

Universidades de Burgos, León y  
Valladolid

Máster universitario

# Inteligencia de Negocio y Big Data en Entornos Seguros



**TFM del Máster Inteligencia de Negocio  
y Big Data en Entornos Seguros**

**Clasificación de individuos a  
partir de imágenes oculares y  
redes neuronales pre-entrenadas.**

Presentado por Ignacio Ponsoda Llorens  
en Universidad de Burgos — 4 de julio de 2022  
Tutores: Dr. José Francisco Díez Pastor y Dr.  
Pedro Latorre Carmona



# Universidades de Burgos, León y Valladolid



## Máster universitario en Inteligencia de Negocio y Big Data en Entornos Seguros

D. José Francisco Díez Pastor, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos, y D. Pedro Latorre Carmona, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Exponen:

Que el alumno Ignacio Ponsoda Llorens, con DNI 21698927Z, ha realizado el Trabajo final de Máster en Inteligencia de Negocio y Big Data en Entornos Seguros titulado 'Clasificación de individuos a partir de imágenes oculares y redes neuronales pre-entrenadas'.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección de los que suscriben, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 4 de julio de 2022

Vº. Bº. del Tutor:

Vº. Bº. del tutor:

D. José Francisco Díez Pastor

D. Pedro Latorre Carmona





## Resumen

La utilización de la biometría para mejorar la seguridad, principalmente en lo referente al acceso de dispositivos electrónicos, es un recurso ampliamente empleado en la actualidad. El iris es uno de los elementos biométricos que mayores dificultades presentan para su suplantación, y por ello, su utilización en este campo ha atraído la atención de la comunidad científica estas últimas dos décadas.

Las redes neuronales han demostrado ser útiles para extraer características del iris. Esta extracción se puede llevar a cabo no solo entrenando una red neuronal desde cero, sino también adaptando una ya pre-entrenada. Respecto a la utilización de técnicas de adaptación de la red neuronal se plantean dos preguntas. ¿Es necesario aislar el iris para llevar a cabo la adaptación de la red neuronal? ¿Se puede aplicar también para conjuntos de datos pequeños?

En este proyecto se han adaptado redes neuronales, inicialmente entrenadas para clasificar diversos objetos, para que sean capaces de identificar a un individuo utilizando su imagen ocular.

Para ello, se han utilizado dos enfoques. En un primer enfoque, las redes neuronales se han adaptado utilizando imágenes oculares completas, mientras que, para el segundo enfoque, se ha adaptado la red neuronal con imágenes oculares donde previamente se ha aislado el iris, ya que a priori, el iris la zona de la imagen ocular que mejor permite la identificación de individuos.

Además, se han utilizado técnicas de ampliación del *dataset* original, a fin de contar con un mayor número de muestras de cada individuo y también, mejorar la robustez de las redes neuronales adaptadas.

Los resultados muestran que las mejores tasas de clasificación se han dado en el enfoque donde se utilizaba la imagen ocular completa, sin que las técnicas de ampliación del *dataset* hayan permitido mejorar la tasa de clasificación.

Como futuras líneas de trabajo, se establecen la utilización de redes neuronales pre-entrenadas distintas, así como testear el modelo con imágenes realizadas fuera del entorno académico.

## Descriptores

biometría, iris, redes neuronales

## **Abstract**

The use of biometrics to improve security, mainly in relation to access to electronic devices, is a widely used resource today. The iris is one of the biometric elements that present the greatest difficulties for its impersonation, and that is why its use in this field has attracted the attention of the scientific community in the last two decades.

In this project, neural networks, initially trained to classify various objects, have been adapted to be able to identify an individual using their eye image.

To do this, two approaches have been used. In the first approach, the neural networks have been adapted using complete eye images, while, for the second approach, the same has been done, but the iris area has been isolated, which is a priori the area of the eye image that best allows the identification of individuals.

In addition, extension techniques of the original dataset have been used, in order to have a greater number of samples of each individual and also to improve the robustness of the adapted neural networks.

The results show that the best classification rates have occurred in the approach where the complete ocular image was used, without improvement in the classification rate by the datasets that had used data augmentation techniques.

In future lines of work, the use of different pre-trained neural networks is established, as well as testing the model with images taken outside the academic environment.

## **Keywords**

biometrics, iris, neural networks, fine-tuning, data augmentation



---

# Índice general

---

Índice general	iii
Índice de figuras	vi
Índice de tablas	vii
<b>1. Introducción</b>	<b>3</b>
1.1. Outline . . . . .	7
<b>Memoria</b>	<b>3</b>
<b>2. Objetivos del proyecto</b>	<b>9</b>
<b>3. Conceptos teóricos</b>	<b>11</b>
3.1. Biometría . . . . .	11
3.2. Inteligencia Artificial . . . . .	12
<b>4. Técnicas y herramientas</b>	<b>23</b>
4.1. Hardware . . . . .	23
4.2. Github . . . . .	24
4.3. Python . . . . .	24
4.4. Scrum . . . . .	25
<b>5. Aspectos relevantes del desarrollo del proyecto</b>	<b>27</b>
5.1. Preparación del dataset . . . . .	28
5.2. Pre-procesamiento de los datos . . . . .	28
5.3. Adaptación de la red neuronal . . . . .	29

