

# **Identificación de huellas de calzado a partir de imágenes con redes neuronales convolucionales**

**Laura Rivera Sánchez**

MU Ingeniería Computacional y Matemática  
Área de Inteligencia Artificial

**Nombre Tutor/a de TFM**

Elena Álvarez de la Campa Crespo

**Profesor/a responsable de la asignatura**

Carles Ventura Royo

**Fecha Entrega** 21/06/2023

**Firma del director autorizando la entrega final del TFM:**



Universitat  
Oberta  
de Catalunya



"Go in a suit if you commit a crime – there are no prints on dress shoes" Srihari (Wired [1])



Esta obra está sujeta a una licencia de  
Reconocimiento-NoComercial-SinObraDerivada [3.0](#)  
[España de Creative Commons](#)

## FICHA DEL TRABAJO FINAL

<b>Título del trabajo:</b>	<i>Identificación de huellas de calzado a partir de imágenes con redes neuronales convolucionales</i>
<b>Nombre del autor:</b>	<i>Laura Rivera Sánchez</i>
<b>Nombre del consultor/a:</b>	<i>Dra. Elena Álvarez de la Campa Crespo</i>
<b>Nombre del PRA:</b>	<i>Carles Ventura Royo</i>
<b>Fecha de entrega (mm/aaaa):</b>	<i>21/06/2023</i>
<b>Titulación o programa:</b>	Máster en ingeniería computacional y matemáticas (URV-UOC)
<b>Área del Trabajo Final:</b>	<i>Inteligencia Artificial</i>
<b>Idioma del trabajo:</b>	<i>Castellano</i>
Palabras clave	<i>Inteligencia artificial, reconocimiento de imágenes, huella</i>
<b>Resumen del Trabajo</b>	
Implementación de una red neuronal convolucional para la predicción de la marca en imágenes de huellas de calzado.	
El proyecto incluye un análisis de comparativo de los resultados de los diferentes experimentos donde se aplican diferentes transformaciones y/o aumentación a las imágenes, utilizar diferentes valores de <i>epoch</i> , diferentes muestras y alteraciones en las capas del modelo.	
Además, se incluyen pruebas de validación cruzada con imágenes de otra base de datos para comprobar el rendimiento del modelo.	
Por último, se ha desarrollado una red neuronal siamesa para la búsqueda de imágenes similares.	

**Abstract**

Implementation of artificial intelligence for the detection of shoe prints in order to identify whether the print exists in the sample or to know its brand / model to facilitate crime investigations.

# Índice

1.	Introducción .....	1
1.1.	Contexto y justificación del Trabajo .....	1
1.2.	Objetivos del Trabajo .....	4
1.3.	Impacto en sostenibilidad, ético-social y de diversidad .....	5
1.4.	Enfoque y método seguido .....	6
1.5.	Planificación del Trabajo .....	6
1.6.	Breve sumario de productos obtenidos.....	12
1.7.	Breve descripción de los otros capítulos de la memoria.....	12
2.	Estado del arte .....	13
2.1.	Huellas .....	13
2.2.	Realización de muestras.....	14
2.3.	Inteligencia artificial y redes neuronales .....	16
2.4.	Herramientas existentes .....	17
2.5.	Bases de datos .....	18
3.	Desarrollo del proyecto .....	21
3.1.	Tecnología .....	21
3.2.	Lectura y división del conjunto de datos .....	21
3.3.	<b>Objetivo 1: Predicción de la marca con la imagen de la huella de calzado .....</b>	25
3.3.1.	Formatear datos .....	25
3.3.2.	Análisis de los datos disponibles.....	26
3.3.3.	Extracción de características y clasificadores .....	27
3.3.4.	Clasificadores .....	29
3.3.5.	Redes neuronales convolucionales .....	30
3.3.5.1.	Modelo propuesto .....	32
3.3.5.2.	Optimizadores.....	34
3.3.5.3.	Preprocesado y generadores de las imágenes .....	34
3.3.5.4.	Aumentación .....	37
3.3.5.5.	Experimentos con la red propuesta .....	38
3.3.5.5.1.	¿Qué preprocesado funciona mejor? .....	39
3.3.5.5.2.	¿Más epoch resulta en mejor resultado? .....	43
3.3.5.5.3.	¿Cuál es el mínimo de muestras por marca aceptable para el modelo? .....	45
3.3.5.5.4.	¿Mayor tamaño de imagen obtiene mejores resultados? .....	48
3.3.5.6.	Variaciones en el modelo.....	50
3.3.5.7.	Autokeras .....	52
3.3.5.8.	ImageNet: Modelo preentrenado .....	53
3.3.6.	Pruebas de validación cruzada .....	56
3.4.	<b>Objetivo 2: Búsqueda de huellas similares .....</b>	60
3.4.1.	Análisis de los datos disponibles.....	60
3.4.2.	Solución propuesta .....	61
3.4.3.	Experimentos.....	62
4.	Conclusiones generales .....	64
5.	Comentarios .....	67
5.1.	Selección de los datos .....	67
5.2.	Ficheros necesarios para la ejecución.....	67

5.3.	Formato de las imágenes.....	68
5.4.	Red neuronal siamesa .....	68
5.5.	Google Colab .....	69
6.	Trabajo futuro .....	69
6.1.	Preprocesado de las imágenes.....	69
6.2.	Red neuronal de similitud.....	69
6.3.	One-shot Learning .....	69
6.4.	Pruebas con imágenes realizadas con el teléfono.....	69
6.5.	Entrenar el modelo con otro tipo de datos .....	70
	Bibliografía.....	71
	Anexos.....	74
a.1	Diagrama de Gantt definitivo .....	74
a.2.	Extracto del document: Shoeprint and tire track collection guide.....	75

# Lista de figuras

Figura 1: Procedimiento para guardar la huella dactilar de un individuo (Interpol)	1
Figura 2: Imagen de huellas de neumáticos [4]	2
Figura 3: Comparación de huella de calzado encontrada en escena de un crimen	3
Figura 4: Fechas claves actividades evaluables según plan docente [11]	6
Figura 5: Gantt del primer bloque correspondiente a la PEC1	8
Figura 6: Gantt del segundo bloque de preprocesamiento	8
Figura 7: Gantt del tercer bloque de creación del modelo	9
Figura 8: Gantt del cuarto, quinto y sexto bloque de modelos preentrenados, comparativas y conclusiones hasta la entrega de la PEC3	10
Figura 9: Gantt del sexto, séptimo y octavo bloque de conclusiones, líneas de trabajo futuras y defensa	10
Figura 10: Foto de Immo Wegmann en Unsplash de una huella dactilar	13
Figura 11: Extracto del manual de toma de fotografías de huellas	14
Figura 12: Ejemplo levantamiento de huella con yeso del tweet [18] de @srtaperito	15
Figura 13: Ejemplo de escáner 3D de la marca artec3d [19]	16
Figura 14: Programa SICAR-6	17
Figura 15: (a) imágenes de escena de crimen, (b) imágenes de referencia FID-300	19
Figura 16: Ejemplo de imagen y datos por imagen para 2D Footwear	20
Figura 17: Código Python para extraer las imágenes FID-300	21
Figura 18: Código Python para extraer las imágenes 2D Footwear	22
Figura 19: Código Python de las funciones <i>get_images</i> y <i>get_images_to_jpeg</i>	22
Figura 20: Código Python de la función <i>plot_image</i>	23
Figura 21: Código Python para la división de los datos	24
Figura 22: Código y resultado de la función <i>filesWithBrand</i>	25
Figura 23: Código Python de la función <i>filterMinSamples</i>	25
Figura 24: Gráfica de muestras por género (A) y marcas que más aparecen (B)	26
Figura 25: Código de las funciones <i>extractSIFT</i> y <i>getFiles</i>	28
Figura 26: Código y resultado de una imagen con sus puntos de interés extraídos con SIFT	28
Figura 27: Código del proceso que crea el diccionario de características	29
Figura 28: Capas del modelo de red neuronal propuesta	32
Figura 29: código que crea el modelo	32
Figura 30: Representación gráfica Dense vs Con2D	33
Figura 31: Código del generador	35
Figura 32: Fragmento de código donde se transforma la imagen	35
Figura 33: Código función que añade aumentación a las imágenes	36
Figura 34: <i>accuracy</i> , <i>loss</i> y matriz de confusión para GBC <i>epoch</i> =10	40
Figura 35: <i>accuracy</i> , <i>loss</i> y matriz de confusión para AGBC <i>epoch</i> =25	41
Figura 36: Tabla resumen de los experimentos según preprocesado	42
Figura 37: Tabla resumen de los experimentos con diferentes valores epoch	44
Figura 38: Matriz de confusión con marcas con mínimo 2 muestras con y sin aumentación	45

Figura 39: Matriz de confusión con marcas con mínimo 10 muestras sin aumentación.....	46
Figura 40: Tabla resumen de los experimentos con diferentes muestras mínimas .....	47
Figura 41: Tabla resumen de los experimentos con diferente tamaño de imagen .....	49
Figura 42: Tabla resumen de los experimentos con variaciones en las capas .	51
Figura 43: Modelo propuesto por autokeras.....	52
Figura 44: Tabla resumen de los experimentos con autokeras.....	53
Figura 45: Tabla resumen de los experimentos con VGG16 .....	55
Figura 46: Tabla resumen de las pruebas de validación cruzada .....	56
Figura 47: Ejemplo de imágenes de la marca Nike .....	56
Figura 48: Ejemplo de imágenes de la marca Adidas .....	57
Figura 49: Ejemplo de imágenes de la marca Converse.....	57
Figura 50: Código para extraer las imágenes de referencia que más aparecen en la tabla .....	60
Figura 51: Representación de la red siamesa (Source: learnopencv.com).....	61
Figura 52: Esquema carpetas para las imágenes del proyecto .....	67
Figura 53: Código de la función que lee y convierte los ficheros .tiff a .jpeg.....	68

# 1. Introducción

## 1.1. Contexto y justificación del Trabajo

En criminología se utilizan diferentes sistemas de reconocimiento para identificar sospechosos, los más conocidos son el reconocimiento de huellas dactilares o huellas de neumáticos de coche. El reconocimiento de huellas dactilares consiste en el reconocimiento de características en la huella que dejan los dedos sobre la superficie que permite identificar a individuos ya que no existen dos personas con la misma huella dactilar, razón por la que también es utilizado como herramienta de biometría. Por ejemplo, la *Interpol* dispone de su propia base de datos con más de 220.000 huellas dactilares que utiliza para identificar huellas encontradas en escenas del crimen o para identificar personas desaparecidas [2].



Figura 1: Procedimiento para guardar la huella dactilar de un individuo (Interpol)

El reconocimiento de huellas de neumáticos se utiliza para identificar la medida y marca de los neumáticos partiendo de que cada marca tiene su propio relieve, además pueden extraer información con las marcas de deslizamiento y de dirección para determinar comportamientos que pueden determinar los detalles de un suceso [3].

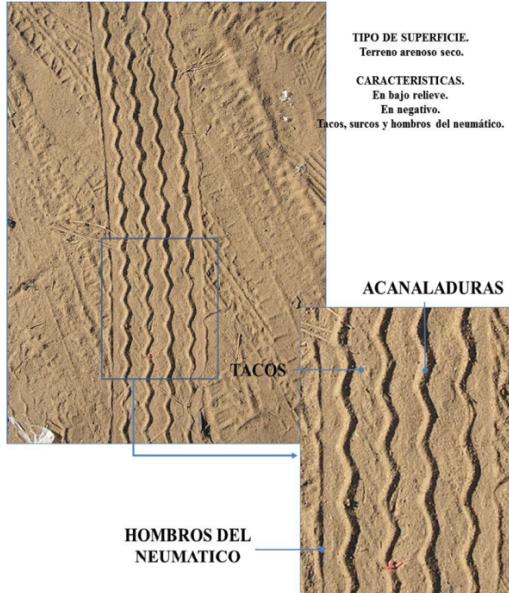


Figura 2: Imagen de huellas de neumáticos [4]

Pero también existen mecanismos para la identificación de individuos por sus huellas plantares, las huellas de calzado o el análisis de la marcha [5]. En general, de este trabajo se encarga la Podología Forense, que se define como la aplicación del conocimiento y experiencia de la Podología/Podiatría en la resolución de casos medicolegales, para localizar e identificar un individuo en la escena de un crimen [6] ya que dispone de conocimientos sobre biomecánica suficientes para identificar características físicas viendo su huella o forma de caminar. Respecto a las huellas de calzado o descalzo, normalmente la policía tiene su propia base de datos de huellas encontradas en las escenas para poder compararlas con muestras extraídas de los sospechosos.

Este trabajo se va a centrar en la detección de huellas de calzado para resolver el problema en el **sector de la criminología**. En concreto, en crear un modelo de redes neuronales que identifique primero una huella en una fotografía, determine si una huella se encuentra en la base datos y, por último, determinar la marca y modelo del calzado. De este sector, existen algunas herramientas que utilizan los departamentos de policía, como SICAR 6 [7] o PRIDE [8] que identifican la marca y modelo de la huella facilitada, aparte de disponer de su propia base de datos con el historial de huellas en escenas del crimen.



Figura 3: Comparación de huella de calzado encontrada en escena de un crimen

Además, este sistema de reconocimiento podría cubrir necesidades de otros sectores, además del de criminología nombrado anteriormente. Por ejemplo, en el **sector médico** ya que una investigación interesante sería complementar esas huellas con datos médicos para determinar factores relacionales con la fisionomía y postura del individuo, por ejemplo, determinar peso o patologías y peculiaridades al andar (si es supinador o pronador, por ejemplo). En esta línea, existen investigaciones para determinar altura y sexo en huellas de pies descalzos, como por ejemplo en el reciente artículo de Md Asadujjaman, Md Harun Or Rashid, Md Sohel Rana & Md Mosharraf Hossain que estudian una estimación de la altura en huellas de adultos [9].

Otro sector podría ser el de **sistemas de seguridad**, desde hace años existen líneas de investigación (Kapil Kumar Kagwanshi, Sipi Dubey) [10] en las que se aplica la lectura de la huella plantar como herramienta de biometría para la identificación única de las personas, como alternativa a la huella dactilar o iris del ojo.

Un último sector, y pensando en una manera de monetizar el proyecto, sería el de **marketing** ya que se podría aplicar para nuevas fórmulas de fidelización de clientes en marcas de calzado, creando un nuevo producto que mediante el uso de una alfombra determine la marca y modelo que utilizan los clientes.

## 1.2. Objetivos del Trabajo

El objetivo de este trabajo no es exclusivamente dar solución general al problema descrito, ya que existen herramientas en el mercado con grandes bases de datos que lo resuelven. El objetivo principal es estudiar si los modelos de redes neuronales convolucionales (*CNN*) son capaces de dar resultados satisfactorios con un conjunto de datos limitado y reconocer la marca de huellas de calzado presentes en una fotografía.

Sin entrar en especificaciones técnicas, este objetivo se divide en los siguientes hitos generales:

- Búsqueda de una base de datos de huellas de zapatos.
- Preprocesado de las imágenes para determinar el mejor método para extraer sus características.
- Realizar aumentación de datos: generar nuevas imágenes a partir de las disponibles para nutrir la red neuronal.
- Crear un modelo con redes neuronales capaz de **identificar la marca de la huella de muestra**.
- Determinar de forma eficaz si la huella está dentro de la base de datos (simular el caso en el que la policía comprueba si la huella extraída del zapato de un sospecho se parece a alguna de las encontradas en la escena del crimen).

Se va a considerar un resultado eficaz si se obtiene una eficacia de como mínimo el 85%. Y se considera aceptable un 65% de eficacia.

### 1.3. Impacto en sostenibilidad, ético-social y de diversidad

Respecto a los aspectos de la dimensión de **sostenibilidad medioambiental**, la creación del modelo en los servidores de *Google Colab* se podría considerar que tiene un impacto negativo ya que su creación implica un uso intensivo de esos servidores que están conectados a la corriente, pero no existe manera de mitigarlo o saber si se utilizan energías renovables. Por otro lado, podría tener un impacto positivo por el hecho de que, si se crea una aplicación móvil para capturar las imágenes y el modelo es capaz de reconocer las huellas con la calidad del dispositivo, no haría falta el uso de cámaras específicas que después suponen un residuo difícil de reciclar. Ambos aspectos están relacionados con el **ODS 12- Responsible consumption and production.**

En la **dimensión ético-social**, el modelo en sí, en un notebook de *Jupyter* no tendría ningún impacto en la sociedad, ni positivo ni negativo, ya que no llegaría a comercializarse. En el caso de ser un producto apto para que la policía lo utilice, tendría un impacto positivo ya que ayudaría a la resolución de casos de investigación y reducir los crímenes. De la misma manera, si se usara de manera malintencionada, se le podría dar un mal uso para acosar/perseguir personas o a alguna comunidad. Estos aspectos corresponden al **ODS 16- Peace, justice and strong institutions.**

Por último, en la **dimensión sobre diversidad, género y derechos humanos** no tiene ningún impacto ni positivo ni negativo. Quizás el impacto negativo, ya nombrado en la dimensión anterior, que con un mal uso se pudiera perseguir algunas comunidades (etnias) vulnerando el **ODS 10- Reduced inequalities.**

#### 1.4. Enfoque y método seguido

Aunque existen herramientas que resuelven el problema, para este trabajo he decidido crear un **modelo desde cero** para experimentar todo el proceso de creación de una red neuronal: preprocesado, parametrización, evaluación y conclusiones.

Se utilizará el lenguaje de programación **Python**, en un cuaderno Jupyter utilizándolos en un servidor **Jupyter** local.

En cuanto a la metodología de trabajo, se ha **dividido los objetivos en 8 bloques** (mencionados en el siguiente apartado) para ir trabajando cada una de las fases del proyecto. Aunque se prevé que las tareas sean secuenciales, las tareas de documentación y redacción se realizarán por cada bloque, para adelantar la redacción de la memoria durante el proceso.

#### 1.5. Planificación del Trabajo

Para la planificación de las tareas, se va a tener en cuenta las fechas de entrega del plan docente de la asignatura:

Nombre	Inicio / Enunciado	Entrega	Solución	Calificación
PEC1 - Definición y plan de trabajo	16/11/2022	16/12/2022	-	23/12/2022
PEC2 - Desarrollo del trabajo - Fase 1	17/12/2022	10/03/2023	-	24/03/2023
PEC3 - Desarrollo del trabajo - Fase 2	11/03/2023	05/05/2023	-	14/05/2023
PEC4 - Redacción de la memoria y presentación	06/05/2023	07/06/2023	-	27/06/2023
PEC5b - Defensa pública	12/06/2023	22/06/2023	-	27/06/2023

Figura 4: Fechas claves actividades evaluables según plan docente [11].

Antes de crear la planificación, se ha dividido el proyecto en diferentes objetivos principales:

#### 1.5.1. Tareas principales

- Preprocesamiento: este grupo de tareas incluye las correspondientes a escoger la base de datos, estudiar y documentar los métodos de extracción de características de las imágenes, aumentación y división del conjunto de datos en entrenamiento, test y validación.
- Red neuronal predicción de marca: incluye las tareas correspondientes a estudiar y documentar los modelos secuenciales conocidos, experimentar con diferentes parámetros para seleccionar el que utilizará el proyecto.
- Uso de modelos preentrenados: tareas para añadir modelos preentrenados al modelo para comparar los datos.
- Comparativa: tareas de redacción y creación de gráficas para la comparativa de resultados.
- Red siamesa para búsqueda de huellas similares: Incluye el desarrollo de una red neuronal siamesa para dará una imagen obtener similares, sin tener en cuenta la marca.
- Conclusiones: aunque durante todas las tareas anteriores se irá redactando los procesos y conclusiones, esta tarea incluye la redacción de conclusiones globales y análisis de hipótesis.
- Líneas de trabajo futuras: Redacción de posibles líneas de trabajo futuras.

#### 1.5.2. Tareas adicionales

En el apartado de líneas de trabajo futuras, se pretende guardar un espacio de tiempo para:

- Experimentar con otra bases de datos para realizar pruebas de validación cruzada.
- Analizar la complejidad de alojar el código *Python* en un servidor, en lugar de en un cuaderno *Jupyter*, y ver que supondría crear una aplicación móvil que mande la imagen a ese servidor para recibir la respuesta sobre la marca y modelo de la huella enviada.

### 1.5.3. Planificación GANTT

A continuación, se detalla cada uno de los bloques de tareas que se tendrán en cuenta y su diagrama Gantt realizado con TeamGantt [12]. El diagrama completo se encuentra en el anexo [[Anexo a.1](#)].

#### Definición y plan de trabajo (Del 16/11/2022 al 23/12/2022)

Incluye todas las tareas correspondientes a la primera entrega sobre investigación y definición de los objetivos del trabajo, además de la redacción del segundo capítulo de estado del arte.

