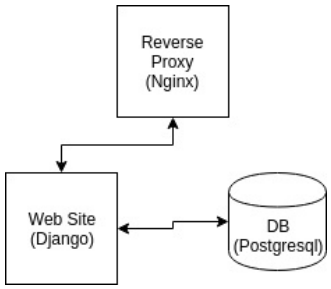

Figura 35. Diagrama de contenedores Docker que forman la plataforma

- **Nginx**: Este contenedor únicamente está configurado para realizar la tarea de proxy inverso. Se encarga de mantener actualizada el certificado SSL (Let's Encrypt [8]) y para que todas las comunicaciones exteriores se realicen por el protocolo seguro **HTTPS**.
- **Django**: Este contiene toda la funcionalidad del servicio web. Se encarga de generar y servir todas las páginas y contenidos. Es el único servicio que se comunica con el proxy inverso para recibir o enviar contenido.
- **Postgresql**: Su principal función es almacenar todo el contenido analizado previamente por la plataforma web (Django), y servir dicha información cuando está la requiera. No tiene acceso a internet y únicamente el contenedor web que hace de intermediario es el único que tiene acceso.

Caso de uso

En la Figura 36 se muestra como funciona paso a paso el servicio web.

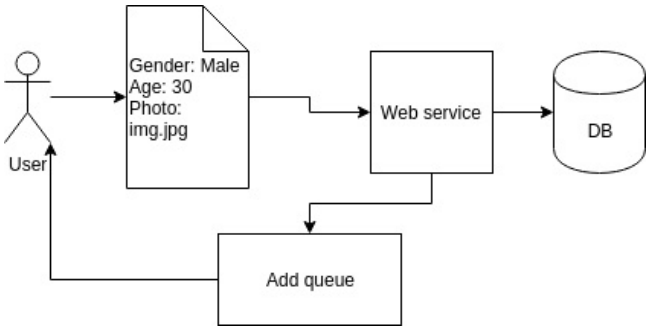

Figura 36. Diagrama caso de uso, usuario rellena formulario

Cuando un usuario accede al sitio web `cancer.deeplk.net` y rellena el formulario que muestra la web.

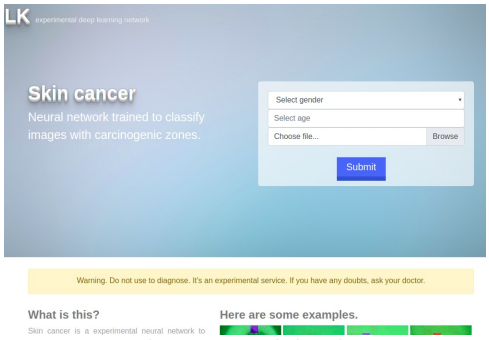

Figura 37. Formulario web

Automáticamente la información se almacena en la base de datos y le retorna al usuario una posición en la cola de espera.

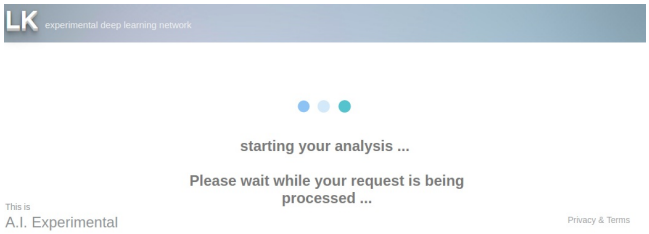

Figura 38. Mensaje que se muestra cuando sus datos comienzan a ser analizados.

Mientras el usuario está en la cola de espera, se le informa de el numero de usuario que hay pendientes delante de el. Cuando su turno llega se le informa que sus datos están siendo analizados, también se le informa durante todo el proceso para que la espera sea mucho más amena.

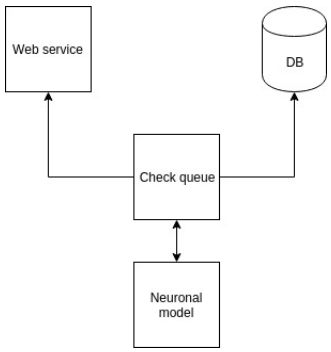

Figura 39. Diagrama gestión cola de espera.

Como se puede observar en la Figura 39, dentro del contenedor web existe un servicio escuchando todas las tareas que entran. Este se comunica directamente con la base de datos para obtener toda la información proporcionada por el usuario y de forma ordenada, procesa y envía la información al modelo neuronal.