<b>6. Trabajos relacionados</b>	<b>33</b>
6.1. Sistema clasificador de iris . . . . .	33
6.2. Iris Recognition Development Techniques: A Comprehensive Review . . . . .	34
6.3. Iris Recognition Using Wavelet Transform and Artificial Neural Networks . . . . .	35
6.4. Reliable pupil detection and iris segmentation algorithm based on SPS . . . . .	35
6.5. An experimental study of deep convolutional features for iris recognition . . . . .	36
6.6. Deep Learning-Based Feature Extraction in Iris Recognition: Use Existing Models, Fine-tune or Train From Scratch? . . .	36
<b>7. Conclusiones y Líneas de trabajo futuras</b>	<b>37</b>
7.1. Conclusiones . . . . .	37
7.2. Líneas de trabajo futuras . . . . .	37
 <b>Apéndices</b>	 <b>39</b>
<b>Apéndice A Plan de Proyecto Software</b>	<b>41</b>
A.1. Introducción . . . . .	41
A.2. Planificación temporal . . . . .	41
A.3. Estudio de viabilidad . . . . .	43
<b>Apéndice B Especificación de Requisitos</b>	<b>47</b>
B.1. Introducción . . . . .	47
B.2. Objetivos generales . . . . .	47
B.3. Catalogo de requisitos . . . . .	47
B.4. Especificación de requisitos . . . . .	48
<b>Apéndice C Especificación de diseño</b>	<b>51</b>
C.1. Introducción . . . . .	51
C.2. Diseño de datos . . . . .	51
C.3. Diseño procedimental . . . . .	53
C.4. Diseño arquitectónico . . . . .	53
<b>Apéndice D Documentación técnica de programación</b>	<b>55</b>
D.1. Introducción . . . . .	55
D.2. Estructura de directorios . . . . .	55
D.3. Manual del programador . . . . .	55
D.4. Compilación, instalación y ejecución del proyecto . . . . .	61

<i>Índice general</i>	v
D.5. Pruebas del sistema . . . . .	62
<b>Bibliografía</b>	<b>63</b>

---

# Índice de figuras

---

1.1. Segmentación de los bordes límbico y pupilar de forma clásica. .	4
1.2. Enfoque utilizando las imágenes sin preprocesamiento. . . . .	5
1.3. Enfoque utilizando la normalización y segmentación del iris. . .	6
3.4. Partes del ojo. . . . .	12
3.5. Representación del funcionamiento de una red neuronal, inspirada en el vídeo, <i>How Deep Neural Networks Work</i> . . . . .	14
3.6. Ejemplo de <i>data augmentation</i> por ruido gaussiano. . . . .	16
3.7. ejemplo de transformación de identidad. . . . .	17
3.8. ejemplo de transformación por reflexión. . . . .	17
3.9. ejemplo de transformación por escalamiento. . . . .	18
3.10. ejemplo de transformación por traslación. . . . .	18
3.11. ejemplo de transformación por rotación. . . . .	19
3.12. Ejemplo de binarización del ojo durante la segmentación. . . . .	20
3.13. Ejemplo de normalización del ojo. . . . .	20
C.1. Estructura de directorios de CASIA-V1. . . . .	52

---

# Índice de tablas

---

3.1. Transformaciones afines aplicadas en el trabajo. . . . .	16
4.2. Características del equipo . . . . .	23
5.3. <i>Datasets</i> utilizados para el <i>fine-tuning</i> . . . . .	27
5.4. Configuración del <i>pipeline</i> . . . . .	30
5.5. Tasa de acierto de los modelos. . . . .	31
A.1. Gastos de contratación del estudiante. . . . .	44
A.2. Gastos de contratación de los tutores (por tutor). . . . .	44
A.3. Gastos totales del proyecto. . . . .	44
A.4. Licencias de las librerías utilizadas. . . . .	45



# Memoria





---

# Introducción

---

El *Oxford Learners Dictionaries* <sup>1</sup> define la biometría como la utilización de características humanas para poder identificar a las personas, lo cual supone una piedra angular de muchos sistemas de seguridad.

Las principales características utilizadas para identificación de individuos son la cara, el iris, y las huellas dactilares. Estas características identificativas se han convertido en elementos fundamentales en la seguridad de los dispositivos electrónicos de la población a nivel mundial.

Esta dependencia de la biometría para acceder a los dispositivos supone indirectamente una dependencia en ella para la protección de la información privada de la población, ya que hoy en día, los dispositivos electrónicos cuentan con gran cantidad de información sensible.

Dentro de la biometría, el iris se utiliza como un elemento de reconocimiento biométrico de gran eficacia, tanto por su inmutabilidad a lo largo del tiempo como por resultar un valor único y personal, que supone que dos personas no puedan ser identificadas con un mismo iris [10]. Tanto es así, que el iris es incluso utilizado en los procesos post-mortem para poder determinar la pertenencia del cuerpo [4].

No obstante, contar con técnicas eficientes de segmentación y extracción de atributos del iris hoy en día sigue siendo un problema. En muchos casos se siguen utilizando técnicas clásicas que buscan encontrar bordes en el iris para poder aislarlo, como se muestra en la imagen 1.1, y ocurre algo similar con la extracción de características. Pero recientemente, se ha demostrado que las redes neuronales pueden ser utilizadas para la segmentación del iris [9] y la extracción de características [2][3].

---

<sup>1</sup>Definición consultada en <https://www.oxfordlearnersdictionaries.com/definition/english/biometric>

Existen dos opciones a la hora de trabajar con las redes neuronales para estos procesos, o bien entrenar una red neuronal desde cero, o bien adaptar una ya existente al conjunto de datos a utilizar. Ha sido demostrado que el uso de redes neuronales adaptadas funcionan mejor que las creadas desde cero cuando se cuenta con *datasets* de un tamaño considerable (10 mil imágenes en el caso de [3]) y si se aísla previamente el iris en las imágenes. Con ello surgen dos preguntas:

1. ¿Como funcionaría la adaptación si se utiliza la imagen ocular completa en vez de una con el iris aislado?
2. ¿Este método sería aplicable también a *datasets* pequeños (menos de mil imágenes), donde el entrenamiento de redes neuronales desde cero se dificulta considerablemente? ¿Sería beneficioso aplicar *data augmentation* en este caso para ampliar artificialmente su tamaño?