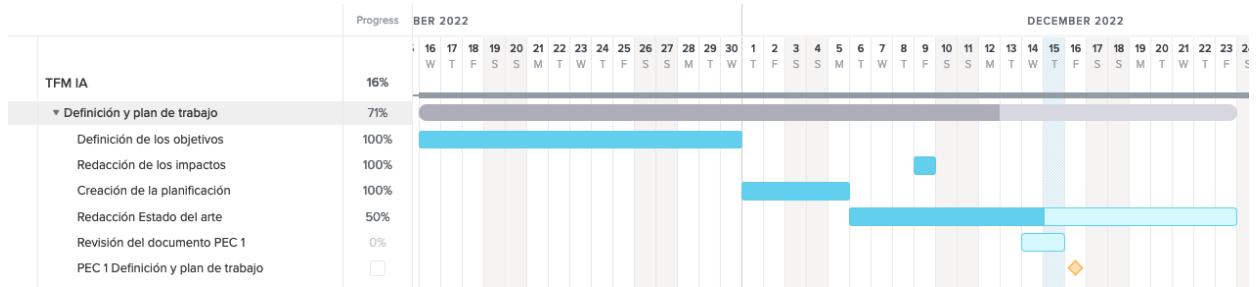


Figura 5: Gantt del primer bloque correspondiente a la PEC1.

#### Preprocesamiento (del 01/01/2023 al 25/01/2023)

Incluye tareas como la elección de la base de datos (iniciada antes de la primera entrega, ya que he considerado que era importante para la definición de objetivos) y todas las tareas correspondientes al preprocesamiento de las imágenes: extracción de características, aumentación y división del conjunto de datos en subconjuntos de entrenamiento, test y validación.

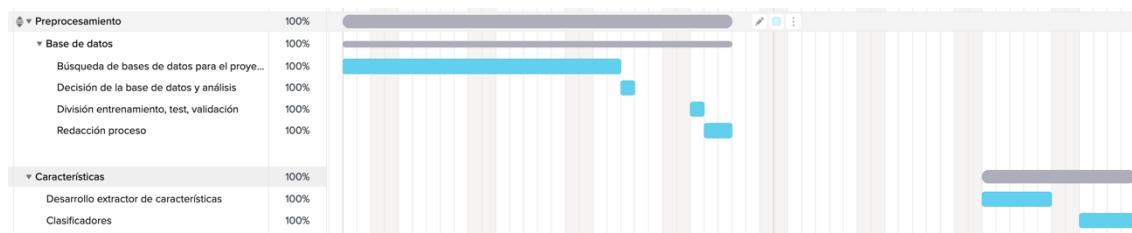


Figura 6: Gantt del segundo bloque de preprocesamiento.

### **Modelos secuenciales (del 26/01/2023 al 09/03/2023)**

Incluye las tareas de estudiar los modelos secuenciales para después programar el modelo que posteriormente se entrenara con los datos del bloque de preprocesamiento con los parámetros que obtuvieron mejores resultados. Durante este proceso también se incluye la redacción del proceso de la memoria y el análisis de resultados.

El final de este bloque coincide con la entrega de la PEC2 de desarrollo – Fase 1, por lo que esta entrega va a incluir los bloques de preprocesado y creación del modelo.

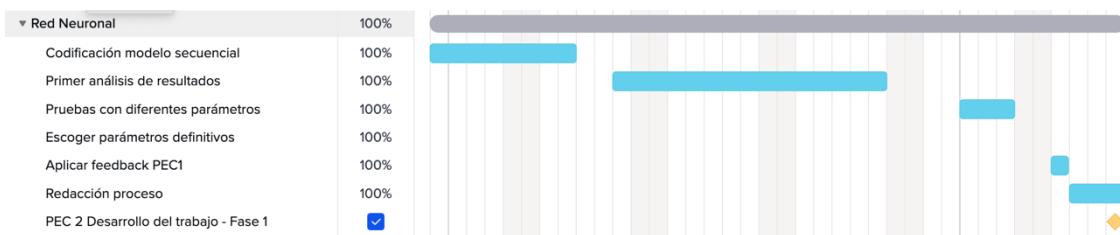


Figura 7: Gantt del tercer bloque de creación del modelo.

### **Modelos preentrenados (del 13/03/2023 al 31/03/2023)**

Incluye las tareas de incorporar datos preentrenados al modelo y analizar los resultados. Añade la redacción del proceso y conclusiones.

### **Red siamesa de similitud (27/03/2023 al 19/04/2023)**

Incluye las tareas de investigar y realizar una primera versión de un red neuronal siamesa para la búsqueda de imágenes similares.

### **Comparativa (del 23/04/2023 al 04/05/2023)**

Incluye las tareas de documentar la comparativa de resultados de los dos bloques anteriores: redacción y creación de gráficas.

### **Conclusiones (del 08/05/2023 al 28/05/2023)**

Incluye las tareas de redacción de las conclusiones generales del trabajo, además de realizar pruebas con fotografías de fuera del conjunto de datos para testear el modelo. En estas fechas coincide la entrega de la PEC3, donde se entregará el material realizado hasta el bloque de comparativa y una primera versión de las conclusiones.

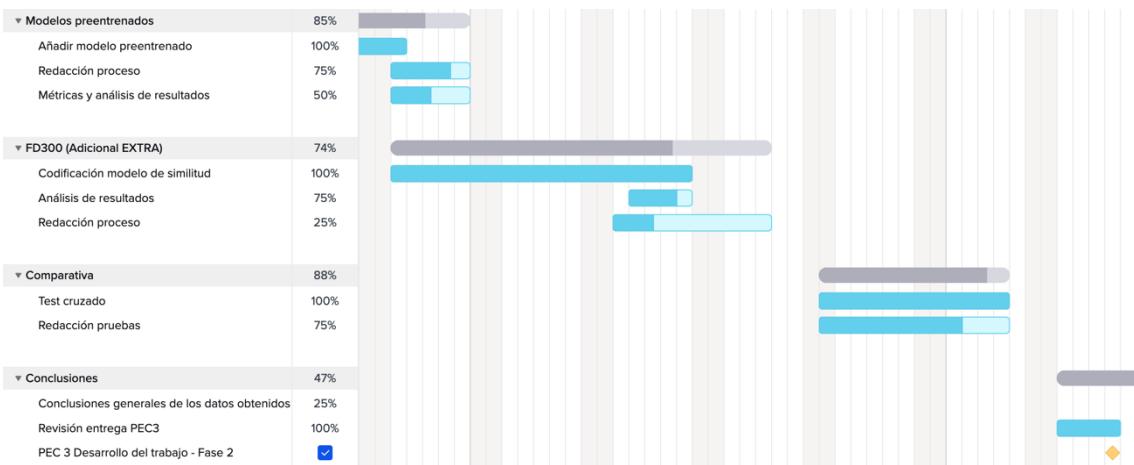


Figura 8: Gantt del cuarto, quinto y sexto bloque de modelos preentrenados, comparativas y conclusiones hasta la entrega de la PEC3

### Líneas de trabajo futuras (del 25/05/2023 al 31/05/2023)

Incluye tareas de redacción de otras líneas de trabajo que se podrían realizar. Además, de forma adicional/opcional me gustaría probar el modelo con otros datos, si se reciben y también investigar qué pasos debería realizar para tener el modelo en un servidor.

### Defensa (del 01/06/2023 al 21/06/2023)

Incluye las tareas de redacción final de la memoria, que se ha ido realizando en los bloques anteriores, grabación y edición de la presentación para la defensa y reservar los días de la defensa. Este bloque coincide con la última entrega de la PEC4 de la memoria y con la PEC 5 de defensa pública.

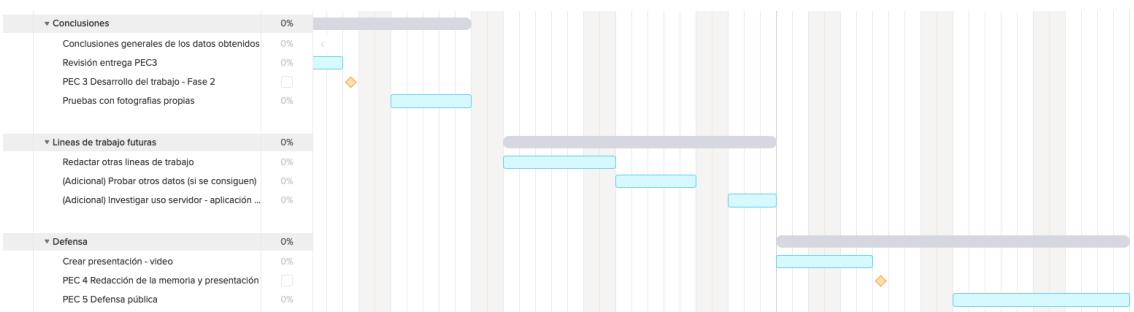


Figura 9: Gantt del sexto, séptimo y octavo bloque de conclusiones, líneas de trabajo futuras y defensa

#### 1.5.4. Análisis de riesgos

Como todo proyecto de cierta envergadura existen algunos riesgos a tener en cuenta para intentar mitigarlos durante el proceso. A continuación, se enumeran los analizados durante la definición y planificación:

- **Base de datos:** Se ha adelantado la tarea de búsqueda de la base de datos a utilizar, ya que existen pocas abiertas para su uso. Este añade el riesgo de disponer de pocos datos para que el modelo entrene, por esa razón se ha añadido la tarea de aumentación de los datos.
- **Retrasos en la planificación:** La planificación se ha realizado priorizando la finalización del producto central: el modelo de predicción de marca y análisis de resultados. Por esa razón, las dos tareas adicionales de ampliación del proyecto (probar otros datos y modelo de similitud) se han dejado para el final, así en caso de que las tareas principales se atrasaran tener margen para finalizar el objetivo real del trabajo.
- **Tiempos de ejecución en Google Colab:** En la experiencia en entregas de otras asignaturas, algunos procesos pueden tardar un tiempo considerable en las colas de *Google Colab*, por lo que es algo a tener en cuenta, además de limitaciones en la memoria (y por esa razón finalmente se ha utilizado *Jupyter*).

## 1.6. Breve sumario de productos obtenidos

El producto final del trabajo son cinco cuadernos Jupyter en los que se encuentran diferentes implementaciones de las funcionalidades objetivo desglosado en las diferentes fases.

Además, un PDF con la memoria del proceso y análisis de los resultados obtenidos, y una presentación, tanto en PDF como en video explicativo para la defensa.

## 1.7. Breve descripción de los otros capítulos de la memoria

El resto de los capítulos de la memoria corresponden, entre otros, a los objetivos principales del trabajo:

- **Capítulo 2 Estado del arte:** Aquí se explican conceptos sobre cómo se usan las huellas en criminología, su extracción y herramientas existentes. Además, se explica las bases de datos encontradas para poder utilizarlas en el trabajo.
- **Capítulo 3 Desarrollo del proyecto**
  - **Capítulo 3.1 Preprocesamiento:** Determinar el mejor método para extraer características, aumentación y división del conjunto de datos en entrenamiento, test y validación.
  - **Capítulo 3.2 Modelo de predicción de la marca:** Experimentar con diferentes parámetros, mostrando resultados y especificar el modelo seleccionado.
    - 3.2.1 Clasificadores.
    - 3.2.3 Red neuronal.
    - 3.2.4 Experimentos con diferentes configuraciones.
    - 3.2.5 Modelo preentrenado.
    - 3.2.6 Resumen de resultados.
  - **Capítulo 3.3 Pruebas de validación cruzada:** Uso de otra base de datos con imágenes de huellas para experimentar en el modelo anterior.
  - **Capítulo 3.4 Red siamesa para la búsqueda de huellas similares:** Crear una red siamesa para obtener huellas similares entre ellas.
- **Capítulo 4 Conclusiones generales.**
- **Capítulo 7 Comentarios.**
- **Capítulo 8 Líneas de trabajo futuras.**

## 2. Estado del arte

Este apartado se destina a explicar conceptos sobre las bases de datos encontradas sobre huellas plantares, como se realizan las muestras en las investigaciones de criminología y esas herramientas existentes para el problema planteado.

### 2.1. Huellas

Según la RAE [13], huella se define como la señal que deja el pie del hombre o del animal en la tierra por donde pasa, y rastro, seña, vestigio que deja alguien o algo.

En las escenas de crímenes éstas son muy importantes ya que pueden ayudar a identificar a las personas implicadas.

Las huellas más conocidas por el público general son las **huellas dactilares** o dactiloscopia, y forma parte de la biometría, una ciencia basada en el reconocimiento de una característica física o biológica para identificar a una persona. No existen dos personas con la misma huella dactilar, por lo que es muy utilizada en criminología para identificar sospechosos o víctimas de catástrofes. Por ejemplo, la *Interpol* dispone del *Sistema Automático de Identificación Dactilar (SAID)* con más de 200.000 huellas almacenadas y en 2019 se realizaron más de 1600 identificaciones gracias a estos datos [2].



Figura 10: Foto de Immo Wegmann en [Unsplash](#) de una huella dactilar

Pero existen otros tipos de huellas que pueden ayudar a resolver un crimen. Según el Blog de CFEC [14], existen diferentes tipos de huellas, como las huellas latentes, que son aquellas que son invisibles, pero se revelan con sustancias químicas, las de arrastre, producidas por el movimiento o traslado de una persona o cosa, o las de frenado, producidas por los neumáticos al frenar.

Por último, otro tipo de huella importante para la resolución de algunos crímenes, y en las que se centra el trabajo, son las huellas de calzado, según Michael Nirenberg [15], existen tres factores que ayudan a resolver algunas investigaciones: huellas plantares, calzado y el análisis de la marcha.

## 2.2. Realización de muestras

Como el proyecto se centra en las huellas de calzado sobre la superficie, se ha buscado documentación de cómo se deben realizar las fotografías de muestra de este tipo de evidencia, para ello he hecho uso del manual facilitado por el Departamento de seguridad pública de Minnesota [16] donde, entre las instrucciones para la extracción de muestras toxicológicas o biológicas, también facilita un manual para realizar la fotografía de marcas de calzado y ruedas.

### 2. Take the photograph directly over the impression.



**Figure 1:** A photograph taken from this position will be distorted.



**Figure 2:** This is the position a photograph should be taken at. Using a tripod is best because it reduces shake and ensures you are directly over the impression. You can check the position of the camera from the side or using a level.

Figura 11: Extracto del manual de toma de fotografías de huellas.

En resumen, los **pasos** a seguir son:

- Realizar la fotografía directamente por encima de la huella.
- A ser posible, utilizar una regla plana en forma de “L” para tomar las dimensiones de la huella.
- Capturar la totalidad de la huella con dicha regla.
- Utilizar iluminación desde diferentes ángulos.
- Realizar varias fotografías.

De manera que, a excepción de la disposición de esa regla en forma de “L”, los pasos sugeridos son bastante asequibles para cualquier persona. Se puede encontrar un extracto de las instrucciones para las huellas de zapatos en el Anexo a.2. Extracto del document: Shoeprint and tire track collection guide.

Existen otras maneras más sofisticadas de extraer muestras de huellas de calzado, como por ejemplo el **levantamiento de huella con yeso**, documentado en un video [17] por el Instituto Universitario La Puebla, en el que básicamente se aplica el yeso encima de la huella en el terreno, que una vez seco ofrece el negativo de la huella.



Figura 12: Ejemplo levantamiento de huella con yeso del tweet [18] de @srtaperito.

Otra metodología es el uso de un **escáner 3d** de mano ligeros para capturar digitalmente la topografía completa de huellas específicas, estos datos se exportan a un programa para poder ser visualizadas.



Figura 13: Ejemplo de escáner 3D de la marca artec3d [19].

### 2.3. Inteligencia artificial y redes neuronales

La inteligencia artificial (IA) es una disciplina académica relacionada con la teoría de la computación cuyo objetivo es emular algunas de las facultades intelectuales humanas [20]. En resumen, lo que pretende es aprender de los datos disponibles para, posteriormente, predecir un resultado o clasificación.

A grandes rasgos, los modelos de IA se pueden dividir en dos grandes grupos: agrupamiento y clasificación. El primero, crea agrupaciones dependiendo de las características de los datos (sin definir previamente esos grupos), y el segundo, clasifica según lo aprendido de unos datos etiquetados.

Por último, el Deep Learning trata de simular el funcionamiento de las neuronas del cerebro, tomando así su nombre para crear una red neuronal en la que cada capa realiza una transformación para llegar a una solución. La peculiaridad de las redes neuronales es que, durante la fase de entrenamiento, aprende de forma automática para seleccionar los valores de variables y constantes que después se utilizaran para la clasificación de nuevas instancias de datos.

## 2.4. Herramientas existentes

Actualmente existen diversas herramientas en el mercado que disponen de una amplia base de datos de huellas de zapatos y pueden ayudar a la policía a obtener más información sobre una huella encontrada.

La más conocida es **SICAR 6** [21] de Foster & Freeman, que hace uso de la base de datos SoleMate, una de las más completas con más de 42.000 muestras. Dada una imagen permite determinar la marca y modelo tanto de huellas de zapatos como de huellas de neumáticos.

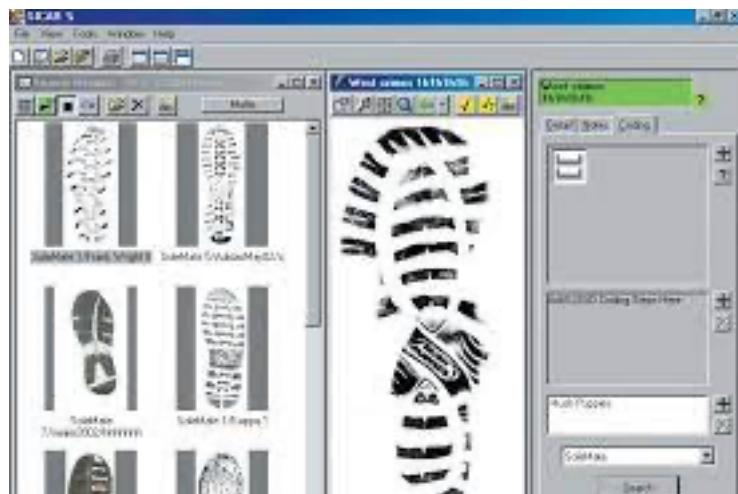


Figura 14: Programa SICAR-6

Además, existen otras herramientas **DigTrace** [22] es una solución de software integrada descargable para la captura y análisis de datos 3D ya sea en un contexto forense (calzado) o huellas de vertebrados. Otra herramienta similar es **PRIDE** [23], de **Hobbit Imaging Solutions** de Holanda, que es un comparador automático de huellas de zapatos.

## 2.5. Bases de datos

Para el proyecto se necesita de una base de datos con imágenes de huellas ya clasificadas. Existen pocas opciones disponibles para su uso abierto, ya que algunas de las herramientas y/o empresas nombradas anteriormente tienen los derechos.

Aunque se ha intentado contactar con algunas de las empresas para el uso académico de sus imágenes no se ha recibido una respuesta satisfactoria. En concreto, se ha contactado con:

- UK government [24]: disponen de un formulario para pedir acceso a su base de datos de huellas, pero me encuentro a la espera de respuesta.
- CTA.ai: Disponen de una base de datos de todas las vistas de zapatos comerciales en kaggle [25], pero la competición no es accesible.
- Foster & Freeman: Empresa propietaria de SoleMate, pero ya no continúan con el proyecto.

Finalmente, después de leer diversos artículos de trabajos en la misma línea, se ha decidido utilizar dos bases de datos para los dos objetivos principales del proyecto: una para la búsqueda de huellas similares y otra para determinar la marca de la huella.

Para el primero objetivo, se utiliza la base de datos **FID-300** [26] de Adam Kortylewski y utilizada en diversas investigaciones sobre criminología, como por ejemplo la del mismo creador con Thomas Albrecht y Thomas Vetter [27], donde utilizan diferentes métodos para agrupar diferentes patrones en una huella. También la utilizan Yanjun Wu, Xianling Dong, Guochao Shi, Xiaolei Zhang y Congzhe Chen [28] en su artículo donde la usan para mostrar un resumen de los análisis existentes.

Para el segundo objetivo, se utiliza la base de datos abierta para su uso en investigación de Soyoung Park y Alicia Carriquiry [29] de la Universidad de Iowa con **150 muestras** de calzado de personas con los datos de género, marca, modelo y talla [30].

### 1.1.1 FID-300

Esta base de datos contiene **300 muestras** de huellas de calzado obtenidas en escenas de crímenes por forenses mediante el escaneo de la muestra en gelatina o realizando la fotografía directamente. Además, incluye **1175 impresiones de referencia** realizadas mediante la aplicación de gelatina en suelas de diferentes zapatos y escaneando el resultado. Además, se dispone de una tabla donde se ha etiquetado cada imagen de muestra (de la escena del crimen) con su correspondiente imagen de referencia.