Los modelos neuronales analizan la imagen completa y posteriormente se trocean en fragmentos de 32 pixeles para analizarlos individualmente. Estos fragmentos analizados sirven para identificar en la imágen resultante las areas más afectadas.

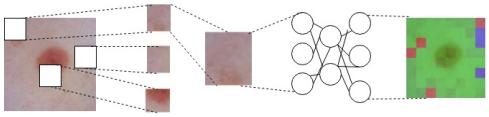

Figura 40. Proceso de análisis de imágenes.

Al recortar la imagen aumentamos los trozos para que pueda ser analizado por la red neuronal, y en función de su resultado se colorea el área de **rojo** si es maligno, **verde** si es benigno, o **azul** si es un área desconocía. Estos fragmentos serán añadidos sobre la imagen original en una capa transparente. Cada area será más o menos translúcida en función del porcentaje del resultado.

Cuando este a terminado de procesar toda la información el servicio se encarga de avisar al servicio web para que este redirija al usuario a la página de resultados que le corresponde.

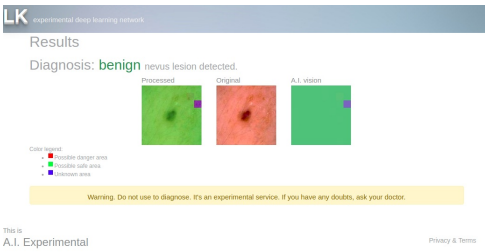

Figura 41. Informe de los datos analizados.

En ese apartado el usuario podrá evaluar si esta o no conforme con el resultado obtenido por el modelo neuronal. También tiene la posibilidad de eliminar dicho informe, si así lo desea.

5. Conclusiones

Tras el desarrollo de varios modelos, queda reflejado que con las redes neuronales convolucionales profundas pre-entrenadas, se obtienen mejores resultados que al entrenarlas para clasificación de lesiones cutáneas. Partir de cero y desarrollar un modelo convolucional, lleva mucho más tiempo y se necesitan muchos más datos si se quiere obtener los mismos resultados que un modelo pre-entrenado.

En la elaboración del TFM se encuentran estudios ya realizados basados en el mismo conjunto de imágenes [3]. La principal motivación de llevarlo acabo, ha sido en trasladar el estudio de clasificación de imágenes de cáncer de piel [3] a la parte más práctica, integrándolo en un servicio web donde se pueda dar uso de ello.

Este trabajo ha obtenido un buen ajuste del modelo. También, a pesar de los bajos recursos que hay en el servidor web, se ha intentado buscar la mejor salida para reducir el tiempo de espera. Para ello, se ha desarrollado un servicio de colas que no sobrecarga el servidor, manteniendo la sesión del modelo neuronal siempre activo y bajando la carga de trabajo ejecutándolo solamente si este tiene alguna tarea pendiente en la cola. También está limitado el envío de imágenes a 1. De esta forma si existe más de un usuario en la cola de espera, no lo bloquea demasiado tiempo.

En este trabajo se ha realizado con éxito las siguientes tareas:

- Se ha **analizado** el **conjunto de datos** obtenido de ISIC.
- Con estos datos, se han **desarrollado** varios **modelos neuronales** de los cuales se han selecionado 3. Además de los modelos pre-entrenados **VGG16, VGG19** y **Inception V3**.
- Para cada uno de los 6 modelos seleccionados, se ha **preprocesado** el **conjunto de datos** ajustando el tamaño por cada modelo neuronal.
- Cada modelo se ha **entrenado** y **evaluado** por separado y en conjunto.
- Se ha escogido el mejor modelo (**Inception V3**), y se ha ajustado lo mejor posible usando técnicas de **aumento de datos** y **fine-tuning**.
- Se ha **diseñado** y **desarrollado** un **servicio web**, preparado para usar la red neuronal seleccionada anteriormente.
- Se ha creado y configurado un **servidor**, dando de alta el **dominio**, configurando las **DNS** y generando el certificado **SSL** para su correcto funcionamiento.

5.2. Perspectivas futuras

- Aumentar los recursos hadware del servicio web para mejorar el tiempo de respuesta de los análisis. Ahora mismo se encuentra limitado para evitar costes inecesarios.

- Una vez que tenemos los recursos hadware suficientes, permitir envíar más de una imagen a analizar. De esta forma podemos cubrir el caso de no tener que envíar una a una las imágenes.

6. Glosario

Algoritmo

Conjunto de operaciones ordenadas que permite realizar un cálculo o tarea.