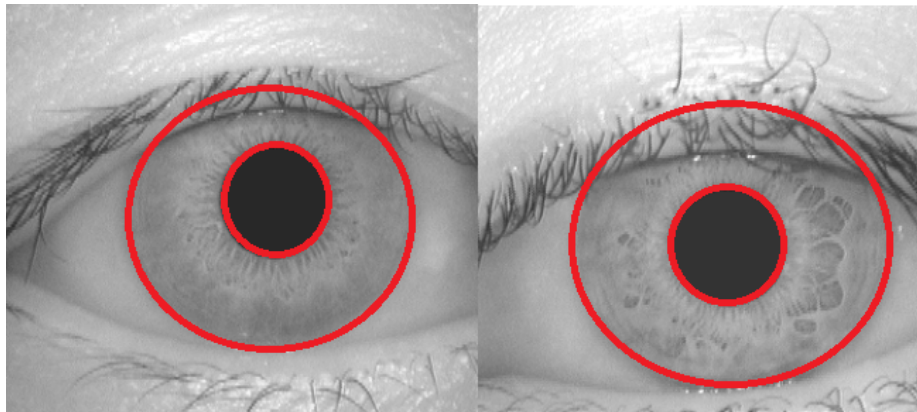


Figura 1.1: Segmentación de los bordes límbico y pupilar de forma clásica.

En este estudio, se ha adaptado una red neuronal pre-entrenada con el fin de identificar distintos objetos para que sea capaz de identificar de individuos a través de sus imágenes oculares. Por ello, proponemos dos enfoques distintos. Ambos llevados a cabo una segunda vez utilizando técnicas de *data augmentation* para comprobar su efecto sobre el modelo.

En ambos enfoques se ha empleado como red neuronal VGG16<sup>2</sup>, que ha sido pre-entrenada con ImageNet<sup>3</sup> para poder distinguir distintos objetos

<sup>2</sup>Esta red neuronal cuenta con 16 capas y ha sido entrenada con más de un millón de imágenes.

<sup>3</sup>ImageNet es un proyecto donde se proporciona una gran base de datos de imágenes para usos no comerciales <https://www.image-net.org/>

en base a una imagen. Esta red ha sido adaptada, aplicando técnicas de *fine-tuning*<sup>4</sup> con un *dataset* de imágenes oculares etiquetadas.

En un primer enfoque, se han utilizado las imágenes oculares completas del *dataset*<sup>5</sup> para adaptar la red neuronal, siguiendo el proceso que se muestra en la imagen 1.2

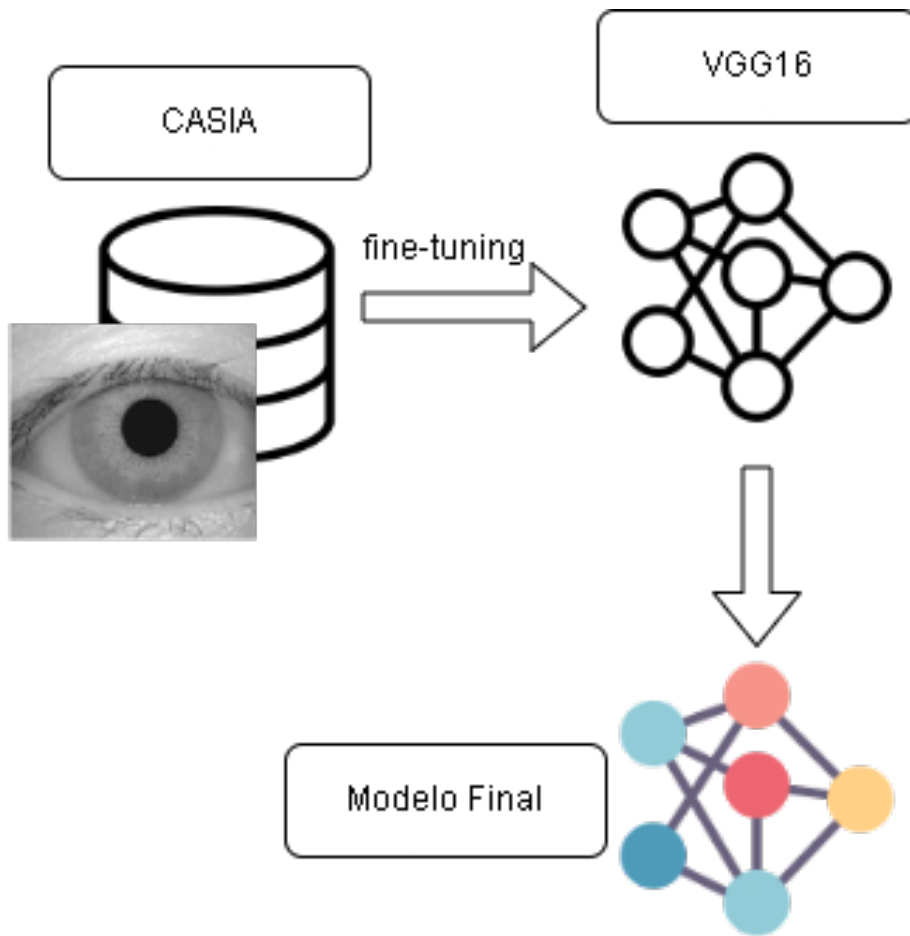


Figura 1.2: Enfoque utilizando las imágenes sin preprocesamiento.

<sup>4</sup>El *fine-tuning* permite adaptar el modelo para que, al llevarse a cabo la clasificación, no muestre resultados relativos al *dataset* con el que ha sido entrenada, sino con el que ha sido adaptada, tal como se muestra en la imagen 1.2. El *fine-tuning* está definido en la sección 3.2.

<sup>5</sup>Revisar sección 3.1

En un segundo enfoque, se ha aplicado el proceso de segmentación<sup>6</sup> y normalización<sup>7</sup> del iris, desarrollado en [2], al *dataset* de imágenes, para posteriormente seguir el mismo proceso de adaptación de la red neuronal, como se muestra en la imagen 1.3.