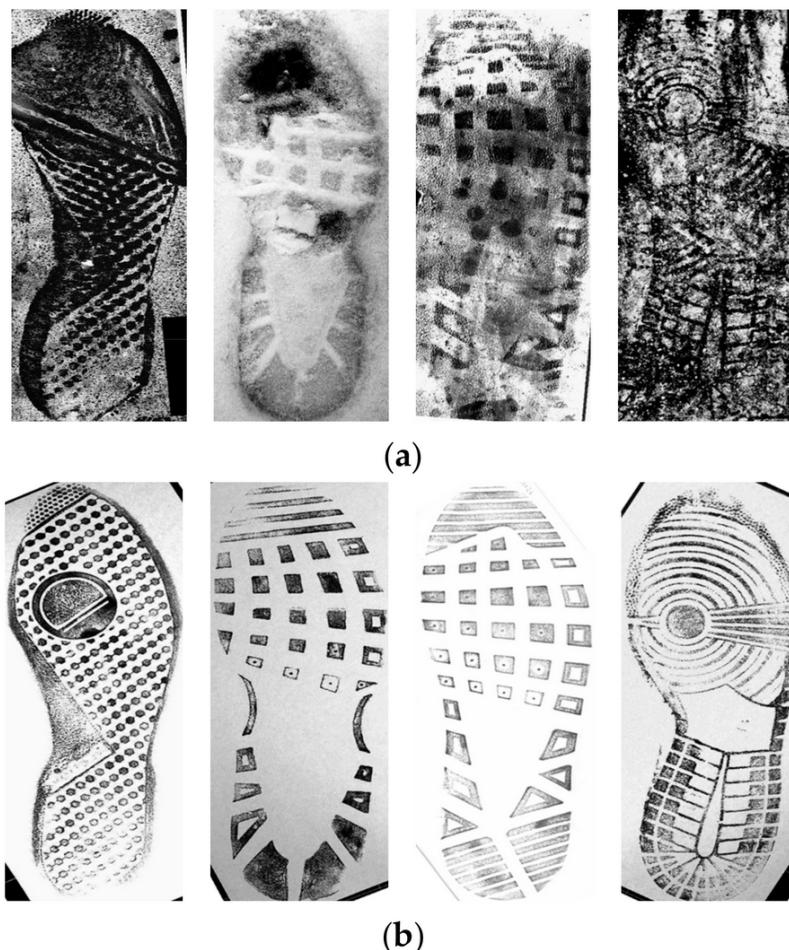


Figura 15: (a) imágenes de escena de crimen, (b) imágenes de referencia FID-300

### 1.1.2 2D Footwear outsole impressions

Esta segunda base de datos está formada por 1.500 impresiones de 150 pares de zapatos diferentes. La información sobre la base de datos contiene ID del par de zapatos, género, marca, modelo y talla.

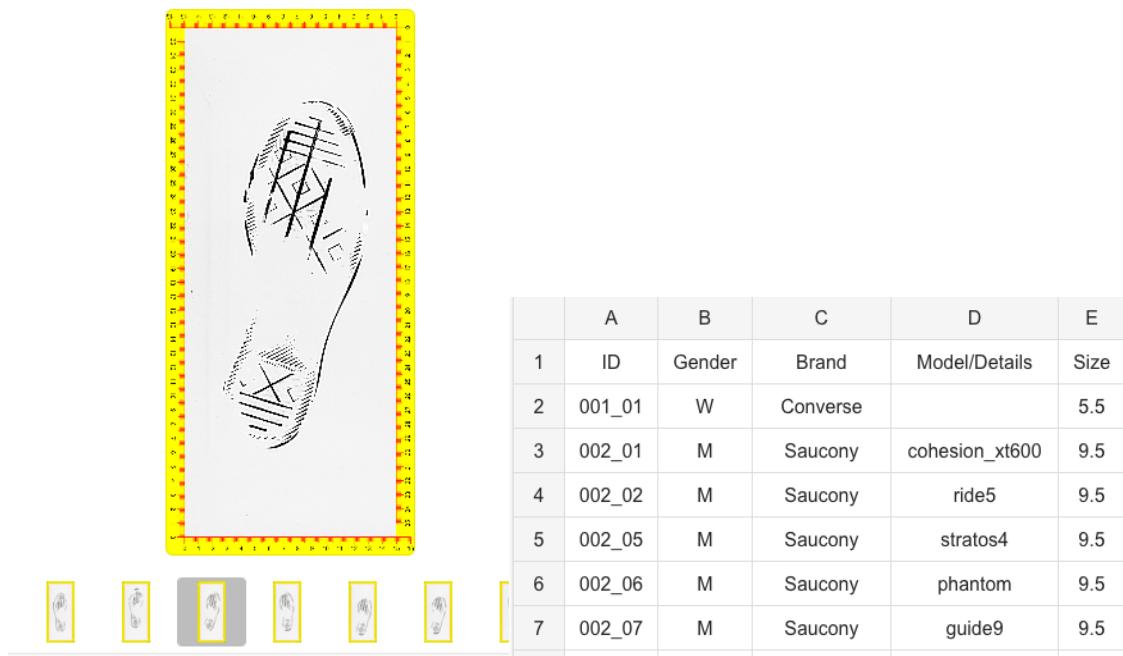


Figura 16: Ejemplo de imagen y datos por imagen para 2D Footwear

# 3. Desarrollo del proyecto

## 3.1. Tecnología

Inicialmente el desarrollo del proyecto se empezó con un cuaderno en *Google Colab*. *Google Colab*, que es una herramienta en la nube que permite crear cuadernos *Jupyter* en un entorno preparado con las principales librerías, además facilita la colaboración con otras personas y permite, de forma gratuita, el uso de *GPU* [31]. Pero, finalmente, se ha utilizado *Jupyter Notebook* por las limitaciones en la *RAM* de *Google Colab*.

Cada uno de los apartados siguientes se han programado en diferentes bloques o cuadernos con lenguaje *Python*.

## 3.2. Lectura y división del conjunto de datos

Este apartado incluye todo lo referente a la lectura de las bases de datos y división de estos en subconjuntos de *train*, *test* y validación, funciones comunes para ambos modelos.

### Lectura ficheros

En ambos casos las imágenes vienen comprimidas en uno o varios ficheros *ZIP* y una tabla en formato *csv* con su información.

Estos ficheros se han añadido en sus respectivas carpetas para facilitar su lectura al ejecutar el programa. Los datos de los dos conjuntos se han guardado en dos carpetas *2dFootwear* y *FID300* respectivamente, para mejor organización.

A continuación, se muestra el código utilizado para leer los datos de las dos bases de datos, en el primer caso (*FID300*) las imágenes se encuentran en un único fichero comprimido, en el segundo las imágenes están divididas en 5 partes.

```
with ZipFile('data/FID300/images.zip', 'r') as zipObj:  
    zipObj.extractall('fid300')  
  
#Lectura de la tabla de resultados  
df_fid300 = pd.read_csv('data/FID300/label_table.csv', delimiter=',')
```

Figura 17: Código Python para extraer las imágenes FID-300

```

def unzipImages(folder='images'):
    with ZipFile('data/2dFootwear/Part1.zip', 'r') as zipObj:
        zipObj.extractall(folder)

    with ZipFile('data/2dFootwear/Part2.zip', 'r') as zipObj:
        zipObj.extractall(folder)
    with ZipFile('data/2dFootwear/Part3.zip', 'r') as zipObj:
        zipObj.extractall(folder)

    with ZipFile('data/2dFootwear/Part4.zip', 'r') as zipObj:
        zipObj.extractall(folder)

    with ZipFile('data/2dFootwear/Part5.zip', 'r') as zipObj:
        zipObj.extractall(folder)

if not os.path.isdir("images_full"):
    unzipImages("images_full")

```

Figura 18: Código Python para extraer las imágenes 2D Footwear

Además, se ha creado una función *get\_images* y *get\_images\_to\_jpeg* para obtener un array con los nombres de fichero dentro de una carpeta y, la segunda para, además, convertir las imágenes *.tiff* a *.jpeg* eliminando el marco de medición para los ficheros de la base de datos 2d Footwear.

```

def get_images_to_jpeg(imgPath):
    dir_list = os.listdir("./"+imgPath)
    result = []
    for f in dir_list:

        im = Image.open("./"+imgPath+"/"+f)
        im2=im.resize((400,912))
        im3 = im2.crop((40,40,320,872)) #Quitar marco medidor
        im3.save("./"+imgPath+"/"+f[0:-4]+'.jpeg')

        result.append(f[0:-4]+'.jpeg')
        os.remove("./"+imgPath+"/"+f)

    print('Nº files:',len(result))
    return result

def get_images(imgPath):
    dir_list = os.listdir("./"+imgPath)
    result = []
    for f in dir_list:
        result.append(f)

    print('Nº files:',len(result))
    return result

```

Figura 19: Código Python de las funciones *get\_images* y *get\_images\_to\_jpeg*

Además, se ha implementado `plot_image` para visualizar un listado de imágenes:

```
def plot_image(imgPath, fileNames):
    for i in range(len(fileNames)):
        filename = fileNames[i]
        img = io.imread(imgPath+filename)

        plt.figure()
        plt.title(str(img.shape)+" , "+str(img.dtype))
        plt.imshow(img)
        print(fileNames)
        plt.show()
```

Figura 20: Código Python de la función `plot_image`

### División del conjunto de datos

Como se dispone de dos bases de datos de imágenes de huellas de calzado, se ha decidido realizar diferentes divisiones de los datos para realizar diversas pruebas:

- **Uso individual:** Ejecutar el modelo en cada base de datos por separado, seleccionando aleatoriamente 70% de las instancias para entrenamiento, 20% para testear y 10% para validación.
- **Uso combinado:** Una vez se tenga el modelo de predicción de marca entrenado, se realizarán pruebas con la otra base de datos. Es decir, , con el modelo entrenado con el 100% de las imágenes de 2d Footwear se harán pruebas con las imágenes de FID-300, tanto con las de referencia como las recogidas en diferentes escenarios.

En total, se prevén dos pruebas de forma individual y una combinada.

A continuación, el código responsable para la división de los datos, al disponer de pocos datos en el caso de la base de datos 2d Footwear, se ha creado, además la función `checkBalancedSample` para asegurar que las marcas que aparecen en los datos de `train`, también aparezcan en `test` y `val`.

```

def checkBalancedSample(train, test, val):
    checkTest = False
    checkVal = False

    #Comprobar si existen en train
    test_in = test.y.isin(train.y).astype(int)
    val_in=val.y.isin(train.y).astype(int)

    #Comprobar que existen todos (todo 1)
    if all(x==1 for x in test_in):
        checkTest = True
    if all(x==1 for x in val_in):
        checkVal = True
    #Devuelve True si en test y val aparecen marcas que existen en train:
    if checkTest and checkVal:
        return True
    return False

from sklearn.model_selection import train_test_split
def split_datafiles(df):
    X_train, X_test = train_test_split(df, test_size=0.2 , random_state=random.randint(0,32),shuffle=True)
    X_train, X_val = train_test_split(X_train, test_size=0.14, random_state=random.randint(0,32),shuffle=True)
    return X_train, X_test ,X_val

#Dividir conjunto de datos:
shoes_train, shoes_test, shoes_val = split_datafiles(df_shoe_brand)

while checkBalancedSample(shoes_train, shoes_test, shoes_val) == False:
    shoes_train, shoes_test, shoes_val = split_datafiles(df_shoe_brand)

```

Figura 21: Código Python para la división de los datos

### 3.3. Objetivo 1: Predicción de la marca con la imagen de la huella de calzado

#### 3.3.1. Formatear datos

Ha sido necesario reformatear los datos, ya que en la tabla del *CSV* los datos están etiquetados en formato *individuo\_calzado*, y no dispone de una fila por cada imagen disponible, así que se ha creado la función *filesWithBrand* para disponer de los datos en formato X: nombre fichero, y: marca.

```
def filesWithBrand(shoeFiles):
    files = []
    brands = []
    for image in shoeFiles:
        files.append(image) #filename
        person = df[df['ID'].str[:6]==image[:6]] #persona+contador de calzado
        brands.append(person['Brand'].iloc[0])
```

Figura 22: Código y resultado de la función *filesWithBrand*

Además, existen registros en los que el valor de la marca del calzado es “None”. Se ha decidido limpiar los registros sin marca o que no aparecen un mínimo de veces en el listado mediante la función *filterMinSamples*:

```
def filterMinSamples(data, minSamples, deleteNone=True):
    if deleteNone == True:
        data=data[data['x']!='None'] #eliminar marca = "None"
        dataone=data[data['y']<minSamples] #marcas con pocas muestras
        data=data[data['y']>=minSamples] #marcas con mínimo "minSamples" muestras
        num_classes=len(data)
        print('Brands with at least '+str(minSamples)+' samples: %d' %num_classes)
        print('Brands with only 1 register: %d' %len(dataone))
    return data, dataone

dfbrandall = pd.DataFrame({'x':values_brand, 'y':counts_brand})

dfbrand, dfbrandone = filterMinSamples(dfbrandall, 5)

num_classes=len(dfbrand)
dfbrand = dfbrand.sort_values('y', ascending = False) #ordenar descendientemente
```

Figura 23: Código Python de la función *filterMinSamples*

### 3.3.2. Análisis de los datos disponibles

Para empezar, mediante el uso de la función `get_images` se recogen los nombres de ficheros (imágenes) que hay dentro de la carpeta `/images`. Además, se lee el fichero `Data-information.csv` que contiene la clasificación por persona, marca y modelo de las muestras (imágenes) tomadas.

Esta base de datos consiste en **1500 imágenes de la huella de 150 pares de zapatos**. El nombre del fichero de las imágenes tiene el formato AAABBL/RCC donde:

1. AAA son tres dígitos entre 001 y 028 que identifica al dueño del par de zapatos.
  2. BB dos dígitos entre 01 y 10 que indica el par de zapatos de un individuo.
  3. L/R para indicar si se trata del pie izquierdo o derecho.
  4. CC el número de replicación (imágenes diferentes del mismo individuo-zapato-pie).
- Además, en la tabla de resultados se registra, por cada par de zapatos de cada individuo el género (hombre o mujer), la talla en formato americano, la marca y, en algunos casos, el modelo de zapato.

Por último, se han analizado los datos de la tabla de resultados se observa que se dispone de 150 registros (correspondiente al número de diferentes zapatos en la muestra) de 28 individuos diferentes. Además, se dispone de 62 marcas de calzado diferentes, y las que más aparecen son Nike, Asics y Adidas. Por último, existen 70 registros de calzado de mujer y 80 de hombre.

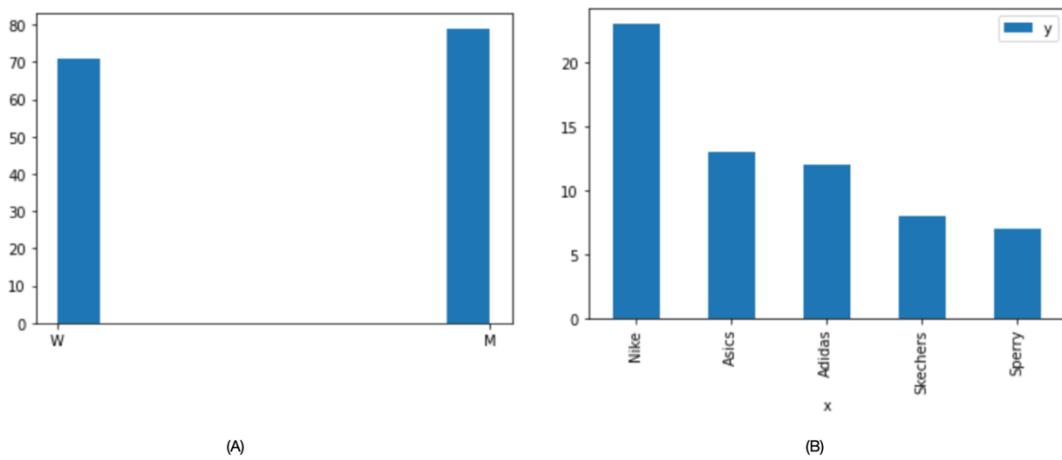


Figura 24: Gráfica de muestras por género (A) y marcas que más aparecen (B)

### 3.3.3. Extracción de características y clasificadores

En este apartado, se ha realizado un experimento con clasificadores conocidos para disponer de resultados con los que comparar los obtenidos con la red neuronal convolucional.

La extracción de características es el proceso de recuperar los rasgos más importantes de los datos sin procesar. Esto es encontrar el conjunto de parámetros que definen la forma de una imagen de manera precisa y única.

Existen diferentes técnicas para extraer las características de las imágenes, para este experimento se ha utilizado la técnica SIFT, que según el artículo *Classification of footwear outsole patterns using Fourier transform and local interest points* (N.Richetelli, MC.Lee, CA Lasky, ME.Gump y JA.Speir) [32], donde comparan el uso de SIFT con RANSAC, obtiene buenos resultados con imágenes de zapatos.

Se ha creado la función *extractSIFT* que se encarga de extraer las características de las imágenes que se mandan como parámetro, y *getFiles* que extrae las características de cada categoría (marca).

Durante la implementación se han añadido puntos donde se guardan los resultados utilizando *pickle* [33] para agilizar ejecuciones futuras, ya que el proceso de extracción de características tarda aproximadamente 40 minutos.

```

#extrae y calcula los descriptores SIFT para el conjunto de imágenes enviado
def extractSIFT(input_files):
    all_features_dict = {}
    feature_extractor = cv.SIFT.create()
    for i, fname in enumerate(input_files):
        rgb = cv.cvtColor(cv.imread("images/"+fname), cv.COLOR_BGR2RGB)
        gray = cv.cvtColor(rgb, cv.COLOR_RGB2GRAY)
        kp, desc = feature_extractor.detectAndCompute(gray, None)
        all_features_dict[fname] = desc
    return all_features_dict

#Esta función extrae las características de cada categoría (marca)
#input: listado de categorías (marcas), listado de ficheros, marca de cada fichero
#output: lista ficheros, lista categorías, lista de características
def getFiles(cat_list, X_files, y_values):
    all_files = []
    all_files_labels = {}
    all_features = {}
    cat_indexes = []
    cat_files = []
    cat_features = []

    #values_train contiene el listado de categorías sin repeticiones
    for cat, label in zip(cat_list, range(len(cat_list))):

        #primero buscar los indices en el listado de cada categoría (dentro bucle)
        cat_indexes = [i for i,x in enumerate(y_values) if x == cat]

        #como se saben las posiciones, se cogen esas imágenes de esa categoría:
        cat_files = [X_files.iloc[i] for i in cat_indexes]
        cat_features = extractSIFT(cat_files)
        all_files = all_files + cat_files
        all_features.update(cat_features)
        for i in cat_files:
            all_files_labels[i] = label
    return all_files, all_files_labels, all_features

```

Figura 25: Código de las funciones *extractSIFT* y *getFiles*.

A continuación, un ejemplo del resultado en una de las imágenes del conjunto de datos:

```

#mostrar una imagen con los puntos de interés
img = cv.imread("images/"+all_files_train[0])
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

features = cv2.SIFT_create()
keypoints = features.detect(gray, None)

img2=cv2.drawKeypoints(gray,keypoints,0,(0,0,255), flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv.imwrite('results/'+all_files_train[0], img2)

```

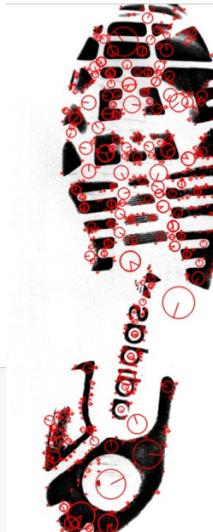


Figura 26: Código y resultado de una imagen con sus puntos de interés extraídos con SIFT.

### 3.3.4. Clasificadores

Una vez se dispone de las características, estas se encapsulan en un diccionario con tamaño de setenta y cinco elementos. Este proceso se denomina *Bag of Features* y consiste [33] en una representación desestructurada y sin información espacial de las características de las imágenes inspirada en el modelo *bag-of-words* del procesamiento del lenguaje natural.

```
#Se crea el Bag of Features con un diccionario de tamaño 75:  
dictionarySize =75  
if not os.path.exists("saved/bow_dict.pkl"):  
    BOW = cv.BOWKMeansTrainer(dictionarySize)  
    for feat in all_features_train:  
        BOW.add(all_features_train[feat])  
    dictionary = BOW.cluster()  
else:  
    with open('saved/bow_dict.pkl', 'rb') as fp:  
        dictionary = pickle.load(fp)  
print(dictionary.shape)  
print(all_features_train[all_files_train[0]].shape) #subdivisión de train: shoes_train
```

Figura 27: Código del proceso que crea el diccionario de características.

Se ha decidido utilizar tres de los clasificadores más comúnmente utilizados para verificar su eficacia con los vectores resultado de la extracción de características de los pasos anteriores.

- **Support Vector Machines (SVM)** [34]: Tiene como objetivo encontrar el hiperplano que clasifica claramente los puntos. Este hiperplano se calcula maximizando el margen de las instancias de entrenamiento en el espacio de destino.
- **DecisionTreeClassifier**: Las normas de clasificación se extraen construyendo un árbol de decisiones con los datos de entrenamiento. En cada nodo del árbol, se utiliza el atributo con mayor diferencia en entropía para dividir los datos.
- **KNeighborsClassifier**: Determina la clase de la información mirando los puntos cercanos. De manera que selecciona la clase (o grupo) que tiene más puntos de la instancia.

Por último, una tabla resumen de los resultados obtenidos usando los clasificadores anteriores:

Clasificador	Accuracy
SVM	0,98
KNeighborsClassifier	0,86
DesicionTreeClassifier	0,75

Se puede observar cómo SVM obtiene mejores resultados que los otros dos clasificadores. El clasificador SVM aparece en bastante literatura de clasificación de imágenes, por ejemplo, en *Footwear for Gender Recognition* (*Yuan Yuan; Yanwei Pang; Xuelong Li*) [35] donde lo utilizan para clasificar el calzado según el género de la persona.