Dominio

Extensión o dominio de Internet es un nombre único que identifica a un sitio web en Internet.

DNS

El sistema de nombres de dominio (Domain Name System) es un sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP como Internet o una red privada.

Modelo

Termino usado en Machine Learning, el modelo representa el algoritmo de aprendizaje, y este se usa para realizar las predicciones.

Red neuronal

Modelo computacional basado en un gran conjunto de unidades neuronales simples, de forma aproximadamente análoga al comportamiento observado en los axones de las neuronas en los cerebros biológico.

Red neuronal profunda

Cuando mencionamos red neuronal profunda se hace referencia al **aprendizaje profundo**. El aprendizaje profundo es parte de un conjunto más amplio de métodos de aprendizaje automático basados en representaciones de datos.

SSL

Capa de puertos seguros (Secure Sockets Layer) protocolo criptográfico, que proporciona comunicaciones seguras por una red, comúnmente Internet.

Tensor

Termino usado en Machine learning, un tensor hace referencia a la matriz de datos de entrada que recibe un modelo neuronal.

7. Bibliografía

1. Buch, V. H., Ahmed, I., & Maruthappu, M. (2018).
   Artificial intelligence in medicine: current trends and future possibilities.
   Br J Gen Pract, 68(668), 143-144.
   http://bjgp.org/content/68/668/143.short

2. Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau & Sebastian Thrun (2017).
   Dermatologist-level classification of skin cancer with deep neural networks.
   Nature volume 542, pages 115–118
   https://www.nature.com/articles/nature21056

3. Chang, Hao (2017).
   Skin cancer reorganization and classification with deep neural network.
   https://arxiv.org/pdf/1703.00534.pdf

4. Kitware, Inc.
   ISIC.
   https://isic-archive.com/#images

5. DigitalOcean.
   https://www.digitalocean.com

6. Amazon Web Service
   https://aws.amazon.com

7. Namecheap
   https://www.namecheap.com/

8. Let's Encrypt
   https://letsencrypt.org/

9. PostgreSQL
   https://www.postgresql.org/

10. Docker
    https://docs.docker.com/engine/

11. Django Software Foundation.
    Django
    https://docs.djangoprojec.com/en/2.0/ref/

12. Keras.
    https://keras.io/

13. Google Brain.
    Tensorflow Image Recognition.
    https://www.tensorflow.org/tutorials/image_recognition

14. Centros para el Control y la Prevención de Enfermedades
    https://www.cdc.gov/spanish/cancer/skin/statistics/index.htm

15. Skin Cancer Foundation
    http://www.cancerdepiel.org/vida-saludable/sobre-el-bronceado/la-oms-publica-alerta-oficial

16 Git
https://git-scm.com/

- 17 GitHub Inc.
https://github.com/
- 18 Enfermedades cutáneas.
https://es.wikipedia.org/wiki/Categor%C3%ADa:Enfermedades_cut%C3%A1neas
- 19 François Chollet
Deep Learning with Python
- 20 Google IA Blog
Train your own image classifier with Inception in TensorFlow https://ai.googleblog.com/2016/03/train-your-own-image-classifier-with.html

8. Anexos

8.1 Alcance de las pruebas unitarias

Una prueba unitaria es una forma de comprobar el correcto funcionamiento de una unidad de **código**. Por ejemplo en diseño estructurado o en diseño funcional una función o un procedimiento, en diseño orientado a objetos una clase. Esto sirve para asegurar que cada unidad funcione correctamente y eficientemente por separado. Además de verificar que el código hace lo que tiene que hacer, verificamos que sea correcto el nombre, los nombres y tipos de los parámetros, el tipo de lo que se devuelve, que si el estado inicial es válido entonces el estado final es válido

A continuación se muestra el informe del alcance de las pruebas unitarias realizadas en todo el **servicio web**. Como ya se indicó anteriormente en apartados anteriores, para el desarrollo del servicio web se llevaran acabo buenas prácticas las cuales ayudan a evitar contratiempos y la creación de errores en el código.

```
Name                                                Stmts   Miss  Cover   Missing
---------------------------------------------------------------------------------
deeplk/__init__.py                                      0      0   100%
deeplk/settings.py                                     19      0   100%
deeplk/urls.py                                          6      0   100%
deeplk/wsgi.py                                          0      0   100%
manage.py                                               7      0   100%
reports/__init__.py                                     0      0   100%
reports/admin.py                                        9      0   100%
reports/forms.py                                       17      0   100%
reports/management/__init__.py                          0      0   100%
reports/management/commands/__init__.py                 0      0   100%
reports/migrations/0001_initial.py                      6      0   100%
reports/migrations/0002_auto_20180227_1253.py           4      0   100%
reports/migrations/0003_auto_20180304_1638.py           4      0   100%
reports/migrations/__init__.py                          0      0   100%
reports/models.py                                     179      0   100%
reports/tests.py                                      292      0   100%
reports/urls.py                                         3      0   100%
reports/views.py                                       65      0   100%
---------------------------------------------------------------------------------
TOTAL                                                 611      0   100%
```