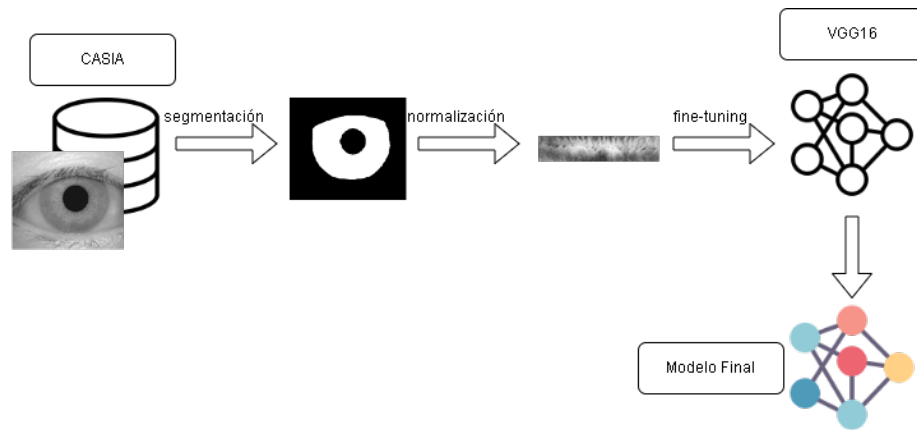


Figura 1.3: Enfoque utilizando la normalización y segmentación del iris.

Así mismo, para ambos enfoques se han creado dos modelos más utilizando técnicas de *data augmentation*<sup>8</sup> para aumentar el número de imágenes oculares por individuo y entrenar al modelo con nuevas variaciones de las imágenes, haciendo el modelo más robusto ante estas posibles nuevas variaciones.

El resultado de esta adaptación de la red neuronal es la creación de un modelo capaz de identificar al individuo que hay detrás de una imagen ocular no etiquetada.

Para determinar que enfoque ha sido capaz de identificar de forma más eficiente a los individuos, se ha comparado la tasa de acierto de los modelos resultantes de las adaptaciones a la hora de clasificar nuevas imágenes oculares.

El objetivo principal de este proyecto ha sido el de analizar cual de estos enfoques es más óptimo para el reconocimiento de individuos a través de sus imágenes oculares, así como analizar las tasas de acierto de los modelos resultantes.

<sup>6</sup>Definición de segmentación en la sección 3.2.

<sup>7</sup>Definición de normalización en la sección 3.2.

<sup>8</sup>El término *data augmentation* hace referencia a un conjunto de técnicas que permite ampliar el *dataset* original con variaciones de el mismo. En la sección *Data augmentation* puede encontrar una explicación más detallada.

## 1.1. Outline

El resto del documento se estructura de la siguiente manera. El capítulo 2 [Objetivos del proyecto](#) define las principales motivaciones del proyecto. El capítulo 3 [Conceptos teóricos](#) se concentra en los aspectos teóricos del proyecto. En el capítulo 5 [Aspectos relevantes del desarrollo del proyecto](#) se muestran los aspectos más relevantes que se han desarrollado. En el capítulo 6 [Trabajos relacionados](#) los trabajos relacionados y en el capítulo 7 [Conclusiones y Líneas de trabajo futuras](#) las conclusiones y las líneas de trabajo futuras.



---

## Objetivos del proyecto

---

En este apartado se detallan los objetivos principales para el desarrollo del proyecto:

- Al re-utilizar los procesos de segmentación y normalización de [2], un primer objetivo ha sido optimizar el código ya existente para mejorar su reproducibilidad y que ello permitiera una mayor flexibilidad a la hora de ejecutar el mismo.
- El segundo objetivo ha sido el de utilizar técnicas de *fine-tuning* con el fin de adaptar los resultados de clasificación de una red neuronal pre-entrenada al *dataset* utilizado en el proyecto, y comprobar la capacidad de la red adaptada para identificar individuos a partir de sus imágenes oculares.
- El tercer objetivo ha sido la aplicación de técnicas de *data augmentation* para comprobar si su utilización supone una mejora en la robustez del modelo.
- Como objetivo final, analizar la capacidad de identificación de las redes neuronales desarrolladas y el impacto que tienen, tanto el aislamiento del iris como el *ata augmentation* en las identificaciones de individuos a partir de su imagen ocular.





---

## Conceptos teóricos

---

En esta sección se desarrollan los conceptos teóricos utilizados necesarios para comprender el proyecto.

### 3.1. Biometría

Como se ha descrito en el capítulo [Introducción](#), la biometría es el estudio que permite la identificación de un individuo a través de determinadas características asociadas a su persona, principalmente la cara, las huellas dactilares y el iris.

#### Partes del ojo

El ojo se divide en tres capas principales, la capa externa llamada esclerótica, la capa intermedia llamada iris y la capa interna llamada retina. Entre la capa externa e intermedia se encuentra el borde límbico, mientras que, en la capa intermedia, el borde pupilar separa el iris de la pupila<sup>9</sup>. Las partes del ojo han sido etiquetadas en la imagen [3.4](#).