El código de los clasificadores se encuentra en el apartado superior “Clasificadores” del cuaderno *TFM\_LauraRivera\_objectivo1\_marca.ipynb*.

Estos resultados se toman como referencia para comprobar la eficacia de las pruebas realizadas sobre la red neuronal convolucional.

### 3.3.5. Redes neuronales convolucionales

Las redes neuronales convolucionales, creadas inspirándose en las neuronas del cerebro, son capaces de aprender de forma progresiva en cada una de sus múltiples capas. Por lo general, las primeras capas, se encargan de la extracción de características con neuronas convolucionales y otras de reducción de la muestra, disminuyendo así su dimensión, mientras otras se encargan de clasificar características de forma local.

Es muy efectiva en matrices bidimensionales haciendo que sea popular su uso para la clasificación de imágenes 2d.

Se han definido unos parámetros a tener en cuenta para realizar experimentos con la red neuronal convolucional:

Parámetro	Descripción
doGray	Convertir las imágenes a escala de grises o no
doBin	Aplicar binarización para convertir la imagen a blanco y negro
doCrop	Recortar la imagen para eliminar columnas y filas en blanco
doRandom	Aplicar aumentación en las imágenes durante el entrenamiento
epoch	Iteraciones de entrenamiento
shape	Tamaño de las imágenes de entrada
minSamples	Número mínimo de muestras por marca. Por defecto 5
Shape	Tamaño de las imágenes

Con estos experimentos se quiere dar respuesta a las siguientes preguntas o **hipótesis**:

1. ¿Qué preprocessado de imágenes funciona mejor?
2. ¿Utilizar más *epochs*, resulta siempre en mejor resultado?
3. ¿Un mayor tamaño de las imágenes utilizadas, resulta en un mejor resultado?
4. ¿Cuál es el mínimo de muestras por marca aceptable para el modelo?

Como se comenta en el apartado anterior, se ha considerado importante filtrar las muestras a aquellas con una mínima participación por marca ya que el conjunto de datos dispone de muchas marcas que disponen de uno o pocas muestras, haciendo que el modelo se encontrara con marcas en los subconjuntos de *test* y validación que no existían en *train*.

Se han determinado las siguientes **consideraciones iniciales**:

- El valor mínimo de muestras por marca por defecto será de cinco, de manera que solo se tendrán en cuenta **marcas con al menos cinco muestras**.
- Se eliminan las muestras **sin marca** (etiquetadas como “None”).
- Las marcas deben aparecer como **mínimo una vez en cada subconjunto**.
- Para la programación del modelo se utiliza la librería **tensorflow** y **scikit-image** para las transformaciones de las imágenes.

### 3.3.5.1. Modelo propuesto

El modelo propuesto para esta red neuronal convolucional está inspirado en la solución del artículo de N.Viswanathan de la Universidad de Standford [32] para la identificación de artistas en cuadros de pintura.

Layer (type)	Output Shape	Param #
<hr/>		
conv2d (Conv2D)	(None, 280, 832, 16)	160
max_pooling2d (MaxPooling2D)	(None, 140, 416, 16)	0
conv2d_1 (Conv2D)	(None, 140, 416, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 70, 208, 32)	0
conv2d_2 (Conv2D)	(None, 70, 208, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 35, 104, 64)	0
flatten (Flatten)	(None, 232960)	0
dense (Dense)	(None, 128)	29819008
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 7)	903
flatten_1 (Flatten)	(None, 7)	0
<hr/>		
Total params:	29,843,207	
Trainable params:	29,843,207	
Non-trainable params:	0	

Figura 28: Capas del modelo de red neuronal propuesta

Que se ha creado mediante una función que permite configurar los parámetros mencionados anteriormente, y cuyo fragmento de código crea el modelo:

```
model_test = models.Sequential([
    Conv2D(16, 3, padding='same', activation='relu', input_shape=shape),
    MaxPooling2D(),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(n, activation='softmax'),
    Flatten()
])
```

Figura 29: código que crea el modelo

A continuación, una breve descripción de las capas utilizadas:

**Dense:** La capa densa o totalmente conectada, se caracteriza porque cada nodo de entrada se relaciona con cada nodo de salida.

**Conv2D:** En cambio la capa convolucional, se le indica por parámetro el número de kernels (16, por ejemplo) y tamaño de los kernel (3x3), para definir que nodos se van a relacionar con los de salida. Esto aporta flexibilidad al aprendizaje, además de disminuir el tamaño de las capas.

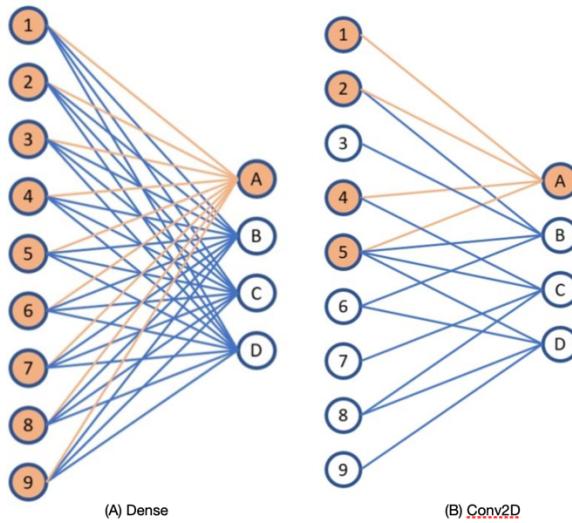


Figura 30: Representación gráfica Dense vs Con2D

Tanto a las capas densas como las convolucionales necesitan de una función de activación. Existen diferentes funciones de activación, a continuación, se nombran las utilizadas en el proyecto:

- **Sigmoid:** Función no lineal que devuelve un resultado entre 0 y 1. Se caracteriza por su representación gráfica en forma de S.
- **Tanh:** Mejora algunos problemas conocidos de la activación sigmoid.
- **Relu:** Función no lineal muy utilizada por su simplicidad y mayor rendimiento que las anteriores.
- **Softmax:** Es común su uso en la última capa del modelo ya que calcula las probabilidades por cada uno de los eventos (o clases).

**Max\_Pooling2D:** Es un proceso de discretización que permite reducir el tamaño de parámetros, por lo que a su vez reduce el coste de cálculo y de tiempo total de ejecución.

**Flatten:** Encargada de convertir la imagen a una dimensión.

**Dropout:** Se encarga de eliminar características a analizar, para quedarse con las más relevantes, en el modelo propuesto se usa la mitad de las características (valor: 0.5).

### 3.3.5.2. Optimizadores

Las redes neuronales son algoritmos paramétricos y en cada iteración, se asigna un peso a esos parámetros conforme esta se entrena. Los optimizadores se encargan de optimizar estos parámetros para reducir el error (*loss*).

Para este proyecto se utiliza **Adam** (*adaptive momento estimation*), este método es una combinación de *RMSprop* y *SGD* y mantiene las tasas de aprendizaje separadas por cada parámetro, manteniendo una única tasa de aprendizaje para todo el entrenamiento. Según el artículo *Adam Optimization Algorithm (Mohammed Alom)* [37] este método es adecuado para cualquier tipo de modelo, además requiere menos memoria que otros y es computacionalmente eficiente

### 3.3.5.3. Preprocesado y generadores de las imágenes

Los generadores se utilizan para generar subconjuntos de los datos en tiempo real e ir alimentando el modelo conforme lo necesita. Estos se utilizan sobre todo cuando se dispone de conjuntos de datos grandes y cargarlos puede afectar a la memoria de la computadora. Además, aunque para este proyecto no se aprovecha esta característica de los generadores, estos subconjuntos se pueden tratar en diferentes *cores* de la GPU del servidor donde se ejecutan, optimizando los tiempos de entrenamiento del modelo en cuestión. Además, permite añadir el preproceso que necesiten, en este caso, las imágenes, en el caso de este proyecto se ha utilizado un Generador<sup>1</sup> para incorporar los diferentes preprocesados en las imágenes, incorporando parámetros y aplicando más o menos procesos dependiendo de estos.

---

```

class DataGenerator2dFootwear(Sequence):
    def __init__(self,data, df_shoe_brand,doRandomize=False,imgPath='images', doGray=True,doBin=True,
                 doCrop = True,batchSize=10):
        # Store parameters
        self.imgPath=imgPath
        self.fileNames=data.copy()
        self.batchSize=batchSize
        self.doRandomize=doRandomize
        self.df_shoe_brand=df_shoe_brand
        self.doGray=doGray
        self.doBin=doBin
        self.doCrop=doCrop
        # Get number of files (to avoid computing them later)
        self.numImages=len(data)
        # Shuffle them if required
        self.on_epoch_end()

    # Input : theIndex - Index of the image to load within self.fileNames.
    # Output : theImage - Loaded (and possibly transformed) image. Must be
    #           of float type with values within [0,1]
    #           theClass - Shoe brand
    def _load_image_(self,theIndex):

        file = self.fileNames[theIndex]

        img = io.imread(self.imgPath+file)
        h,w,c = img.shape #guardar el shape por si se hace crop poder hacer el resize

        #transformations...
        theImage = img_as_float(img)

        theImage=theImage /255.0 #normalizar (quito rescaling del modelo)

        if self.doRandomize: #añadir aumentación a las imágenes:
            theImage=create_variation(img,random.choice([True, False]),random.choice([True, False]),random.choice([True, False]))
        #Buscar la imagen en el csv para extraer la Marca:
        person = self.df_shoe_brand[self.df_shoe_brand['X'].str[:6]==file[:6]] #persona+contador de calzado
        theClass = person['factor_brand'].iloc[0]
        return theImage,theClass

```

Figura 31: Código del generador

Como se indica al inicio de este capítulo, se aplican diferentes transformaciones a las imágenes para experimentar que preprocesso obtiene mejores resultados. Estas transformaciones se realizan en la función `_load_image_` del generador:

```

if self.doGray:#escala de grises
    img = rgb2gray(img)
    #plot_image_grey(img)
if self.doBin: #blanco y negro
    test_binary_high,img = cv.threshold(img,0, 255, cv2.THRESH_BINARY)
if self.doCrop: #quitar columnas/filas blancas
    img = crop_image(img)
    img = cv2.resize(img, (h,w), interpolation = cv2.INTER_AREA)

```

Figura 32: Fragmento de código donde se transforma la imagen

Estas transformaciones son:

- **Convertir a grises** mediante la función `rgb2gray`, que convierte la imagen a gris y a una dimensión.
- **Convertir a blanco y negro** aplicando un *threshold* binario, modificar todos los pixeles a partir de un rango para que sean únicamente blanco o negro.
- **Eliminar espacios** en blanco recortando aquellas filas y columnas totalmente en blanco.

Estas transformaciones se realizan con la intención de eliminar características que no influyen en el reconocimiento del patrón de la marca, por ejemplo, la tonalidad de gris, en este caso se considera interesante saber la posición de la huella (pixeles negros), no el color o tonalidad de estos. Lo mismo con el espacio alrededor de la huella, todas las imágenes han estado tomadas con una guía métrica, si se elimina el espacio libre alrededor, no debería importar el tamaño para identificar la marca.

Por último, en el caso de indicar que se aplique aumentación, se utiliza la función `create_variation`, que crea variaciones de la imagen añadiendo, de forma aleatoria, diferentes alteraciones:

```
def create_variation(theImage,doFlip,doNoise,doRotate):
    image = img_as_float(theImage)
    if doFlip==True:
        image = np.fliplr(image)
    if doNoise==True:
        image = util.random_noise(image)
    if doRotate==True:
        image = transform.rotate(image, random.randint(-45, 45),mode='symmetric')
    return image
```

Figura 33: Código función que añade aumentación a las imágenes

### 3.3.5.4. Aumentación

La aumentación es una técnica efectiva [39] para reducir el sobreentrenamiento agregando más muestras de entrenamiento perturbando las imágenes ya disponibles. Puede ser particularmente efectivo en tareas de clasificación de imágenes, ya que a veces es difícil disponer de más ejemplos etiquetados y porque las clases no deberían cambiar bajo pequeñas perturbaciones locales. Estas distorsiones no añaden únicamente ruido a la imagen, sino que pretenden mover los pixeles para introducir mayores cambios en el espacio vectorial de entrada a la vez que conservan el significado semántico de la muestra.

El sobreentrenamiento (**overfitting**) se da cuando una inteligencia artificial aprende de manera que solo es aplicable sobre la muestra de entrenamiento y ya no se puede generalizar ese conocimiento a la población general (muestra de *test*).

Tal y como mencionan en el artículo *Understanding artificial intelligence based radiology studies: What is overfitting (Simukayi Mutasa, Shawn Sun, and Richard Ha)* [33] este sobreentrenamiento puede ocurrir cuando se disponen de una muestra pequeña de datos, como en este caso no es posible añadir más muestras al conjunto de datos, se suele utilizar la aumentación para ampliar el conjunto de muestras con variaciones de las originales, las transformaciones más comunes:

- Flip (voltear)
- Crop (recortar)
- Rotate (rotación)
- Noise (añadir ruido o distorsiones en las imágenes)

Otro recurso para el sobreentrenamiento es utilizar la capa Dropout en el modelo, esta se encarga de eliminar características que no aportan suficiente información al modelo.

Como la muestra de datos disponible para este proyecto es limitada, se ha decidido utilizar aumentación e incorporar *Dropout* al modelo.

### 3.3.5.5. Experimentos con la red propuesta

Para todos los experimentos, se muestra la siguiente información para poder analizar y comparar las diferentes variantes del modelo:

- **Eficacia** (en *train*, *validation* y *test*)
- **Loss** (en *train*, *validation* y *test*): evalúa la desviación entre predicciones y los valores reales.
- **Top 1**: Porcentaje de muestras en las que la predicción de marca correcta corresponde a la de mayor probabilidad (equivalente a la eficacia en la predicción con los datos de *test*).
- **Top 3**: Porcentaje de muestras de los datos de *test* en las que la marca correcta se encuentra dentro de las tres con mayor probabilidad.
- **% de similitud**: Comprueba si el valor de similitud (entre 0 y 1) es aceptable, para ello se han comprobado los cuantiles 25%, 50%, 75%.
- **Matriz de confusión**: muestra en forma de matriz el desempeño del algoritmo ya que cada columna representa las predicciones y las filas la clase real permitiendo tener una visión de los aciertos y fallos.
- **Tiempo**: En los experimentos de tamaño de la imagen y de uso de diferentes capas en el modelo, también se ha considerado la variable tiempo para analizar resultados.

Se decide estos parámetros porque, no solo se desea comprobar si la marca con mayor probabilidad es la correcta, sino también saber ese porcentaje de similitud, ya que comprobando solo el de mayor porcentaje, se podría dar como solución una marca con muy baja similitud, aunque fuera la de mayor valor.

Los experimentos se han realizado en un *Macbook Pro* con procesador Intel i7 Ram 16 GB y grafica *Radeon Pro 555 2 GB*.

### 3.3.5.5.1. ¿Qué preprocesado funciona mejor?

Para comprobar que preprocesado obtiene mejores resultados se han realizado diferentes combinaciones de transformaciones de las imágenes en combinación con dos valores de epochs (10 y 25) y un tamaño de imagen concreta (280,832) correspondiente a la mitad, aproximadamente, del tamaño original. Las diferentes transformaciones de las imágenes se especifican con las siguientes siglas para simplificar:

A	Uso de aumentación
G	Convertir a escala de grises
B	Convertir a blanco y negro ( <i>Binary</i> )
C	Recortar filas y columnas en blanco

Se puede observar que **transformar las imágenes a blanco y negro y recortar los espacios en blanco**, obtiene una **eficacia superior** al evaluar las imágenes de *test* (61% y 79% en el caso de utilizar *epoch*=10 y 66% y 77% con *epoch*=25), eso se debe a que se limita al modelo a aprender de las características de la huella sin importar el color o tonalidad de gris ni la posición de la huella dentro del espacio de la imagen. En la matriz de confusión se puede observar en la diagonal como la marca etiquetada corresponde a la de mayor porcentaje, aunque si observamos las columnas de porcentaje de similitud solo el 11% está por encima del 75%, por lo que las huellas evaluadas aciertan con poca similitud.

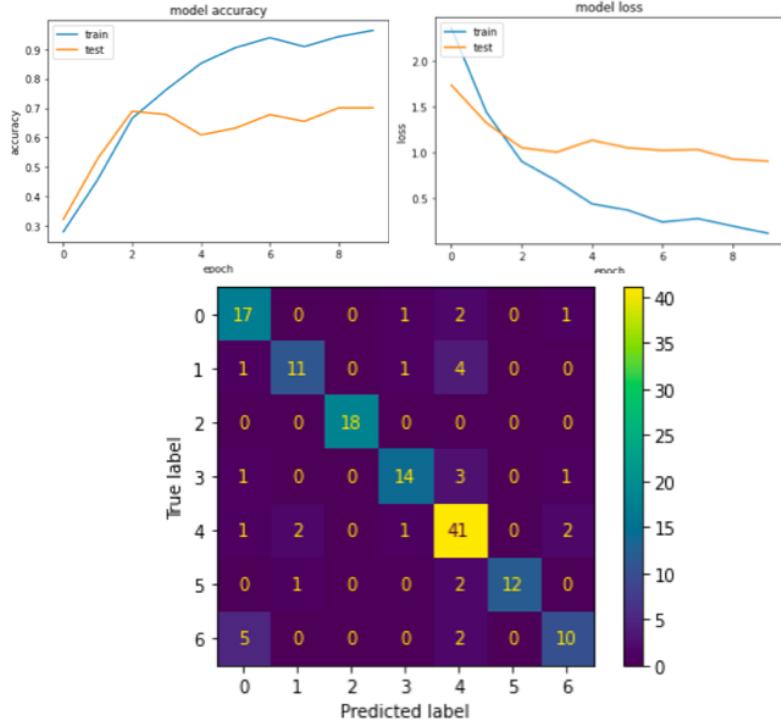


Figura 34: *accuracy*, *loss* y matriz de confusión para GBC  $epoch=10$

Además, con el uso de **aumentación** en las imágenes de entrenamiento se obtiene **peores resultados** en cuanto a la eficacia aplicándolo tanto en la imagen sin preprocessar (26% con  $epoch=10$  y 15% con  $epoch=25$ ) como en la imagen en blanco y negro (15% con y 43%). Se considera un comportamiento habitual si tenemos en cuenta que esto añade variaciones en las imágenes para evitar que el modelo se sobreentrene. También, se observa que los valores de *loss* en el caso del uso de aumentación son elevados respecto a no utilizarlo.

Por otro lado, se observa que, aunque la eficacia es inferior, el porcentaje de similitud es mayor, por lo que se podría concluir que la aumentación contribuye a una mayor seguridad en el resultado en esas muestras con una predicción correcta, por ejemplo, en el caso de  $epoch=10$  y preprocessado AGBC se obtiene una eficacia del 15%, pero todas ellas tienen una **similitud del 75%** o más.

Hay que destacar que en el experimento aumentado con  $epoch=25$ , este porcentaje de similitud, contrario al resultado esperado, disminuye al 25%, aunque la eficacia aumenta al 43%. En este punto se podría abrir debate sobre si realmente cada marca dispone de suelas únicas o existe mucha similitud entre ellas.

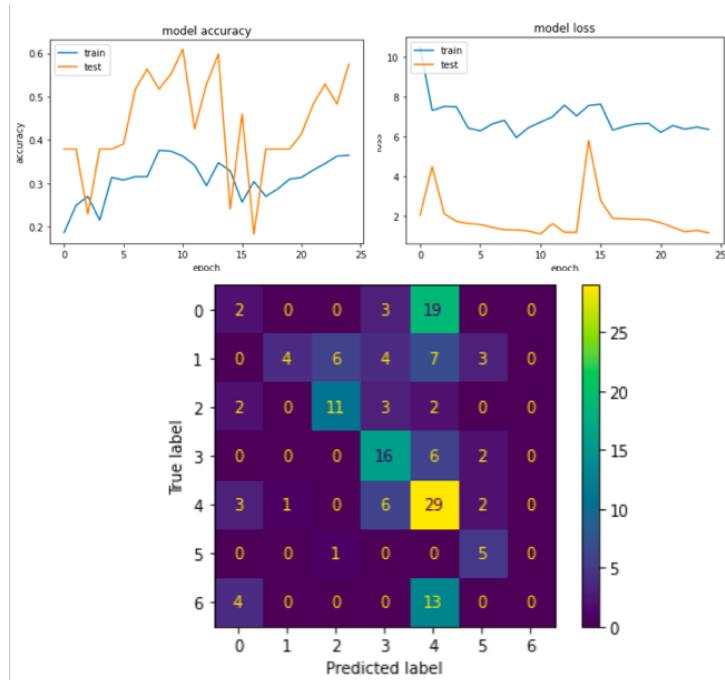


Figura 35: *accuracy*, *loss* y matriz de confusión para AGBC *epoch*=25

En la imagen se puede observar cómo tanto la eficacia como el *loss*, a diferencia de la gráfica anterior, no se estabiliza. Además, en la matriz de confusión se puede observar como la predicción que más aparece suele ser el 4, correspondiente a la marca Nike que dispone de mayor número de muestras.