8.3 Diagrámas tensorboard

En esta sección se mostrarán los diagramas de los modelos neuronales **personalizados**. Estos diagrámas se han obtenido a partir del servicio tensorboard que ofrece la librería tensorflow.

8.3.1 Personalizado 128 y 32

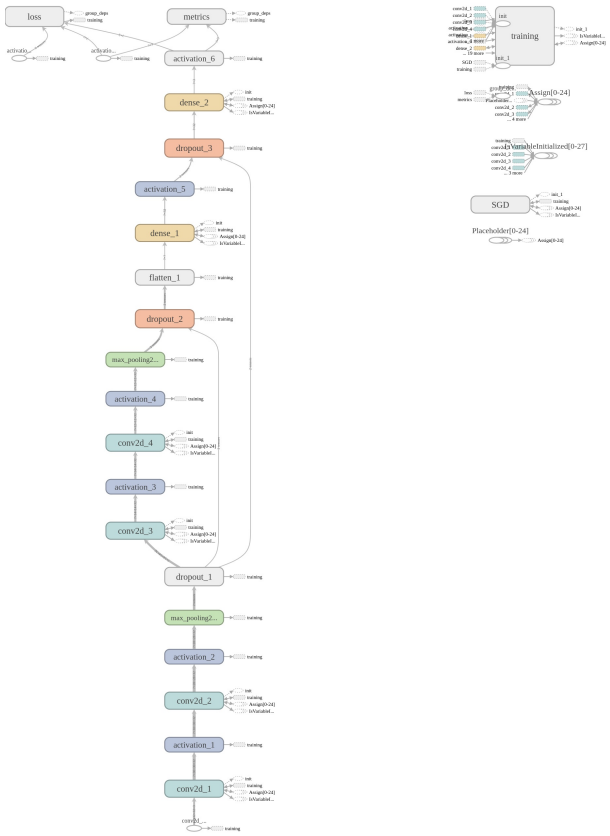

Figura 42. modelo neuronal convolucional personalizado, imagen generada con tensorboard

8.3.2 Personalizado 32 max

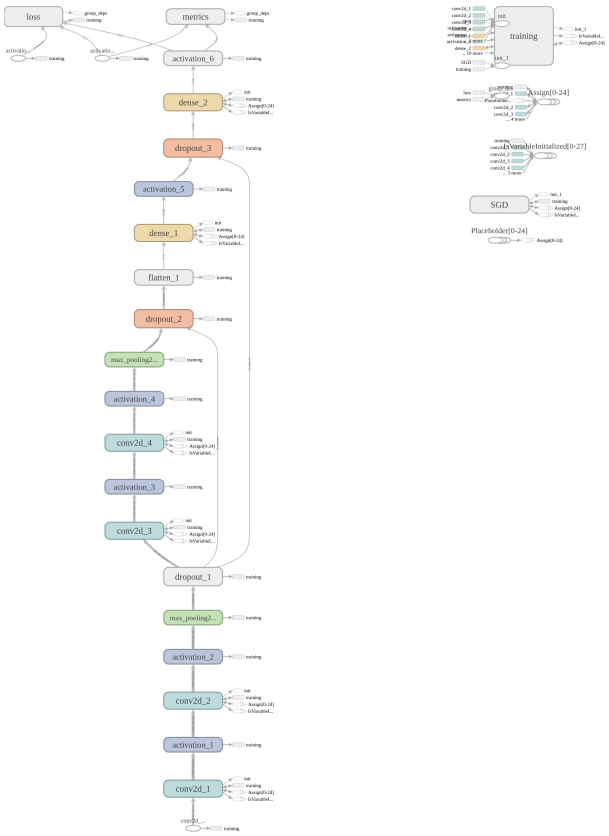

Figura 43. modelo neuronal convolucional personalizado, imagen generada con tensorboard