---

<sup>9</sup>Información accesible desde <https://www.cigna.com/es-us/individuals-families/health-wellness/hw/anatoma-y-funcin-del-ojo-hw121946>. 2022, 22 de junio

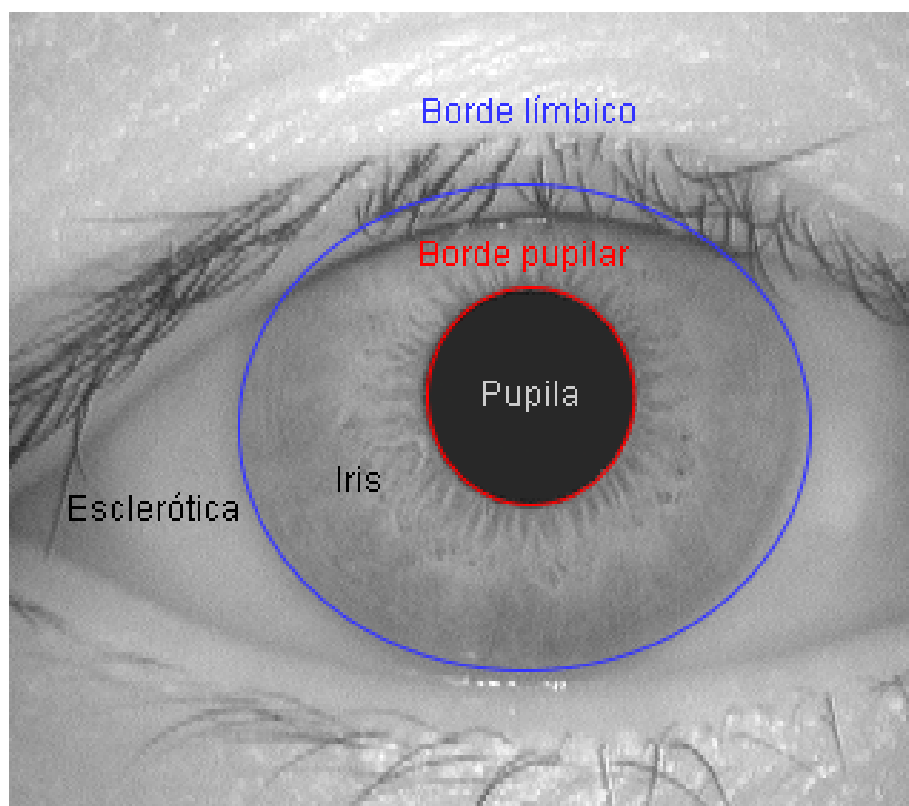


Figura 3.4: Partes del ojo.

### ***Dataset CASIA-IrisV1***

El *dataset* CASIA-IrisV1 ha sido el utilizado en el proyecto para adaptar las redes neuronales. Se trata de una base de datos que contiene 756 imágenes del iris de un total de 108 sujetos. Dichas fotos fueron tomadas por el [Center for Biometrics and Security Research](#) en dos sesiones, donde se tomaron 3 y 4 muestras respectivamente por cada individuo, con una resolución de 320x280. La pupila fue automáticamente remplazada por la propia organización para evitar que en ella se reflejasen las luces de las fotografías<sup>10</sup>.

## **3.2. Inteligencia Artificial**

La Real Academia Española define la inteligencia artificial como una disciplina cuyo objetivo principal es la creación de programas capaces de

---

<sup>10</sup>El proceso de la toma de muestras se describe en <http://www.cbsr.ia.ac.cn/IrisDatabase.htm>.

realizar funciones similares a los de la mente humana<sup>11</sup>. En este proyecto ha sido utilizada principalmente para la segmentación del iris, definida más adelante, y la adaptación de la red neuronal al *dataset*.

### ***Deep Learning***

El *deep learning* es un tipo de inteligencia artificial que se dedica a resolver los problemas que, siendo intuitivos para el ser humano, son complejos para la inteligencia artificial. Para resolverlos, la inteligencia artificial intenta replicar la toma de decisiones que hacen los seres humanos a través de la experiencia y la jerarquización de conceptos.

Al basarse en la experiencia, el *deep learning* no necesita que se le definan todos los parámetros para poder completar la tarea, puesto que es capaz de aprender por sí misma. Por otro lado, la jerarquización de conceptos permite utilizar conceptos complejos al basarlos en conceptos más fáciles de entender [7].

En el *deep learning*, la red neuronal cuenta con una capa de entrada y una capa de salida, y entre ellas cuenta con una o varias capas ocultas. Las capas están conectadas entre ellas por conexiones ponderadas, que determinan la importancia de cada elemento de la capa y que permite, en el caso de este proyecto, la identificación de individuos a través de su imagen ocular en la capa de salida. Este funcionamiento ha sido representado en la imagen 3.5.

---

<sup>11</sup>En base a la definición de <https://dle.rae.es/inteligencia>.