A continuación, la tabla resumen de los experimentos realizados con el modelo con diferentes combinaciones de aumentación, preprocesado y *epochs*:

Epoch	Shape	Pre	Train				Test				% similitud marca
			Acc	Val_acc	Loss	Val_loss	Top 1	Top 3	25%	50%	
10	(280,832,3)	-	0,31	0,32	1,85	1,83	1,88	0,30	0,56	0,3	0
		A	0,31	0,37	11,06	10,00	11,82	0,26	0,57	0,26	0,26
		G	0,31	0,32	1,84	1,84	1,87	0,31	0,56	0,3	0
		GB	0,94	0,52	0,18	1,69	1,43	0,61	0,84	0,12	0,12
		AG	0,79	0,06	0,62	4,43	4,17	0,15	0,46	0,31	0,15
	(280,832,1)	GBC	0,96	0,70	0,12	0,90	0,73	0,79	0,94	0,11	0,11
		AGBC	0,15	0,14	13,58	13,70	13,60	0,15	0,46	0,15	0,15
		-	0,31	0,32	1,81	1,84	1,90	0,1	0,56	0,3	0
		A	0,86	0,18	0,35	8,10	9,58	0,15	0,57	0,15	0,15
		G	0,31	0,32	1,81	1,83	1,89	0,3	0,56	0,30	0
25	(280,832,1)	GB	0,99	0,59	0,03	1,92	1,72	0,66	0,90	0,12	0,12
		AG	0,90	0,06	0,26	11,69	11,00	0,15	0,57	0,16	0,16
		GBC	0,99	0,79	0,03	1,35	1,11	0,77	0,96	0,11	0,11
		AGBC	0,36	0,57	6,36	1,17	1,40	0,43	0,79	0,42	0
		-	-	-	-	-	-	-	-	-	0

Figura 36: Tabla resumen de los experimentos según preprocesado

### 3.3.5.5.2. ¿Más epoch resulta en mejor resultado?

Para comprobar el impacto del número de *epochs* en el resultado se ha escogido cinco configuraciones de preprocessado: sin preprocessado ni aumentación, sólo aumentación, imágenes en blanco y negro, imágenes en blanco y negro sin espacios en blanco sin y con aumentación. Los experimentos (Fig.37) se realizan con cuatro valores de epoch diferentes: 5, 10, 25 y 50.

Para empezar, se observa que en el caso de no aplicar ninguna transformación en las imágenes la eficacia, tanto durante el entrenamiento y evaluación, no varía e, **independientemente de las iteraciones, siempre ronda el 31% y 30%**. Esto se puede deber a que se dispone de una muestra limitada de datos y, aunque se aumente el valor de *epoch*, la red neuronal no aprende más.

En cambio, aplicando únicamente **aumentación**, se observa como la eficacia en entrenamiento aumenta progresivamente hasta el 95% con *epoch*=50, pero la eficacia con *test* se mantiene deficiente con un 15%.

Si se aplican las transformaciones sin aumentación, se puede observar que un aumento en el valor de epoch mejora muy poco la eficacia en general (5% más respecto epoch=5 y epoch=50), además se observa, es que el valor *loss* de entrenamiento disminuye, pero esto no parece afectar a la eficacia de evaluación. Esto significa que el modelo está **sobreentrenado** y requiere de mecanismos para evitarlo.

Por último, al aplicar las **transformaciones con aumentación**, se observa que la eficacia aumenta en general, excepto en el salto de *epoch*=25 a *epoch*=50, donde se puede observar que la eficacia decae del 43% al 38%. Hay que destacar que, aunque la eficacia disminuye ligeramente, el porcentaje de similitud aumenta, reconociendo el 13% de las marcas con una similitud de hasta el 75%.

De estos experimentos, se puede concluir que el **valor de epoch utilizado influye tan solo si utilizamos aumentación**, ya que damos la oportunidad a la red neuronal que observe más variaciones de las imágenes, pero un valor muy elevado no necesariamente obtendrá una mejor eficacia, aunque el porcentaje de similitud aumenta.

Shape	pre	epoch	Train				Test				% similitud marca
			Acc	Val_acc	Loss	Val_loss	Top 1	Top 3	25%	50%	
(280,832,3)	-	5	0,32	0,32	1,84	1,83	1,87	0,30	0,56	0,30	0
		10	0,31	0,32	1,85	1,83	1,88	0,30	0,56	0,30	0
		25	0,31	0,32	1,81	1,84	1,90	0,1	0,56	0,30	0
	A	50	0,31	0,32	1,8	1,85	1,92	0,30	0,56	0,30	0
		5	0,50	0,09	1,34	1,94	1,95	0,04	0,33	0	0
		10	0,31	0,37	11,06	10,00	11,82	0,26	0,57	0,26	0,26
	GB	25	0,86	0,18	0,35	8,10	9,58	0,15	0,57	0,15	0,15
		50	0,95	0,18	0,13	8,24	8,21	0,15	0,57	0,31	0,15
		10	0,94	0,52	0,18	1,69	1,43	0,61	0,84	0,12	0,12
(280,832,1)	GBC	25	0,99	0,59	0,03	1,92	0,66	0,66	0,90	0,12	0,12
		50	0,99	0,58	0,03	2,55	2,00	0,65	0,90	0,12	0,12
		5	0,85	0,66	0,43	1,04	1,06	0,68	0,91	0,11	0,11
	AGBC	10	0,96	0,70	0,12	0,90	0,73	0,79	0,94	0,11	0,11
		25	0,99	0,79	0,03	1,35	1,11	0,77	0,96	0,11	0,11
		50	0,99	0,72	0,01	1,34	1,03	0,81	0,97	0,11	0,11
		5	0,30	0,45	6,26	1,50	1,71	0,30	0,75	0,26	0,26
		10	0,15	0,14	13,58	13,70	13,60	0,15	0,46	0,15	0,15
		25	0,36	0,57	6,36	1,17	1,40	0,43	0,79	0,42	0
		50	0,47	0,35	6,67	8,19	7,10	0,38	0,51	0,13	0,13

Figura 37: Tabla resumen de los experimentos con diferentes valores epoch

### 3.3.5.5.3. ¿Cuál es el mínimo de muestras por marca aceptable para el modelo?

Por defecto, el resto de los experimentos se han realizado con marcas con como mínimo 5 muestras, el objetivo de este apartado es ver que ocurre al añadir marcas con más o menos muestras. Todos estos experimentos se han realizado con tamaño de imagen (280,832),  $epoch = 10$  y con cuatro transformaciones diferentes (Fig. 40).

Se puede observar que, si se utilizan todas las muestras, incluidas las marcas con una única muestra la eficacia disminuye considerablemente, respecto a utilizar solo las marcas con al menos 5 muestras, para cualquiera de las transformaciones testeadas, ya que, en el mejor de los casos, con imágenes en blanco y negro recortadas, es del 53% respecto el 79%. Además, en el caso del porcentaje de similitud solo el 15% es elevado, el resto de los aciertos se encuentran por debajo del 25%.

Si nos fijamos en los resultados para registros con marcas con al menos dos muestras, se puede observar que se obtiene una eficacia del 70% aunque con muy poca similitud, pero al aplicar aumentación disminuye al 24%, que si observamos su matriz de confusión siempre predice las mismas dos marcas, seguramente las que disponen de más muestras.

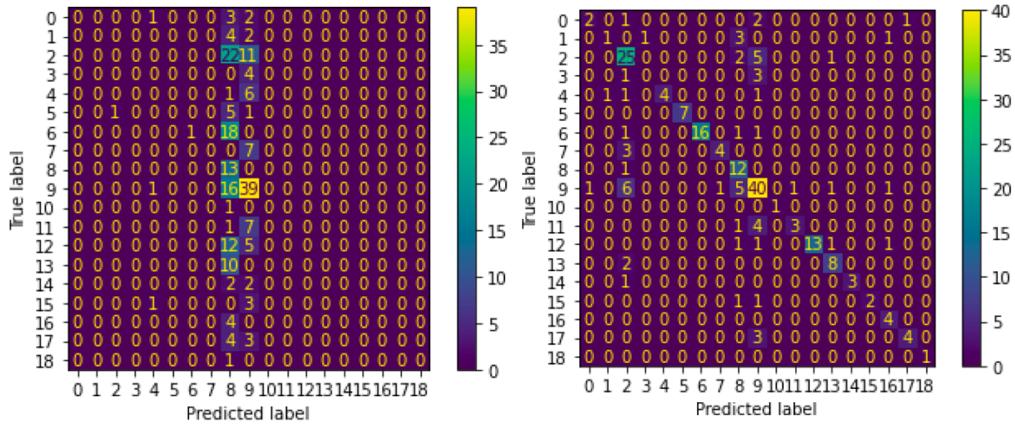


Figura 38: Matriz de confusión con marcas con mínimo 2 muestras con y sin aumentación.

Utilizar más o menos muestras implica añadir más o menos marcas (clases) a clasificar, se puede observar que en el caso de utilizar todas las muestras hay 59 marcas diferentes, respecto las 3 si nos centramos en aquellas de 10 o más muestras, esto tiene un impacto directo en el porcentaje de similitud, ya que en el primer caso no se alcanza en ningún

caso el 25%, mientras que en el último se alcanza el 75% de similitud en el 57% de muestras.

Seguramente, si el modelo se centrara en una única marca, por ejemplo, es Nike o no, los resultados fueran aún mejor. A continuación, la matriz de confusión utilizando solo las 3 marcas con más muestras:

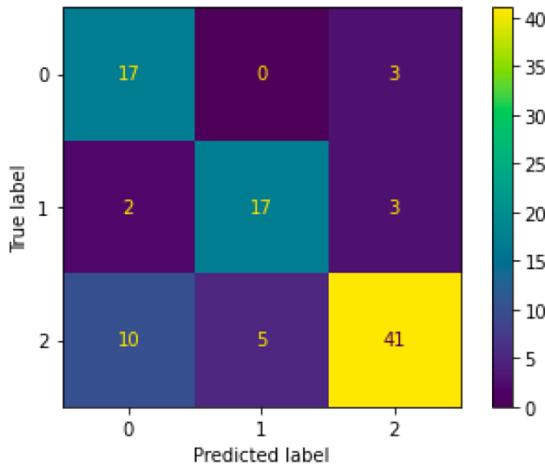


Figura 39: Matriz de confusión con marcas con mínimo 10 muestras sin aumento.

Se podría concluir que al disponer de un número reducido de clases con un mayor volumen de muestras permite a la red neuronal distinguir mejor las imágenes de cada clase.

En el caso opuesto, si se dispone de muy pocas muestras por clase, esto afecta directamente a la eficacia ya que no dispone de suficientes muestras para buscar similitudes y patrones que permitan una clasificación fiable. En estos casos, se podría utilizar otro tipo de técnicas, como el ***One-Shot Learning***, que en lugar de utilizar varias muestras para aprender las características de una clase y devolver una predicción de la clase, utiliza una única observación para comparar e identificar si es de una clase o no [41].

min Sample	N	brands	pre	Train			Test			% similitud marca	
				Acc	Val_acc	Loss	Val_loss	Loss	Top 1	Top 3	25%
5	770	7	-	0,31	0,32	1,85	1,83	1,88	0,30	0,56	0,3
			A	0,31	0,37	11,06	10,00	11,82	0,26	0,57	0,26
			GBC	0,31	0,32	1,81	1,84	0,73	0,79	0,56	0,3
3	930	12	AGBC	0,15	0,14	13,58	13,70	13,60	0,15	0,46	0,15
			-	0,25	0,22	2,30	2,34	2,26	0,32	0,55	0
			A	0,64	0,23	1,13	7,60	7,15	0,22	0,50	0,34
2	1070	19	GBC	0,08	0,08	14,83	14,89	15,16	0,05	0,13	0,06
			AGBC	0,19	0,23	8,24	2,13	2,17	0,22	0,58	0,22
			-	0,20	0,23	2,65	2,63	2,59	0,26	0,47	0
1	1470	59	A	0,50	0,05	1,61	7,28	8,32	0,02	0,32	0
			GBC	0,95	0,68	0,19	1,87	1,34	0,70	0,91	0,15
			AGBC	0,19	0,28	7,93	2,30	2,36	0,24	0,45	0
10	56	3	-	0,16	0,18	3,55	3,49	3,60	0,16	0,33	0
			A	0,40	0,006	2,25	13,03	13,38	0,01	0,18	0,01
			GBC	0,68	0,58	1,05	1,8	1,80	0,53	0,75	0,15
			AGBC	0,16	0,16	3,75	3,75	3,78	0,15	0,30	0
			-	0,47	0,41	1,05	1,08	1,00	0,57	1	1
			A	0,50	0,49	8,03	8,20	8,71	0,45	1	0,45
			GBC	0,96	0,83	0,09	0,66	0,93	0,76	1	0,57
			AGBC	0,24	0,23	12,1	12,3	11,34	0,29	1	0,29

Figura 40: Tabla resumen de los experimentos con diferentes muestras mínimas

### 3.3.5.5.4. ¿Mayor tamaño de imagen obtiene mejores resultados?

Se ha experimentado con el tamaño de las imágenes para poder observar si éste influye en el resultado. Para ello, se ha seleccionado una muestra de configuraciones para ver su resultado con imágenes de mayor y menor tamaño. Esta configuración es *epoch*=10 utilizando las transformaciones a blanco y negro recortando los espacios en blanco, con y sin aumentación, en total, dos configuraciones diferentes por cada uno de los cuatro tamaños (Fig. 41).

En el caso de imágenes pequeñas 104x228 se ha omitido la transformación *crop* ya que recortaba la imagen entera al eliminar los blancos, lo que se puede interpretar que las imágenes resultan de muy mala calidad. También se intentó realizar el experimento con el **tamaño original** de las imágenes (1645x4212) pero se obtenía errores por la memoria de computación y se ha utilizado tamaños desde 411x1035 hasta 104x228.

Para empezar, se observa que el **tiempo** necesario para entrenar el modelo tiene **correlación con el tamaño** de las imágenes, con imágenes mayor ha tardado alrededor de 30 minutos, mientras que con las menores 13 minutos. Esto tiene explicación en que la red neuronal toma como entrada el total de pixeles de las imágenes.

Si se observa la eficacia de evaluación *Top1*, se observa que es ligeramente mejor conforme el tamaño es mayor, pero donde se observa mayor diferencia es en los casos que se ha aplicado **aumentación**, ya que para imágenes de 104x228 es de **27%** pero para imágenes de 411x1035 de **64%**.

Se podría concluir que el tamaño de la imagen tiene relación directa con el tiempo y memoria necesarios para entrenar el modelo, así que es algo para tener en cuenta al inicio del proyecto. También que mayor tamaño favorece en el caso de utilizar **aumentación**, ya que al disponer de más pixeles dispone de más detalles para su aprendizaje, pero **sin aumentación** no se observa mucha diferencia entre las de tamaño de 411x1035 (eficacia del 79%) y las de tamaño a la mitad 205x526 (78%). Para imágenes más pequeñas, las transformaciones daban problemas y la eficacia disminuye bastante.

Estos experimentos se encuentran en el apartado “Diferentes medidas de las imágenes” del cuaderno *TFM\_LauraRivera\_objectivo1\_image\_size\_layers\_tests.ipynb*.

Shape	pre	Time	Train			Test			% similitud marca			
			Acc	Val acc	Loss	Val loss	Loss	Top 1	Top 3	25%	50%	75%
(411,1035,1)	GBC	33:54	0,98	0,79	0,03	0,68	1,03	0,79	0,96	0,26	0,26	0,26
	AGBC	22:33	0,98	0,70	7,14	0,90	0,99	0,64	0,92	0,10	0,10	0,10
(280,832,1)	GBC	29:59	0,97	0,82	0,10	0,87	0,82	0,77	0,97	0,26	0,26	0,26
	AGBC	21:36	0,46	0,58	1,43	1,19	1,18	0,54	0,85	0,1	0,1	0
(205,526,1)	GBC	18:18	0,99	0,74	0,05	1,33	1,14	0,78	0,92	0,26	0,26	0,26
	AGBC	14:57	0,31	0,57	7,57	1,22	1,31	0,52	0,87	0,10	0,10	0
(104,228,1)	GB	13:01	0,85	0,77	0,38	0,87	0,87	0,67	0,96	0,29	0	0
	AGB	13:29	0,37	0,29	5,85	2,00	2,21	0,27	0,71	0,21	0	0

Figura 41: Tabla resumen de los experimentos con diferente tamaño de imagen

### 3.3.5.6. Variaciones en el modelo

Durante el proceso de creación del modelo, se han testeado diferentes variantes con más o menos capas, en la Figura 42 se muestra una comparativa utilizando una configuración por defecto de epoch=10, tamaño 280x832 y preprocesado a blanco y negro (con y sin aumentación). El experimento, disponible en el apartado “Variaciones en las capas” del cuaderno, consiste en eliminar capas del modelo del proyecto para comprobar si estas afectan al resultado y comprender mejor su funcionalidad.

Se puede observar que al eliminar la capa **Dropout**, la **eficacia disminuye** considerablemente tanto en entrenamiento como evaluación, además los valores de *loss* aumentan. También se observa que al utilizarla con un valor inferior (0.1) o mayor (0.8) respecto al utilizado por defecto (0.5) la eficacia también se ve disminuida, esto último ocurre porque, seguramente, en el primer caso se estén eliminando demasiadas características, y en el segundo se estén utilizando demasiadas que no aportan valor a la clasificación.

En el caso de eliminar la capa **Dense** añadida antes de añadir la de Dropout no afecta apenas al resultado de la eficacia, pero **reduce ligeramente el tiempo** de ejecución.

La diferencia más significativa es en el uso de la capa **maxPooling**, para empezar los **parámetros para entrenar aumentan 15 veces** su tamaño, haciendo que el **tiempo se ejecución aumente** más de una hora, además la eficacia en *test* también se reduce.

La eficacia también se ve reducida al prescindir de las **capas convolucionales ocultas**, además el número de parámetros se triplica y el tiempo de ejecución aumenta un 30%.

Por último, se han realizado pruebas con dos **funciones de activación** alternativas, *sigmoid* y *tanh*, pero en ambos casos, tanto con y sin aumentación, se obtienen peores valores de eficacia. Cabe remarcar que en el caso de *sigmoid* se obtiene un 75% de similitud en el 33% de las predicciones.

variación	params	time	pre	Train			Test			% similitud marca	
				Acc	Val_acc	Loss	Val_loss	Loss	Top 1	Top 3	25%
Ninguna	29843207	09:07	GBC	1	0,81	<0,01	1,61	1	0,82	0,98	0,15
	08:46	AGBC	0,31	0,37	11,06	10,00	11,82	0,26	0,57	0,26	0,15
Sin dropout	29843207	10:53	GBC	0,15	0,19	13,68	12,96	13,71	0,15	0,38	0,15
	09:46	AGBC	0,19	0,18	13,37	13,15	13,60	0,15	0,42	0,15	0,15
Dropout 0.1	29843207	11:29	GBC	0,17	0,16	13,28	13,52	13,71	0,15	0,37	0,15
	09:50	AGBC	0,31	0,37	11,06	10,00	11,82	0,26	0,57	0,26	0,26
Dropout 0.8	29843207	09:53	GBC	0,74	0,82	0,71	0,53	0,62	0,74	0,96	0,15
	09:47	AGBC	0,33	0,59	7,10	1,24	1,66	0,38	0,78	0,31	0
Sin softmax	29842304	09:06	GBC	0,18	0,31	19,81	10,51	10,11	0,33	0,56	0,41
	09:12	AGBC	0,21	0,18	10,05	6,65	9,78	0,15	0,46	0,46	0,46
Sin Dense	1654023	08:37	GBC	1	0,79	0,001	1,20	0,79	0,80	0,96	0,15
	07:14	AGBC	0,38	0,56	6,17	1,15	1,36	0,38	0,79	0,42	0
Sin maxPooling	477126407	1:39:30	GBC	0,30	0,31	11,15	11,11	10,78	0,33	0,60	0,33
	1:07:58	AGBC	0,08	0,08	14,74	14,82	14,23	0,11	0,42	0,11	0,11
Sin conv extra	119276711	12:01	GBC	0,30	0,31	11,21	11,11	10,78	0,33	0,60	0,33
	11:16	AGBC	0,13	0,15	13,97	13,70	13,60	0,15	0,46	0,15	0,15
Act = sigmoid	29843207	09:06	GBC	0,17	0,16	13,28	13,52	13,71	0,15	0,37	0,15
Act = tanh	29843207	08:50	GBC	0,30	0,31	11,18	11,11	10,78	0,30	0,60	0,33
	08:59	AGBC	0,15	0,15	13,58	13,70	13,60	0,15	0,46	0,15	0,15
Sin Flatten	Error cross entropy										

Figura 42: Tabla resumen de los experimentos con variaciones en las capas

### 3.3.5.7. Autokeras

*Autokeras* [42] es un sistema basado en Keras que utiliza componentes pre-creados para generar un modelo que obtenga buenos resultados con la muestra facilitada. Se ha decidido utilizarlo para comprobar el modelo que se obtenía y ver si realmente obtiene un buen resultado.

A continuación, una captura del modelo creado por *autokeras*:

```
model = clf.export_model()

# Capas que ha creado como mejor modelo:
model.summary()

plot_history_2(ahistory)

Model: "model"

Layer (type)          Output Shape       Param #
===== =====
input_1 (InputLayer)   [(None, 832, 280, 3)]    0
cast_to_float32 (CastToFloa  (None, 832, 280, 3)    0
t32)

normalization (Normalizatio  (None, 832, 280, 3)    7
n)

conv2d (Conv2D)         (None, 830, 278, 32)     896
conv2d_1 (Conv2D)        (None, 828, 276, 64)    18496
max_pooling2d (MaxPooling2D (None, 414, 138, 64)    0
)

dropout (Dropout)        (None, 414, 138, 64)    0
flatten (Flatten)        (None, 3656448)        0
dropout_1 (Dropout)       (None, 3656448)        0
dense (Dense)            (None, 7)                25595143
classification_head_1 (Soft  (None, 7)
max)

=====
Total params: 25,614,542
Trainable params: 25,614,535
Non-trainable params: 7
```

Figura 43: Modelo propuesto por autokeras

En este caso, autokeras espera como parámetro de entrada las imágenes en color y sin preprocesar, por lo que no se le ha añadido ningún proceso para ello. Se han realizado experimentos diferentes valores de *epoch* utilizando toda la muestra y parte de ella:

minSample	N	brands	epoch	Train		Test	
				Acc	Loss	Acc	Loss
5	770	7	10	1	0,02	0,45	1,8
5	770	7	25	1	$1,51 \cdot 10^{-4}$	0,57	1,50
1	1470	59	10	1	0,01	0,24	4,86
1	1470	59	25	1	$2,61 \cdot 10^{-4}$	0,23	5,54

Figura 44: Tabla resumen de los experimentos con autokeras

Se puede observar que, aunque a la hora de entrenar el modelo propuesto por *autokeras* obtiene una eficacia del 100%, después al evaluarlo con los datos de *test* se ve como ésta disminuye casi a la mitad. Esto podría estar indicando que el modelo está **sobreentrenado**. Esto corrobora la necesidad de preprocesado y aumentación en el conjunto de imágenes para mejorar el desempeño de la red neuronal propuesta.

### 3.3.5.8. ImageNet: Modelo preentrenado

*ImageNet* es un proyecto que proporciona una gran base de datos de imágenes con sus correspondientes etiquetas que indican el contenido de éstas.

Durante el entrenamiento de una red neuronal se calculan pesos óptimos para hacer la clasificación. Una vez entrenada la red, se pueden guardar esos pesos y arquitectura del modelo para utilizarla en el futuro. *Keras* dispone diferentes modelos pre-entrenados que se pueden utilizar. Aunque estos modelos no se traten exclusivamente de imágenes de huellas, su gran contenido de datos hace que tenga una gran capacidad de aprendizaje.

Existen diferentes modelos entrenados con esta base de datos, entre ellos:

- **VGG16:** Desarrollado por la Universidad de Oxford, dispone de una arquitectura más sencilla que consta de 16 capas convolucionales y totalmente conectadas. Además, utiliza pequeños filtros convolucionales 3x3 y múltiples capas convolucionales apiladas antes de aplicar el max-pooling, lo que hace que la red sea más profunda. Tiene un mayor número de parámetros (alrededor de 138 millones), lo que puede hacer que sea más lenta de entrenar y más propensa al sobreajuste.

- **ResNet** (red residual): Desarrollada por *Microsoft Research*, introduce el concepto de conexiones residuales que permiten a la red aprender funciones residuales de forma más eficaz. Puede tener muchas más capas que VGG16 (por ejemplo, *ResNet-50*, *ResNet-101*, *ResNet-152*), pero tiene menos parámetros que VGG16 (*ResNet-50*, la que se utiliza en el experimento, tiene unos 25 millones de parámetros), lo que lo hace más eficiente y menos propensa al sobreajuste.

Se ha experimentado con el modelo *VGG16* con epoch = 10 y utilizando el tamaño de imagen por defecto (224,224,3), con y sin aumentación en el conjunto de *train* y especificando si se quiere desbloquear las primeras capas para entrenarlas o no.

Al entrenar los modelos con diferentes parámetros se ha observado que en los que se desbloquean las primeras capas utiliza más tiempo, esto se debe a que estamos compilando el modelo, mientras que en las otras pruebas se transfiere el conocimiento de vgg16 directamente.

En las pruebas realizadas se puede observar cómo al utilizar *VGG16* con transferencia se obtiene una eficacia en entrenamiento del 99% con poco *loss* (comparado con el modelo del proyecto), aunque de la evaluación con las imágenes del subconjunto de *test* la eficacia continúa siendo del 15%. En cambio, en las pruebas desbloqueando algunas capas y volviéndolo a entrenar, esta eficacia aumenta hasta el 26%.

Los resultados obtenidos se resumen en la siguiente tabla:

Modelo	tune	pre	Train			Test			% similitud marca		
			Acc	Val_acc	Loss	Val_loss	Top 1	Top 3	25%	50%	75%
VGG16	0	-	0,23	0,20	2,26	1,99	2,13	0,19	0,51	0,44	0
	A	0,99	0,11	0,02	5,97	5,91	0,15	0,53	0,15	0,15	0,15
4	-	0,31	0,35	7,86	1,84	1,88	0,26	0,60	0	0	0
	A	0,32	0,35	1,81	1,79	1,91	0,26	0,60	0,26	0	0

Figura 45: Tabla resumen de los experimentos con VGG16

### 3.3.6. Pruebas de validación cruzada

Aprovechando que se dispone de dos conjuntos de datos, se ha creado otro cuaderno que utiliza un modelo entrenado con el 100% de los datos de la primera base de datos (2d Footwear) y se realiza una prueba con las huellas de la otra base de datos (FID300), aunque esta última no dispone de la marca.

Para estos experimentos se aplica un análisis cualitativo del resultado, comprobando de forma visual si la huella de la predicción se parece a huellas del conjunto de datos de entrenamiento etiquetados con esa marca. Además, se han entrenado dos modelos, uno sin aumentación y epoch=10 y el otro con aumentación y epoch=30.

A continuación, una tabla resumen de los resultados obtenidos:

epoch	Pre	Train	
		Acc	Loss
10	GBC	0,99	0,03
	AGBC	0,30	7,89
30	GBC	0,98	0,08
	AGBC	0,47	6,43

Figura 46: Tabla resumen de las pruebas de validación cruzada

Para los experimentos primero se muestra tres ejemplos aleatorios de 3 marcas conocidas y relativamente fácil de reconocer en las imágenes de *test*: Nike, Adidas y Converse.

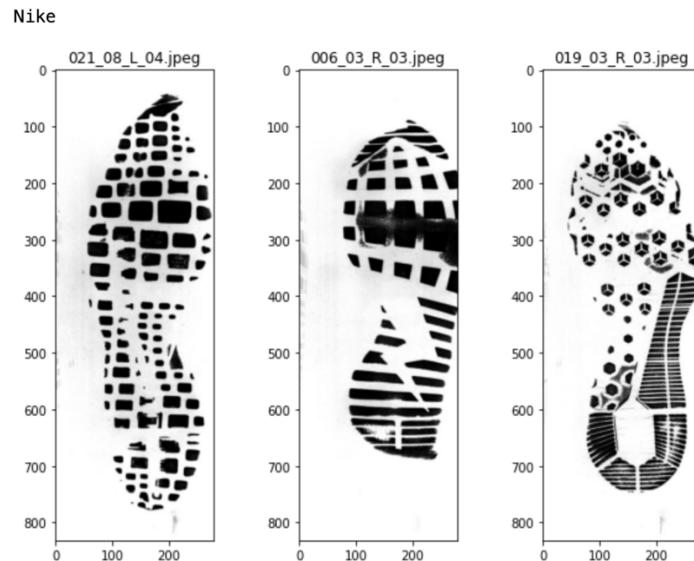


Figura 47: Ejemplo de imágenes de la marca Nike

Adidas

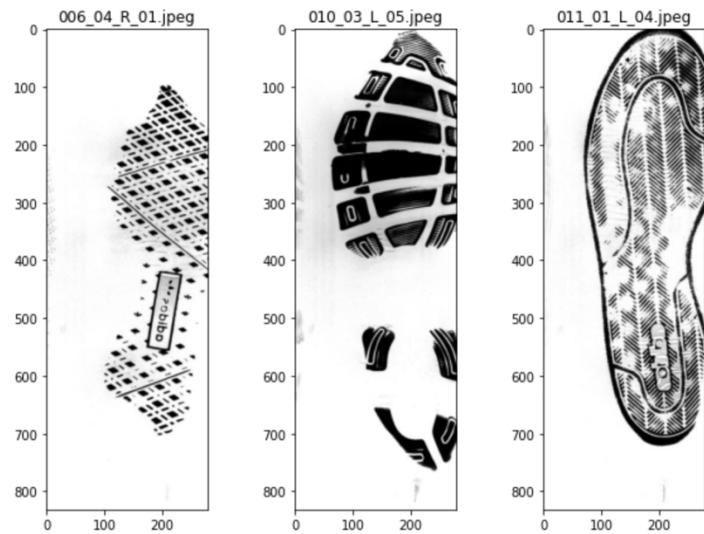


Figura 48: Ejemplo de imágenes de la marca Adidas

Converse

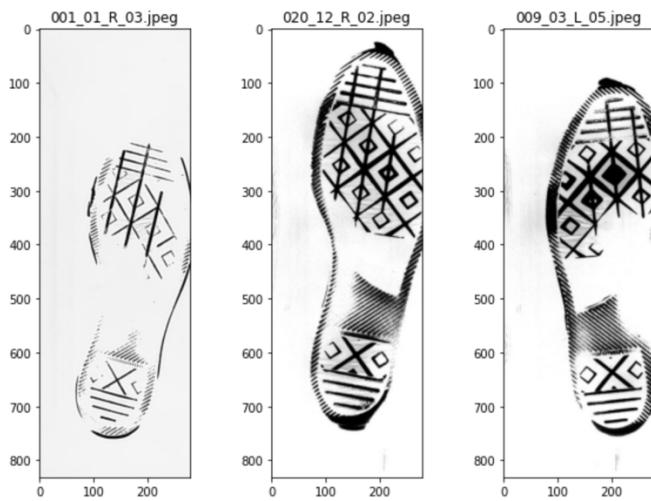
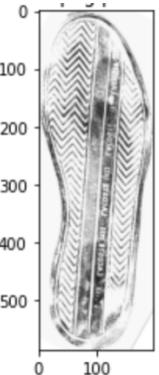


Figura 49: Ejemplo de imágenes de la marca Converse

Se han preseleccionado 5 imágenes de referencia de la base de datos utilizada para la validación, mostrando su predicción y el vector de probabilidades por marca. Los resultados se resumen en la siguiente tabla:

Brand	Imagen	Epoch	Aum	Pred	softmax
Nike		10	No	Nike	<p>[['Asics' 0.0] ['Skechers' 0.0] ['Sperry' 1.0300133379181193e-37] ['Adidas' 4.7645809998044803e-32] ['Nike' 1.0] ['Converse' 0.0] ['Saucony' 0.0]]</p>
			Si	Nike	<p>[['Asics' 4.582438486826212e-13] ['Skechers' 1.8367098846283625e-06] ['Sperry' 4.083015522904354e-11] ['Adidas' 1.3769531070906282e-09] ['Nike' 0.9999982118606567] ['Converse' 1.466114716204441e-23] ['Saucony' 1.3492091126166715e-34]]</p>
		30	No	Nike	<p>[['Asics' 0.0] ['Skechers' 0.0] ['Sperry' 0.0] ['Adidas' 1.664374664854697e-29] ['Nike' 1.0] ['Converse' 0.0] ['Saucony' 0.0]]</p>
			Si	Nike	<p>[['Asics' 4.607729128802696e-13] ['Skechers' 0.01462328433904785] ['Sperry' 1.1322342174935329e-07] ['Adidas' 9.737241634866223e-05] ['Nike' 0.9852792024612427] ['Converse' 3.108489021741434e-09] ['Saucony' 1.6071777533834598e-25]]</p>
		10	No	Nike	<p>[['Asics' 0.0] ['Skechers' 0.0] ['Sperry' 0.0] ['Adidas' 5.090725398653341e-19] ['Nike' 1.0] ['Converse' 0.0] ['Saucony' 0.0]]</p>
			Si	Nike	<p>[['Asics' 4.541421492337827e-13] ['Skechers' 1.8178720893047284e-06] ['Sperry' 4.0468624978862167e-11] ['Adidas' 1.3690182321113298e-09] ['Nike' 0.9999982118606567] ['Converse' 1.4427518116354156e-23] ['Saucony' 1.3267115971509974e-34]]</p>
Adidas		10	No	Nike	<p>[['Asics' 0.0] ['Skechers' 0.0] ['Sperry' 0.0] ['Adidas' 1.6361974069886503e-29] ['Nike' 1.0] ['Converse' 0.0] ['Saucony' 0.0]]</p>
			Si	Nike	<p>[['Asics' 4.607625045394137e-13] ['Skechers' 0.014679527841508389] ['Sperry' 1.1469976612943356e-07] ['Adidas' 9.733696788316593e-05] ['Nike' 0.9852230548858643] ['Converse' 3.10841263839734e-09] ['Saucony' 1.6099762374447313e-25]]</p>
		30	No	Nike	<p>[['Asics' 0.0] ['Skechers' 0.0] ['Sperry' 0.0] ['Adidas' 1.6361974069886503e-29] ['Nike' 1.0] ['Converse' 0.0] ['Saucony' 0.0]]</p>
			Si	Nike	<p>[['Asics' 4.607625045394137e-13] ['Skechers' 0.014679527841508389] ['Sperry' 1.1469976612943356e-07] ['Adidas' 9.733696788316593e-05] ['Nike' 0.9852230548858643] ['Converse' 3.10841263839734e-09] ['Saucony' 1.6099762374447313e-25]]</p>
Converse		10	No	Nike	<p>[['Asics' 0.0] ['Skechers' 0.0] ['Sperry' 0.0] ['Adidas' 5.111020274982329e-19] ['Nike' 1.0] ['Converse' 0.0] ['Saucony' 0.0]]</p>
			Si	Nike	<p>[['Asics' 4.5458238952592045e-13] ['Skechers' 1.8246668105348363e-06] ['Sperry' 4.2311130293848365e-11] ['Adidas' 1.3678906896075205e-09] ['Nike' 0.9999982118606567] ['Converse' 1.460042380500412e-23] ['Saucony' 1.3471519975027325e-34]]</p>
		30	No	Nike	<p>[['Asics' 0.0] ['Skechers' 0.0] ['Sperry' 0.0] ['Adidas' 1.669296168178991e-29] ['Nike' 1.0] ['Converse' 0.0] ['Saucony' 0.0]]</p>

			Si	Nike	[['Asics' 4.57749289861642e-13] [['Skechers' 0.01469817757605063] [['Sperry' 1.1435368207912688e-07] [['Adidas' 9.709243022371083e-05] [['Nike' <b>0.9852045774459839</b> ] [['Converse' 3.0966436082024984e-09] [['Saucony' 1.5903218912506662e-25]]]
Lacoste (None)		10	No	Nike	[['Asics' 0.0] [['Skechers' 0.0] [['Sperry' 0.0] [['Adidas' 5.197114784100497e-19] [[Nike' <b>1.0</b> ] [['Converse' 0.0] [['Saucony' 0.0]]]]]
			Si	Nike	[['Asics' 4.60985958607163e-13] [['Skechers' 1.8328428268432617e-06] [['Sperry' 4.0960262959188753e-11] [['Adidas' 1.3850237623458383e-09] [['Nike' 0.9999982118606567] [['Converse' 1.478335276031582e-23] [['Saucony' 1.3723603820206203e-34]]]
		30	No	Nike	[['Asics' 0.0] [['Skechers' 0.0] [['Sperry' 0.0] [['Adidas' 1.6690542232297273e-29] [[Nike' <b>1.0</b> ] [['Converse' 0.0] [['Saucony' 0.0]]]]]
			si	Nike	[['Asics' 4.648070123236536e-13] [['Skechers' 0.01467146910727024] [['Sperry' 1.1460061699608559e-07] [['Adidas' 9.765329741640016e-05] [['Nike' <b>0.9852307438850403</b> ] [['Converse' 3.1313409643018986e-09] [['Saucony' 1.6354984761784763e-25]]]

Los resultados no han sido satisfactorios, ya que siempre clasifica la marca “Nike” aunque se pueda observar en la imagen que se trata de otra marca. Esto nos da indicios de que **la red neuronal creada solo funciona en el escenario ideal** de evaluar imágenes extraídas y procesadas de la misma manera.

Aunque añadiendo aumentación no se obtiene diferente resultado, se puede observar que da una oportunidad al resto de marcas y disminuye el porcentaje de similitud de Nike.

Para poder utilizar el modelo con imágenes en otro formato sería necesario disponer de más datos, por ejemplo, añadir el **modelo**, ya que, dependiendo de este, aunque se trate de la misma marca, el dibujo de la suela puede variar. Además, de mayor número de muestras y mejorar el preprocesado de las imágenes.

Teniendo en cuenta que uno de los objetivos del proyecto era prescindir de cámaras especializadas y utilizar un dispositivo móvil para tomar las imágenes, el modelo actualmente no sería adecuado, ya que no tiene en cuenta variaciones.

Estas pruebas cruzadas se encuentran en el cuaderno *TFM\_LauraRivera\_objetivo1\_cross\_validation.ipynb*.

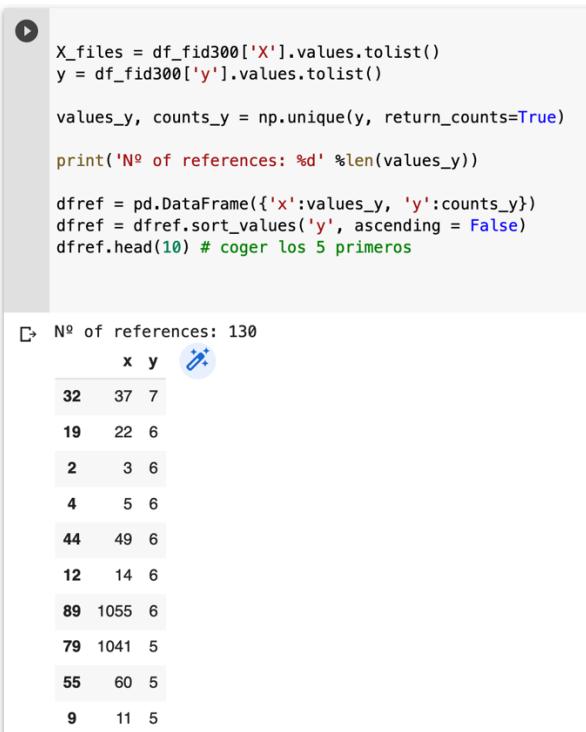
### 3.4. Objetivo 2: Búsqueda de huellas similares

El objetivo de este segundo apartado del proyecto es crear y experimentar con una red neuronal siamesa capaz de devolver imágenes similares a la que se envía por parámetro. La idea es simular la situación de recoger una huella en una escena del crimen y compararla con las que existen en la base de datos de sospechosos.

#### 3.4.1. Análisis de los datos disponibles

Para el proyecto se ha decidido utilizar las imágenes de las carpetas *references* y *cropped* y mantenerlas en dos conjuntos de datos separados. Se dispone de 1175 muestras en la primera carpeta y 300 en la segunda.

Por último, el fichero *label\_table.csv* contiene la correspondencia entre imágenes de la carpeta *cropped* (de la escena del crimen) y *references* (tomadas después al calzado de diferentes individuos). También se ha analizado los datos y se ha podido ver de las 300 líneas, hay 130 valores diferentes de y (imagen de referencia), porque algunas imágenes de referencia aparecen varias veces en los datos.



```
X_files = df_fid300['X'].values.tolist()
y = df_fid300['y'].values.tolist()

values_y, counts_y = np.unique(y, return_counts=True)

print('Nº of references: %d' %len(values_y))

dfref = pd.DataFrame({'x':values_y, 'y':counts_y})
dfref = dfref.sort_values('y', ascending = False)
dfref.head(10) # coger los 5 primeros
```

⇒ Nº of references: 130

x	y
32	37
19	22
2	3
4	5
44	49
12	14
89	1055
79	1041
55	60
9	11

Figura 50: Código para extraer las imágenes de referencia que más aparecen en la tabla.

### 3.4.2. Solución propuesta

La solución propuesta se encuentra en el cuaderno *TFM\_LauraRivera\_objetivo2\_similares.ipynb* y se trata de una reproducción del código de Luciano Naveiro en Medium [40] donde se entrena una red siamesa para después utilizar su espacio latente para buscar otras imágenes similares.