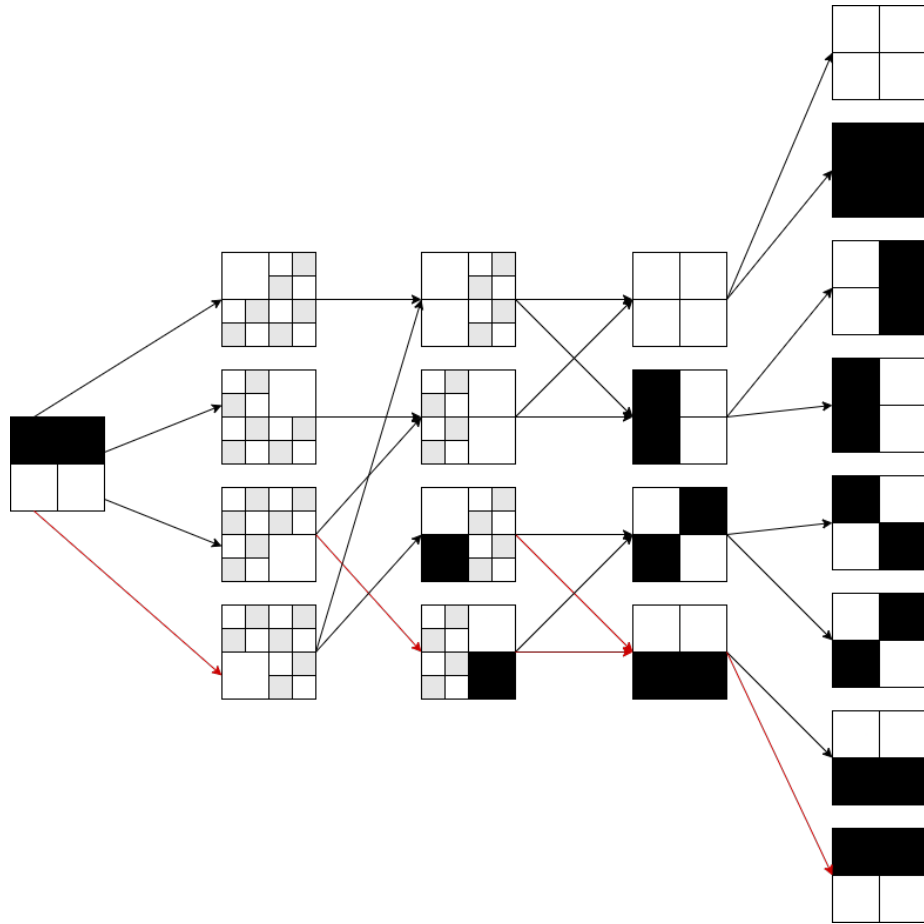


Figura 3.5: Representación del funcionamiento de una red neuronal, inspirada en el vídeo, [How Deep Neural Networks Work](#).

## *Pipeline*

Los *pipelines* son secuencias de procesos encadenados, donde el *output* del proceso anterior funciona como *input* del siguiente proceso.

Tomando como ejemplo la imagen 1.3, un primer proceso sería la segmentación, que tomaría como entrada el *dataset* de CASIA y tendría como salida las imágenes segmentadas. A su vez, estas imágenes segmentadas, serían la entrada del proceso de normalización, y así sucesivamente.

La utilización de *pipelines* en el proyecto ha permitido un mayor control de los distintos procesos, al poder establecer una configuración general en la cadena de procesos, y ha permitido modificar de forma más sencilla el orden de los procesos, permitiendo así experimentar con nuevas combinaciones.

## ***Data augmentation***

La utilización de técnicas de *data augmentation* es un proceso común en el análisis de imágenes, y en aquellos proyectos donde se utilicen procesos estadísticos.

Consiste en aumentar el tamaño del *dataset* con la creación artificial de nuevas imágenes, que son producidas a partir de imágenes del *dataset* original.

Para ello, la imagen original se modifica, comúnmente con la aplicación de ruido gaussiano o transformaciones geométricas de tipo afín, obteniendo como resultado una imagen que deriva de la original, pero que cuenta con ciertas diferencias, que serán más o menos pronunciada dependiendo de las técnicas de *data augmentation* que se le apliquen, así como de los parámetros utilizados para realizar las modificaciones<sup>12</sup>.

En el caso de los procesos de entrenamiento de las redes neuronales, es común la utilización de técnicas de *data augmentation* principalmente por dos situaciones, aunque estas no son limitantes:

- **Número insuficiente de datos:** en este caso, el *data augmentation* se aplica porque el *dataset* no es lo suficientemente grande como para conseguir unos resultados significativos en la creación de una red neuronal.
- **Aumento de la robustez del modelo:** el segundo supuesto principal por el cual se utiliza *data augmentation* es la utilización de elementos que añadan complejidad a la creación del modelo, lo cual le proporcionará una mejor actuación ante la aparición de nuevas complejidades.

### **Ruido gaussiano**

La primera de las técnicas de *data augmentation* utilizadas es el ruido gaussiano. También conocido como ruido blanco, esta técnica provoca que los píxeles de una imagen cambien su valor siguiendo una distribución gaussiana, como se puede observar en la figura 3.6 .

---

<sup>12</sup>Por ejemplo, si se rota una imagen 2 grados, la diferencia con la original será mucho menor que si se rota 180 grados.

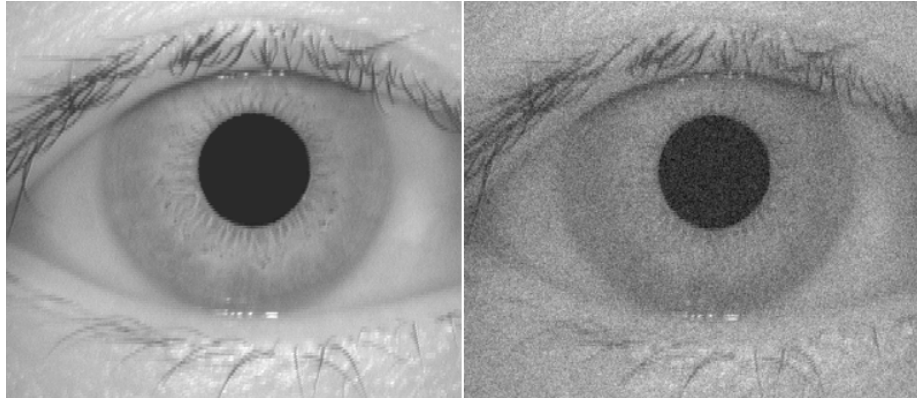


Figura 3.6: Ejemplo de *data augmentation* por ruido gaussiano.

### Transformaciones afines

Las transformaciones afines permiten aumentar el tamaño del dataset mediante la transformación de imágenes. Conservan el paralelismo de sus líneas rectas y paralelas y, de alguna forma, simulan una nueva perspectiva de la imagen original. Las transformaciones afines utilizadas en el proyecto han sido recogidas en la tabla 3.1 y se describen a continuación.

Transformaciones afines				
Identidad	Reflexión	Escalamiento	Traslación	Rotación
Figura 3.7	Figura 3.8	Figura 3.9	Figura 3.10	Figura 3.11

Tabla 3.1: Transformaciones afines aplicadas en el trabajo.

**Identidad** La transformación de identidad es un tipo de transformación afín en el que la imagen se copia sin ningún otro cambio, y se utiliza para la reutilización de los datasets.

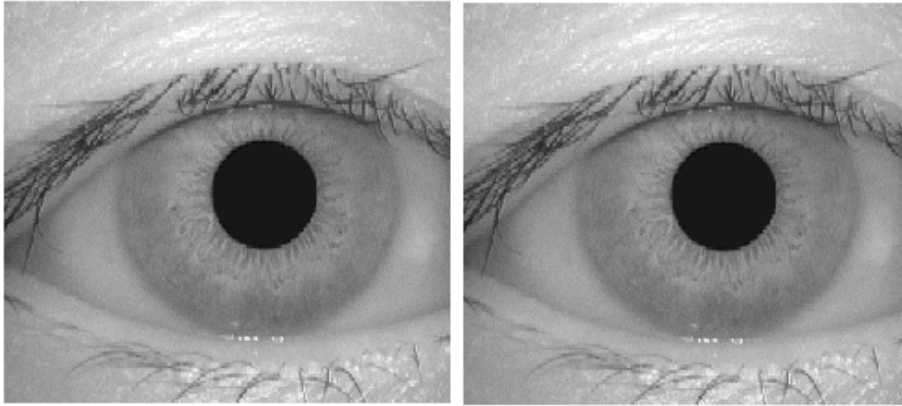


Figura 3.7: ejemplo de transformación de identidad.

$$\text{Identidad} : \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3.1)$$

**Reflexión** Se trata de un mapeo aplicado a la imagen a partir de un eje.

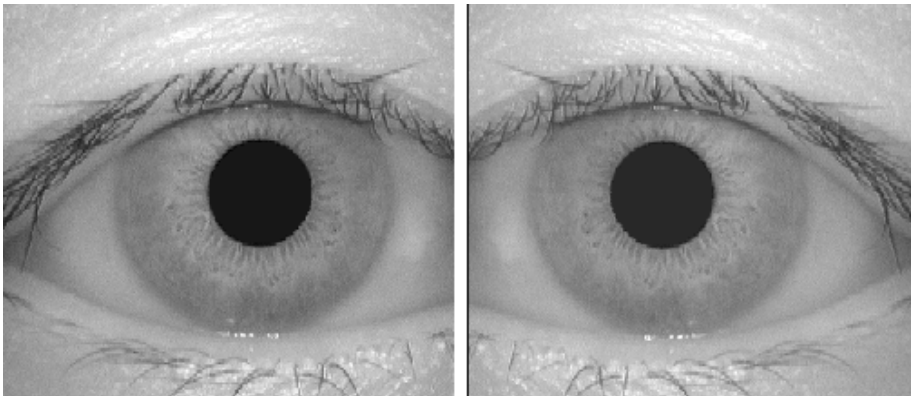


Figura 3.8: ejemplo de transformación por reflexión.

$$\text{Reflexión} : \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3.2)$$

**Escalamiento** Esta transformación modifica la escala de la imagen original, ya sea ampliándola o disminuyéndola.

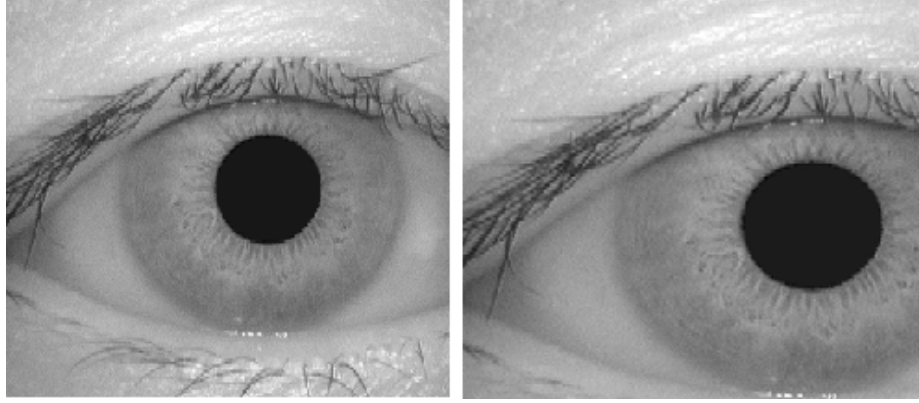


Figura 3.9: ejemplo de transformación por escalamiento.

$$\text{Escala miento} : \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3.3)$$

**Traslación** La imagen cambia de plano de coordenadas, pero no se modifican ni su tamaño, ni su forma, ni su orientación.

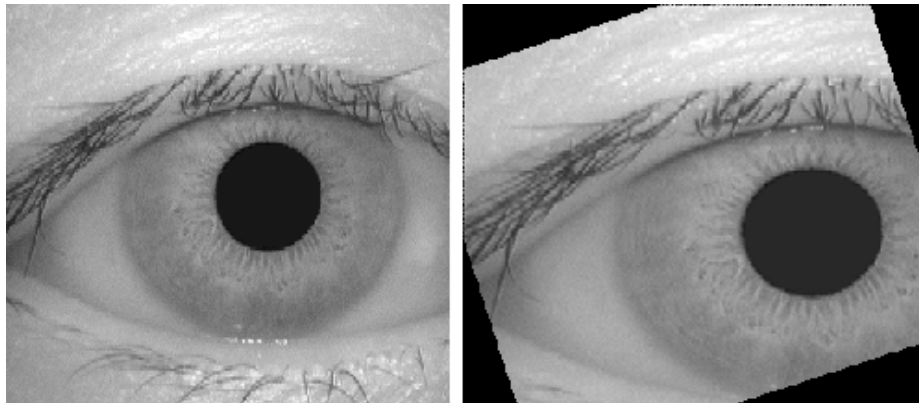


Figura 3.10: ejemplo de transformación por traslación.



$$\text{Traducción : } \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3.4)$$