Una red neuronal siamesa, simplificando, está formada de tres partes en su arquitectura de capas:

- **Codificador:** Las primeras capas del modelo codificar la imagen en un conjunto de características.
- **Espacio latente:** Consiste en una capa totalmente conectada que guarda en un vector los valores de esas características extraídas en el codificador.
- **Descodificador:** Las últimas capas vuelven a reconstruir la imagen a partir de las características del espacio latente.

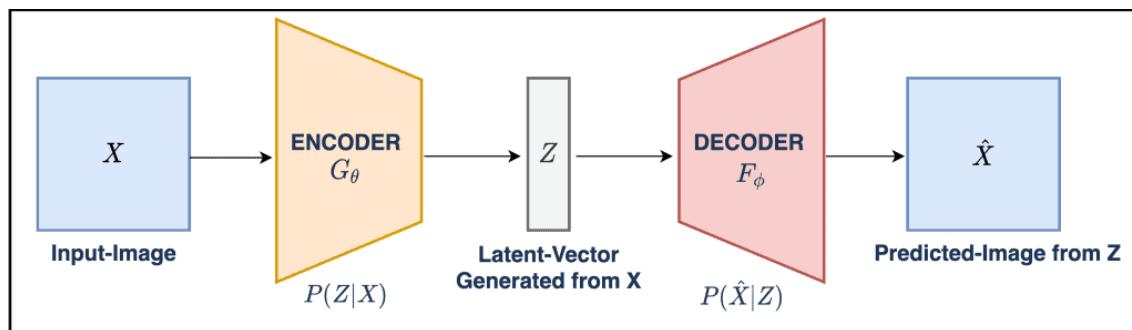


Figura 51: Representación de la red siamesa (Source: learnopencv.com)

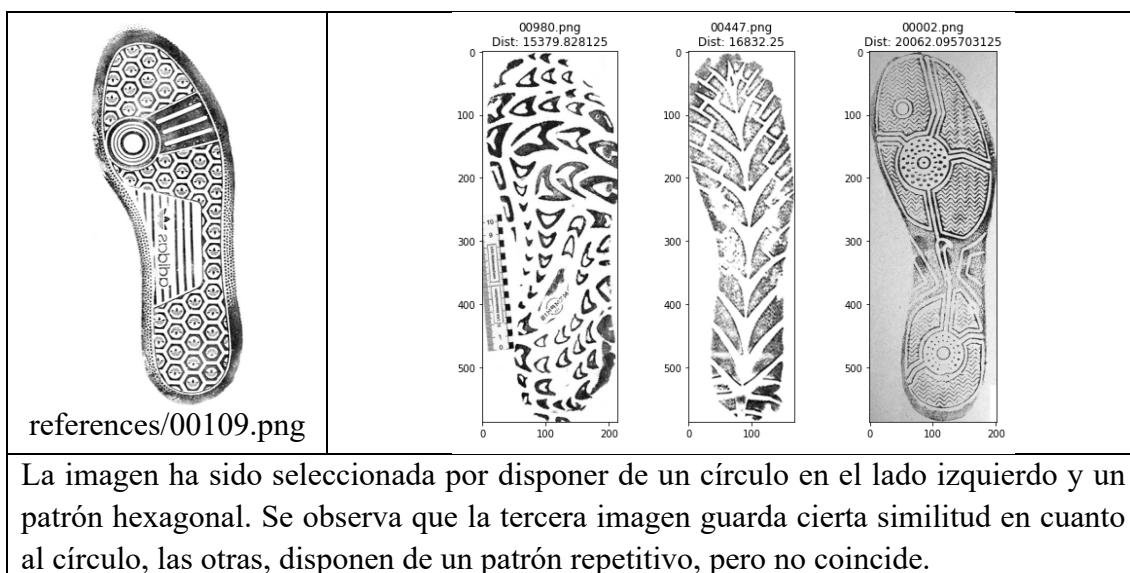
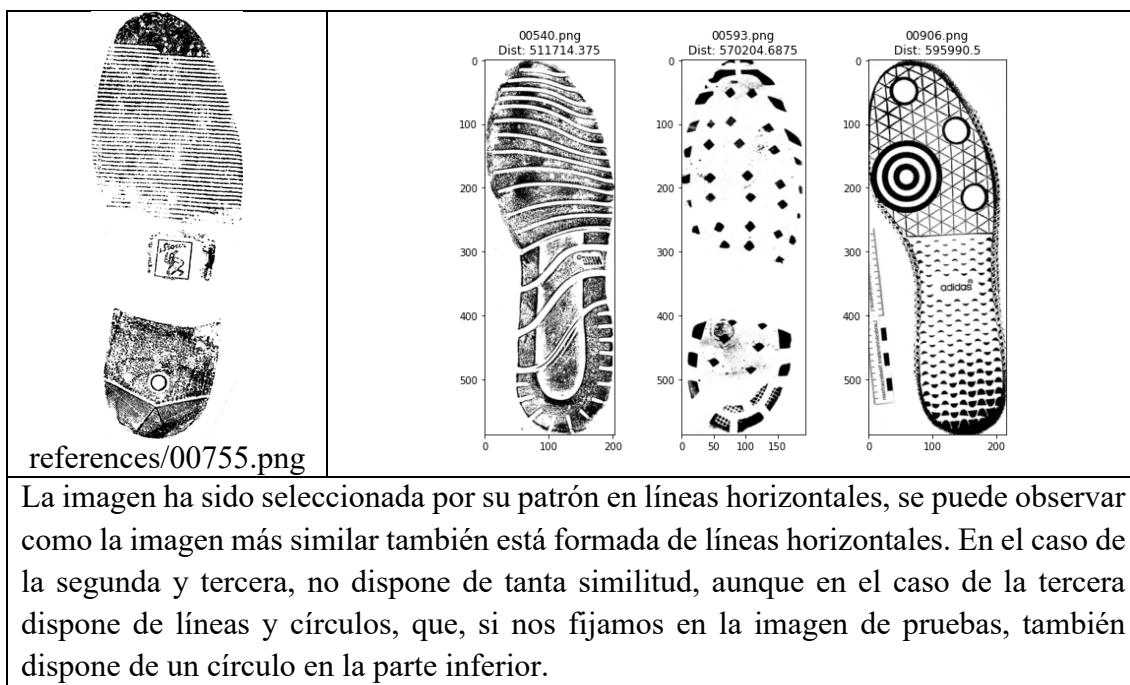
En el cuaderno se ha creado y entrenado la red siamesa guardando el modelo resultante en un fichero. Después, por cada una de las imágenes se realiza una predicción (cada una de ellas es codificada a un vector de características y posteriormente descodificada para su reconstrucción, esto permite que se guarden las características de todas en ese espacio latente y guardarla en una matriz).

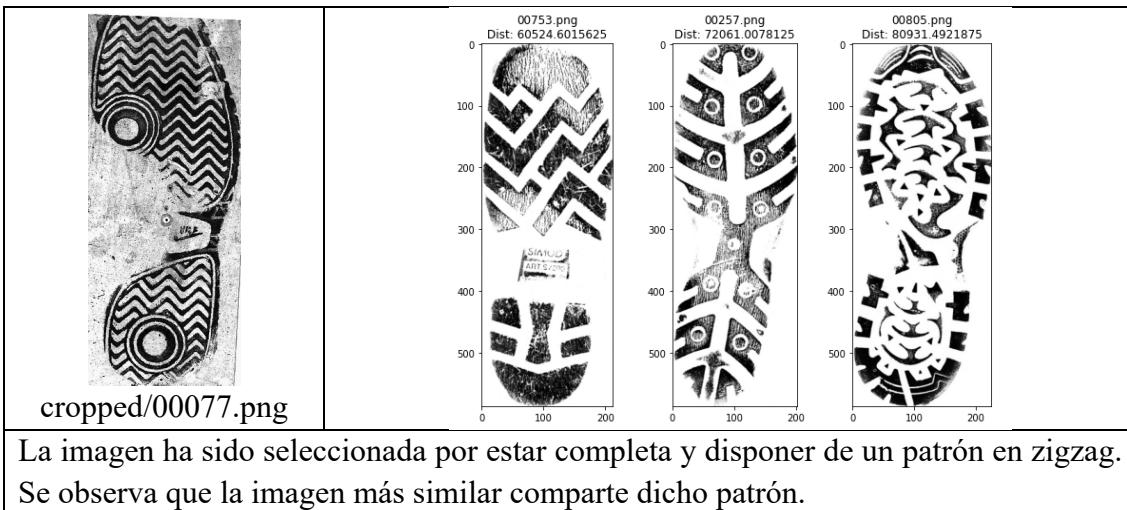
Por último, se ha creado una función *getTop3Similar* que, dada una imagen, codifica sus características y, después, calcula la distancia euclídea respecto la matriz de valores de las características de las imágenes de entrenamiento. Finalmente, devuelve las tres imágenes más similares (con menor distancia) y las menos similares (con mayor distancia).

### 3.4.3. Experimentos

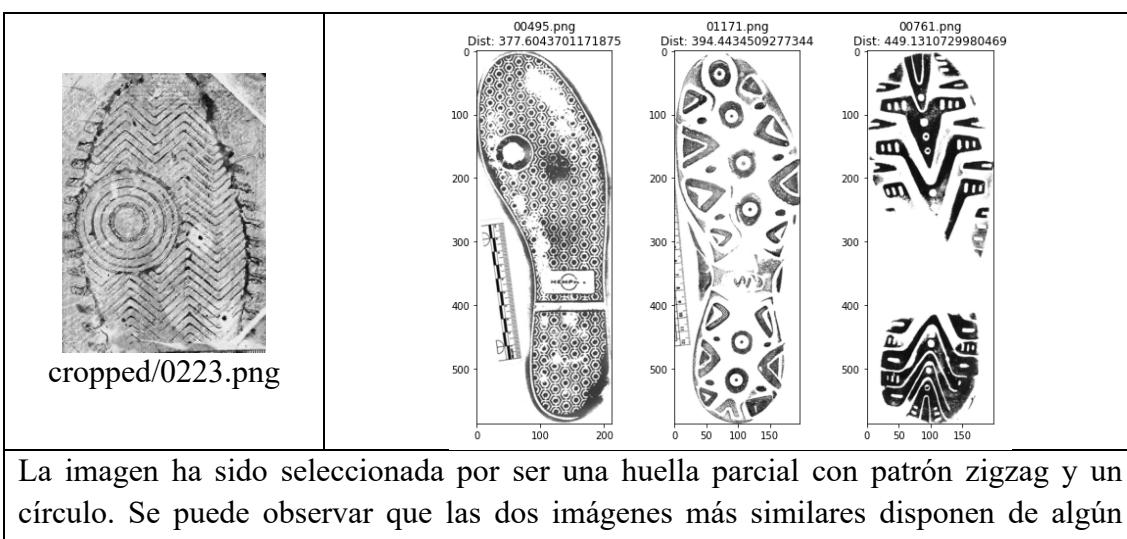
Se han realizado dos tipos de experimentos: con huellas enteras del subconjunto de referencia y con huellas parciales o poco nítidas del subconjunto de huellas en escenas reales y recortadas (*cropped*).

Para ello, se han preseleccionado imágenes con alguna característica en concreto. A continuación, algunos de los experimentos realizados mostrando la imagen de pruebas, las tres más similares y un comentario:





La imagen ha sido seleccionada por estar completa y disponer de un patrón en zigzag. Se observa que la imagen más similar comparte dicho patrón.



La imagen ha sido seleccionada por ser una huella parcial con patrón zigzag y un círculo. Se puede observar que las dos imágenes más similares disponen de algún círculo y un patrón repetitivo similar al zigzag.

En general, en las pruebas realizadas se intuye cierta similitud en cuanto a patrones, pero, por ejemplo, si la imagen de pruebas dispone de un círculo en el lado derecho, las imágenes similares probablemente dispongan de algún círculo, pero no necesariamente en la misma ubicación.

En cuanto a las imágenes menos similares, más o menos siempre se obtienen las mismas, puede que sea debido a que, además, son las más oscuras de las disponibles en el conjunto de entrenamiento.

## 4. Conclusiones generales

Las redes neuronales convolucionales son una técnica muy utilizada para la clasificación de imágenes. En el problema propuesto, se afronta la dificultad de disponer de una muestra limitada y un número elevado de clases.

Esta peculiaridad hace que, además, haya sido necesario reducir aún más el conjunto de datos para utilizar solo las muestras de aquellas marcas con al menos 5 imágenes.

Respecto a las **transformaciones** en las imágenes para trabajar con ellas, se ha observado que la red neuronal consigue mejores resultados si las imágenes se transforman, previamente, a blanco y negro y, además, se eliminan los espacios en blanco de alrededor.

En general, también se ha observado que el uso de **aumentación** disminuye la eficacia y obtiene valores superiores en el *loss*, pero aumenta el porcentaje de similitud, por lo que las muestras donde el modelo acierta la marca lo hacen con un porcentaje de seguridad mayor.

También, en las pruebas aumentando las **iteraciones (epoch)**, se ha observado que solo en el caso de utilizar aumento se observa alguna mejora, de lo contrario, como se dispone de pocos datos no interfiere excesivamente en la eficacia.

En cuanto al **mínimo de muestras por marca**, se confirma que, con mayor número de clases, se obtienen peores resultados, aunque también aumente el total de imágenes del conjunto de datos. En cambio, al focalizar el modelo en pocas clases, clasifica mejor.

Por último, se ha confirmado que el **tamaño de la imagen** solo obtiene mejores resultados en el caso de utilizar aumento durante el entrenamiento, de lo contrario, solo afecta al **tiempo** de ejecución.

Para una mejor comprensión de las capas que componen una red neuronal convolucional se ha experimentado quitando alguna de ellas, comprobando que la capa **maxPooling** disminuye mucho el tiempo de ejecución sin afectar a la eficacia, por lo que se recomienda su uso. Algo parecido ocurre al añadir **capas convolucionales adicionales**, que reducen el total de parámetros y el tiempo de ejecución.

Además, se observa que la capa **Dropout** contribuye a la eficiencia, incluso en los experimentos que utilizan aumentación. También es importante el valor que se le asigne, ya que si es muy pequeño o demasiado grande también disminuye la eficacia.

Una curiosidad que se ha observado es que el uso de una capa **Dense** antes de este Dropout no afecta al resultado, por lo que se podría prescindir de ésta.

Por último, la importancia de una capa conectada con activación **Softmax** para obtener la probabilidad de que una imagen sea de cada una de las marcas, en lugar de una única solución con la marca más probable, porque durante el proyecto se ha podido observar que una marca no dispone de un estilo único de huella y que algunas marcas comparten rasgos, además esto permite ver el porcentaje de similitud de marca, aunque esta no sea la más probable.

Después de todos los experimentos, se ha podido comprobar que con la configuración epoch = 10 y transformando las imágenes a blanco y negro sin espacios en blanco se obtiene una eficacia del 82%, que respecto al obtenido con SVM (98%) se puede considerar bastante satisfactorio.

Posteriormente se ha realizado la **validación cruzada**, para ello, se ha entrenado el modelo con el 100% de las imágenes y obtenido una eficacia de 98% sin aumentación y un 47% con aumentación, después se utilizan imágenes seleccionadas de otra base de datos de las que se puede intuir la marca para probar mediante observación su correcto funcionamiento. En este caso, aunque la eficacia es elevada, el modelo siempre **predice erróneamente** que todas las imágenes son de la marca Nike. Esto reafirma la hipótesis de que el modelo solo funciona en imágenes en el formato de la primera base de datos. Para solucionarlo haría falta más volumen de datos, tanto en número de imágenes como en información, por ejemplo, clasificar según marca y modelo, ya que se ha observado que zapatos de la misma marca tienen diferentes huellas dependiendo del modelo.

Resumiendo, esta red neuronal solo podría considerarse una buena solución si todas las imágenes se toman de la misma manera que en la primera base de datos, pero para el objetivo inicial que era prescindir de las cámaras actuales y simplificar el proceso de toma de huellas, **no sería una buena solución**. Se debería trabajar en disponer de más volumen

de datos, además de explorar más opciones para mitigar los problemas asociados esa limitación, y completar la información de marca con el modelo porque se ha observado que no existe un patrón único por marca.

Para finalizar, y con un objetivo totalmente académica, se ha implementado una **red neuronal siamesa** para buscar imágenes similares simulando una situación real de recoger una huella en una escena del crimen y compararla con las que existen en la base de datos de sospechosos.

Esto se consigue entrenando esta red formada por un codificador, que transforma las imágenes en un vector de característica, y un descodificador, que utiliza esas características para predecir la misma imagen. Ese vector de características se almacena en una capa denominada **espacio latente**, que posteriormente se utiliza para medir la distancia euclídea de las características de la imagen de entrada respecto a las de la matriz. Las imágenes con menor distancia se consideran las más similares.

En los experimentos realizados en este apartado, se ha observado que, en general, las imágenes tienen alguna similitud, sobre todo en los patrones de dibujo (por ejemplo, zigzag) y contener formas concretas como círculos, pero no en su totalidad.

## 5. Comentarios

Este proyecto ha sido una excelente oportunidad para poner en práctica lo aprendido durante el máster y, sobre todo, para comprender en profundidad el concepto de red neuronal convolucional y siamesa. La búsqueda de conjuntos de datos ha sido un reto, también el disponer de poco volumen de éstos, aunque también ha ofrecido la oportunidad de experimentar con este tipo de problemas e intentar solucionarlos.

A continuación, se nombran diferentes consideraciones o imprevistos que se han localizado durante el proceso del proyecto.

### 5.1. Selección de los datos

Se encontraron algunas limitaciones durante la selección del conjunto porque **existen pocas bases de datos de uso libre**. Esto hizo que finalmente se decidiera utilizar dos bases de datos para los dos objetivos principales y se reestructurara las tareas en la planificación inicial.

### 5.2. Ficheros necesarios para la ejecución

Para la ejecución del código Python en Google Colab o Jupyter Notebook es necesario disponer de las imágenes siguiendo la siguiente **estructura**:

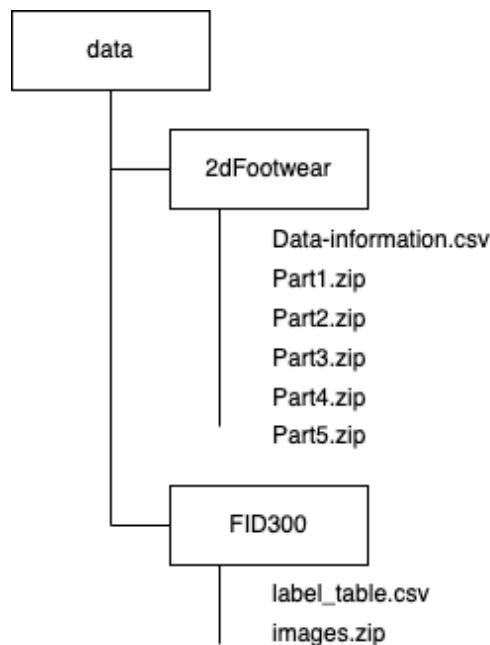


Figura 52: Esquema carpetas para las imágenes del proyecto

Los cinco cuadernos que forman el proyecto se encuentran en el github:  
[https://github.com/laurivsan/TFM\\_AI\\_2023/tree/main/code](https://github.com/laurivsan/TFM_AI_2023/tree/main/code)

Las imágenes se encuentran en sus respectivas fuentes de datos especificadas tanto en la bibliografía como en el *Readme* del repositorio.

2D Footwear	<a href="https://iastate.figshare.com/articles/figure/2D_Footwear_outsole_impressions/11624073/1">https://iastate.figshare.com/articles/figure/2D_Footwear_outsole_impressions/11624073/1</a>
FD-300	<a href="https://fid.dmi.unibas.ch/">https://fid.dmi.unibas.ch/</a>

### 5.3. Formato de las imágenes

Las imágenes de la base de datos 2d Footwear se encuentran en formato .tiff. Ha sido necesario convertirlas a .jpeg ya que la librería **pytorch**, según su documentación [34], solo soporta .jpeg y .png. Este proceso hace que el proceso inicial de cargar las imágenes tarde unos minutos extra.

```
def get_images_to_jpeg(imgPath):
    dir_list = os.listdir("./"+imgPath)
    result = []
    for f in dir_list:
        im = Image.open("./"+imgPath+"/"+f)
        im.save("./"+imgPath+"/"+f[0:-4]+'.jpeg')
        result.append(f[0:-4] + '.jpeg')
        os.remove("./"+imgPath+"/"+f)

    print('Nº files:', len(result))
    return result
```

Figura 53: Código de la función que lee y convierte los ficheros .tiff a .jpeg.

### 5.4. Red neuronal siamesa

La segunda parte del proyecto se ha realizado para entender mejor cómo funcionan las redes neuronales siamesas, ya que el concepto apareció durante el máster, pero no se había creado ninguna de cero, por lo que esta parte del proyecto solo muestra una primera versión de la solución y sería mejorable.

## 5.5. Google Colab

Inicialmente se pretendía utilizar *Google Colab* y aprovechas la *GPU* de este para agilizar algunos procesos, pero durante los primeros experimentos aparecían muchos **problemas por falta de memoria en su versión gratuita**. Finalmente se decidió utilizar *Jupyter Notebook* directamente en la máquina, con los inconvenientes de horas de uso para entrenar los modelos. Por esta razón, los modelos más costosos en tiempo se han guardado en fichero.

# 6. Trabajo futuro

## 6.1. Preprocesado de las imágenes

Para el proyecto se ha utilizado aumentación y binarización (conversión a blanco y negro eliminando los espacios en blanco) como preprocesado en las imágenes, pero existen otros tipos de procesado de imágenes que se podrían aplicar para experimentar el resultado del modelo.

## 6.2. Red neuronal de similitud

Para la implementación de esa primera versión de red neuronal siamesa para el caso de búsqueda de imágenes similares se podría experimentar con otras mejoras como, por ejemplo, diferentes funciones de distancia.

## 6.3. One-shot Learning

En la base de datos de muestras de calzado con información de marca existen muchas marcas con una única muestra que se han descartado para el proyecto, se podría investigar y utilizar *On-shot Learning* para poder crear predicciones de aquellas marcas con una única muestra.

## 6.4. Pruebas con imágenes realizadas con el teléfono

Uno de los objetivos de crear este proyecto es que, con más trabajo y experimentos, llegar a utilizar imágenes realizadas con un dispositivo móvil, sin necesidad de aparatoología científica. Para ello, sería necesario crear una base de datos con más imágenes para seguir entrenando el modelo. En este caso, también añadiría información del medio de la huella (arena, barro, papel, asfalto) e información sobre el modelo para mejorar la predicción, ya que se ha comprobado que una misma marca no utiliza siempre suelas similares.

## 6.5. Entrenar el modelo con otro tipo de datos

Sería interesante utilizar el modelo propuesto con otro tipo de imágenes, por ejemplo, huellas descalzas con información médica relevante de los individuos, como muestra el artículo *Prediction of Gender and Stature by Footprint Analysis (Rehman RS, Butt M, Ambreen A, Asif R, Khan T, Samina Mushtaq)* [42] donde realizan una predicción de altura y género según la huella descalza.

También sería interesante utilizar imágenes de otro ámbito, imágenes de objetos 3D, y ver su rendimiento y si el modelo es apto en esos casos.

# Bibliografía

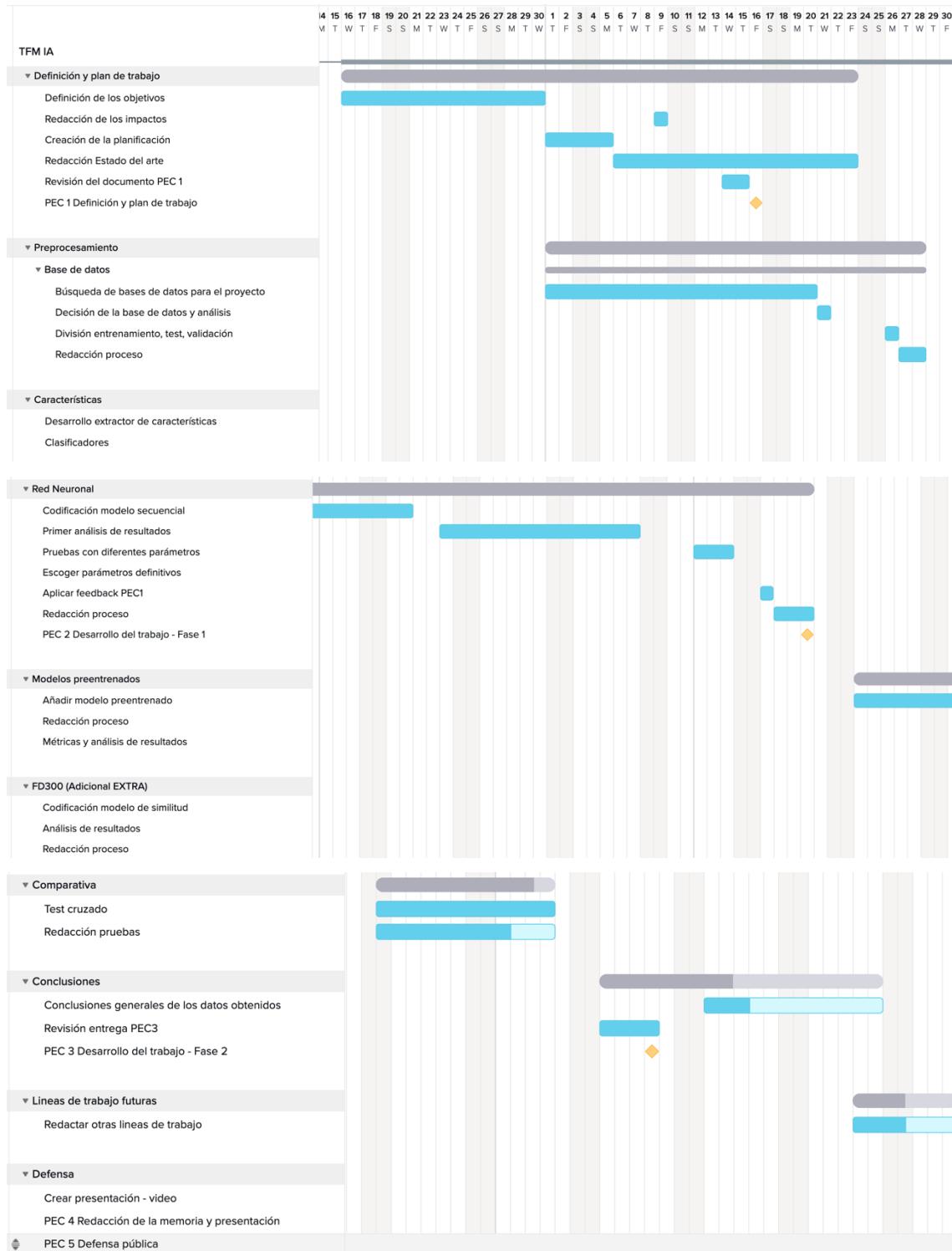
- [1] R. Singel, «Wired,» 20 11 2008. [En línea]. Available: <https://www.wired.com/2008/11/professor-sees/>.
- [2] Interpol, «Interpol - Huellas dactilares,» [En línea]. Available: <https://www.interpol.int/es/Como-trabajamos/Policia-cientifica/Huellas-dactilares#:~:text=Existen%20tres%20patrones%20principales%20de,d enominados%20arcos%2C%20curvas%20y%20espirales..> [Último acceso: 2022 12 10].
- [3] Forenscope, «Pista de Neumáticos & Huella,» [En línea]. Available: <https://forenscope.com/es/blog-detail/pista-de-neum%C3%A1ticos-huella>. [Último acceso: 2023 01 28].
- [4] H. O. Lopez, «Investigación de Huellas de Neumático,» [En línea]. Available: <file:///Users/laura/Downloads/Dialnet-InvestigacionDeHuellasDeNeumatico-4761231.pdf>. [Último acceso: 2023 01 30].
- [5] M. Nirenberg, «Gait, Footprints, and Footwear: How Forensic Podiatry Can Identify Criminals,» Enero 2016. [En línea]. Available: <https://www.policechiefmagazine.org/gait-footprints-and-footwear-how-forensic-podiatry-can-identify-criminals/>.
- [6] S. C. J. A. Pablo Martinez, «Dialnet,» [En línea]. Available: <https://dialnet.unirioja.es/servlet/articulo?codigo=7345717#:~:text=La%20Podolog%C3%A3a%20Forense%20o%20Medicina,la%20escena%20de%20un%20crimen..> [Último acceso: 2023 01 30].
- [7] F. & Foreman, «bcluae,» [En línea]. Available: <https://www.bcluae.com/solemate-footwear-identification>.
- [8] H. is, «Hobbit PRIDE,» [En línea]. Available: <https://hobbit-is.nl/forensic-intelligence/pride/?lang=en>.
- [9] M. H. O. R. M. S. R. & M. M. H. Md Asadujjaman, «Stature estimation from footprint measurements in Bangladeshi adults,» *Forensic Sciences Research* 7:2, 124-131, DOI: 10.1080/20961790.2020.1776469, 2022.
- [10] S. D. Kapil Kumar Nagwanchi, «Biometric Authentication using Human Footprint,» *International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868*, vol. 3, nº 7, pp. DOI: 10.5120/ijais12-450568, 2012.
- [11] U. O. d. Catalunya, «Asignatura TFM IA en el campus,» [En línea]. Available: [campus.uoc.edu](http://campus.uoc.edu). [Último acceso: 09 12 2022].
- [12] TeamGantt, «TeamGantt: Get a top-rated gantt chart for free, forever.,» [En línea]. Available: [www.teamgantt.com](http://www.teamgantt.com). [Último acceso: 09 12 2022].
- [13] R. A. Española, «RAE Definición de huella,» [En línea]. Available: <https://dle.rae.es/huella>. [Último acceso: 2022 12 10].
- [14] wikicrim, «CFEC,» [En línea]. Available: <https://www.estudiocriminal.eu/blog/concepto-y-tipos-de-huella/>. [Último acceso: 2012 12 10].

- [15] M. Nirenberg, «Gait, Footprints, and Footwear: How Forensic Podiatry Can Identify Criminals,» 2016. [En línea]. Available: <https://www.policechiefmagazine.org/gait-footprints-and-footwear-how-forensic-podiatry-can-identify-criminals/>.
- [16] M. D. o. P. Safety, «Evidence submission,» [En línea]. Available: <https://dps.mn.gov/divisions/bca/bca-divisions/forensic-science/Pages/evidence-submission.aspx>. [Último acceso: 2022 12 07].
- [17] I. U. L. Pueblo, «YouTube: Criminalistica de campo: Levantamiento de huella pie calzado,» 23 10 2013. [En línea]. Available: <https://www.youtube.com/watch?v=pB7jUtKDUtw>. [Último acceso: 12 12 2022].
- [18] @srtaperito, «Twitter,» [En línea]. Available: <https://twitter.com/srtaperito/status/910522790624800768>.
- [19] Artec3D, «Artec3d,» [En línea]. Available: <https://www.artec3d.com/es/portable-3d-scanners/artec-eva>. [Último acceso: 12 12 2022].
- [20] UOC, «Introducción a la inteligencia artificial,» de *Inteligencia artificial avanzada*, PID\_00250574.
- [21] bclue, «SOLEMATE FPX-Footwar Identification,» 2021. [En línea]. Available: <https://www.bcluae.com/solemate-footwear-identification>.
- [22] «DigTrace - Manuals,» 2016. [En línea]. Available: <https://www.digtrace.co.uk/manuals>.
- [23] H. I. Solutions, «Showprint matcher,» 2019. [En línea]. Available: <https://hobbit-is.nl/forensic-intelligence/pride/?lang=en>.
- [24] assets.publishing.service.gov.uk, «Forensic Information Databases Service (FINDS),» [En línea]. Available: [https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment\\_data/file/955328/FINDS-S-023\\_-\\_Issue\\_2\\_-\\_Process\\_for\\_Release\\_from\\_the\\_Forensic\\_Information\\_Databases\\_for\\_Research\\_Purposes.pdf](https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/955328/FINDS-S-023_-_Issue_2_-_Process_for_Release_from_the_Forensic_Information_Databases_for_Research_Purposes.pdf).
- [25] Kaggle, «Kaggle,» [En línea]. Available: <https://www.kaggle.com/competitions/gradient-shoes-photo-types>.
- [26] A. Kortylewski, «FID - Footwear Impression Database,» 2016. [En línea]. Available: <https://fid.dmi.unibas.ch/>.
- [27] T. A. T. V. Adam Kortylewski, «Unsupervised Footwear Impression Analysis and Retrieval from Crime Scene Data,» [En línea]. Available: [https://fid.dmi.unibas.ch/FID\\_ACCV14.pdf](https://fid.dmi.unibas.ch/FID_ACCV14.pdf).
- [28] X. D. G. S. X. Z. C. C. Yanjun Wu, «Crime Scene Shoepoint Image Retrieval: A Review,» 11 07 2022. [En línea]. Available: <https://www.mdpi.com/2079-9292/11/16/2487>.
- [29] S. P. a. A. Carriquiry, «A database of two-dimensional images of footwear outsole impressions,» ELSEVIER, nº 105508, 2020.
- [30] A. C. Soyoung Park, «2D Footwear outsole impressions,» 01 07 2020. [En línea]. Available: [https://iastate.figshare.com/articles/figure/2D\\_Footwear\\_outsole\\_impressions/11624073/2?file=21217842](https://iastate.figshare.com/articles/figure/2D_Footwear_outsole_impressions/11624073/2?file=21217842).

- [31] O. G. Yalçın, «Towards Data Science,» 23 11 2020. [En línea]. Available: <https://towardsdatascience.com/4-reasons-why-you-should-use-google-colab-for-your-next-project-b0c4aaad39ed>.
- [32] M. C. L. M. y. J. N. Richetelli, «Classification of footwear outsole patterns using Fourier transform and local interest points,» June 2017. [En línea]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0379073817300841>.
- [33] python, «pickle - Python object serialization,» [En línea]. Available: <https://docs.python.org/3/library/pickle.html>.
- [34] S. O. A. B. A. DRAPER, «Research Gate,» 17 Enero 2011. [En línea]. Available: [https://www.researchgate.net/publication/48190777\\_Introduction\\_to\\_the\\_Bag\\_of\\_Features\\_Paradigm\\_for\\_Image\\_Classificationand\\_Retrieval](https://www.researchgate.net/publication/48190777_Introduction_to_the_Bag_of_Features_Paradigm_for_Image_Classificationand_Retrieval).
- [35] M. Hearst, S. Dumais, E. Osuna, J. Platt y B. Scholkopf, «IEEE,» Julio 1998. [En línea]. Available: <https://ieeexplore.ieee.org/document/708428>.
- [36] Y. Yuan, Y. Pang y X. Li, «IEEE,» Enero 2010. [En línea]. Available: <https://ieeexplore.ieee.org/abstract/document/4914855>.
- [37] N. Viswanathan, «Stanford,» 2017. [En línea]. Available: <http://cs231n.stanford.edu/reports/2017/pdfs/406.pdf>.
- [38] M. Alom, «ResearchGate,» June 2021. [En línea]. Available: [https://www.researchgate.net/publication/352497171\\_Adam\\_Optimization\\_Algorithm](https://www.researchgate.net/publication/352497171_Adam_Optimization_Algorithm).
- [39] Szeliski, «Computer Vision: Algorithms and Applications 2nd Edition,» de 5.3 *Deep neural networks*, p. 275.
- [40] S. S. a. R. H. Simukayi Mutasa, «<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8150901/>,» [En línea].
- [41] J. y. S. P. C. Chica, «Universidad de Alicante,» [En línea]. Available: <http://rua.ua.es/dspace/handle/10045/104718>.
- [42] «Autokeras,» [En línea]. Available: <https://autokeras.com/>.
- [43] L. Naveiro, «medium: Querying Similar Images with TensorFlow,» 19 May 2022. [En línea]. Available: <https://medium.com/@luchonaveiro/querying-similar-images-with-tensorflow-59e3a7aad40e>.
- [44] PyTorch, «PyTorch READ\_IMAGE,» [En línea]. Available: [https://pytorch.org/vision/main/generated/torchvision.io.read\\_image.html](https://pytorch.org/vision/main/generated/torchvision.io.read_image.html).
- [45] B. M. A. A. A. R. K. T. S. M. Rehman RS, «medwinpublishers,» 01 November 2021. [En línea]. Available: <https://medwinpublishers.com/IJFSC/prediction-of-gender-and-stature-by-footprint-analysis-in-punjab-population-pakistan.pdf>.
- [46] M. Konkiewicz, «Towards Data Science,» 19 12 2020. [En línea]. Available: <https://towardsdatascience.com/how-to-successfully-add-large-data-sets-to-google-drive-130beb320f1a>.

# Anexos

## a.1 Diagrama de Gantt definitivo



a.2. Extracto del documento: Shoeprint and tire track collection guide

## Shoeprint and Tire Track Collection Guide

### **Shoeprint Impression Collection**

Proper collection and photographs of footwear and tire evidence is essential to capture the detail observed in the field. Improper collection and/or photographic techniques have a direct effect on the analysis in the laboratory and can cause limitations in the forensic comparison. The laboratory suggests the following guideline for collection and photography of footwear and tire evidence:

#### **Photography**

Here are some tips to help you take the best impression photographs possible.

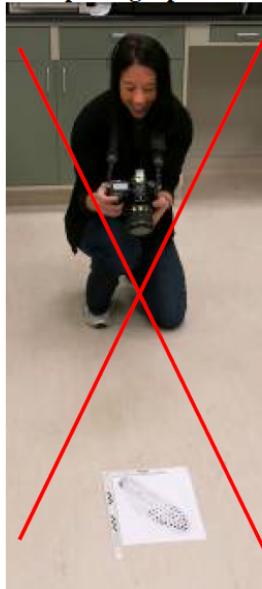
1. Take photographs carefully.
2. Take the photograph from directly over the impression using a tripod.
3. Use a flat ruler in the plane of the impression; 'L' shaped rulers are best.
4. Capture the entire impression, filling the frame.
5. Use lighting from different angles.
6. Take multiple photographs
7. Submit a CD with original images and paperwork describing the images.

Note: It may be helpful to put an item number or other designation in the photo.

#### **1. Take Photographs Carefully.**

The care that is taken in collecting a DNA swab or a latent print lift should be the same as when capturing photographs that will be used for comparison.

#### **2. Take the photograph directly over the impression.**



**Figure 1:** A photograph taken from this position will be distorted.



**Figure 2:** This is the position a photograph should be taken at. Using a tripod is best because it reduces shake and ensures you are directly over the impression. You can check the position of the camera from the side or using a level.

## Shoeprint and Tire Track Collection Guide

### 3. Use a ruler in the plane of the impression; 'L' shaped rulers are best.

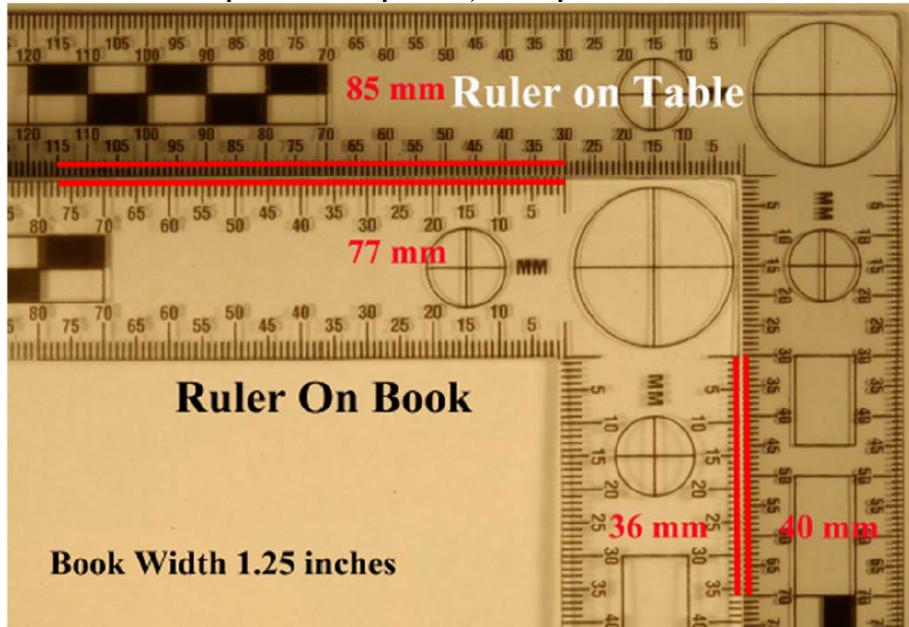
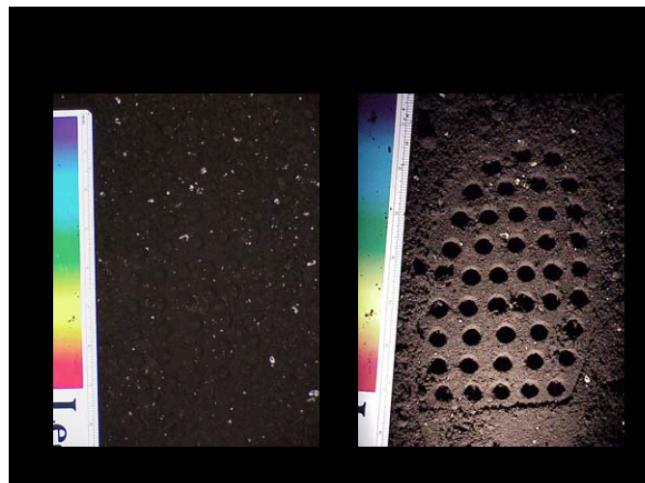


Figure 3

When photographing an impression that has depth it is important to push the scale in to the same depth as the impression. If this is not done the size of the impression will be altered in the photograph as seen in the rulers pictured in Figure 3.

### 4. Use lighting from different angles and take multiple photographs.



When photographing try different lighting techniques.

Try taking the same picture

- with a flash
- without a flash
- side lighting (flash light or detachable flash) from different angles

Lighting from different angles may help bring out details in the impression. If it is very sunny outside shielding your impression and introducing your own light may be helpful as well.

Figure 4 : The impression on the left is top lit (with an attached flash) and the impression on the right is side lit (with a flashlight or removable flash at an angle).