**Rotación** Esta transformación aplica una transformación de  $\theta$  grados del plano.

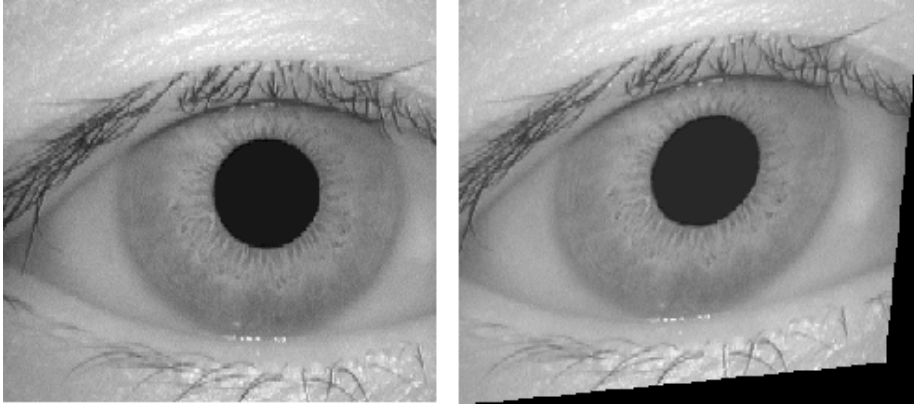


Figura 3.11: ejemplo de transformación por rotación.

$$\text{Rotación : } \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3.5)$$

## Pre-procesamiento

El preprocesamiento es la manipulación de los datos para que estos tengan el formato requerido para llevar a cabo su procesamiento.

En el caso de este proyecto, la fase de preprocesamiento es la fase en la que se extrae el iris de la imagen, puesto que, tal como indican diferentes estudios [2] [1] [10] [4] [8] [13] [9], dentro de la imagen ocular, es el iris el que permite identificar a las personas de una forma eficiente.

## Segmentación

Detección de los bordes *límico* y *pupilar*<sup>13</sup> utilizando el detector de bordes de Canny [5]. Estos bordes son clave para el aislamiento del iris [2].

<sup>13</sup>Definidos en la sección 3.1.

Una vez detectados los bordes, se procede a una binarización de la imagen 3.12, de forma que quede clara la división entre iris y resto del ojo.

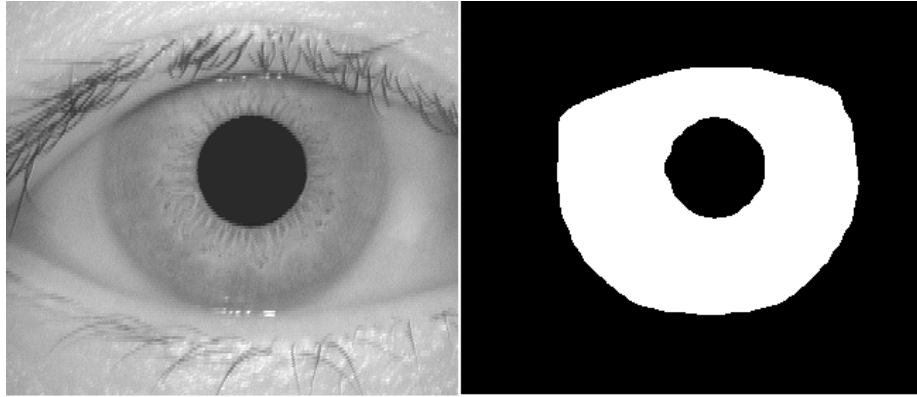


Figura 3.12: Ejemplo de binarización del ojo durante la segmentación.

### Normalización

Se puede definir como la proyección del iris a coordenadas polares (figura 3.13), utilizando el método Daugman<sup>14</sup> [6], de manera que se igualen los tamaños de las diferentes imágenes y permitan su comparación.

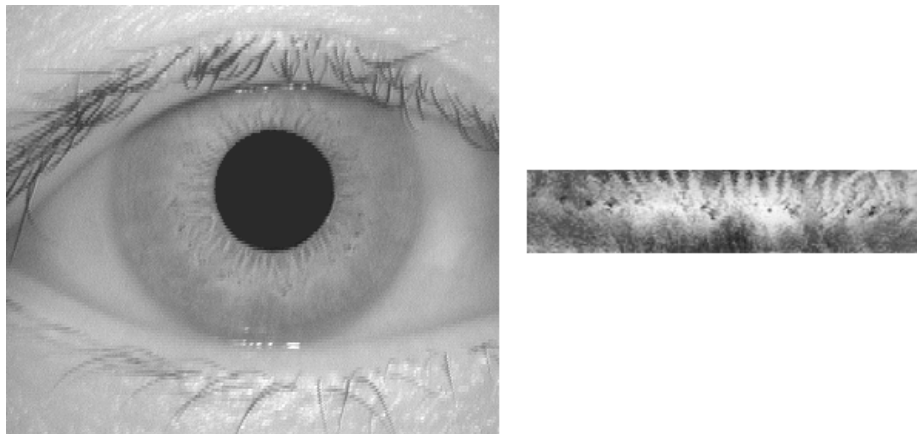


Figura 3.13: Ejemplo de normalización del ojo.

---

<sup>14</sup>Transforma el iris de coordenadas cartesianas a polares.

***Fine-tuning***

Es un proceso que se basa en adaptar redes neuronales, que previamente han sido entrenadas para reconocer ciertos objetos, con el fin de que pasen a reconocer los objetos de otro *dataset*. Ello permite beneficiarse de las capacidades de la red neuronal a la hora de identificar objetos, teniendo que entrenar solamente las capas relacionadas con la identificación y el etiquetado final.

En el caso de este proyecto, el *fine-tuning* ha sido utilizado para obtener un clasificador de imágenes para el dataset de CASIA sin tener que crear una red neuronal desde cero, sino adaptando la red neuronal VGG16 para dicha función.



---

# Técnicas y herramientas

---

En esta sección se describen los instrumentos y recursos utilizados a lo largo del proyecto.

## 4.1. Hardware

Para la ejecución de los *notebooks*, se ha contado con una máquina de la Universidad de Burgos, cuyas características se recogen en la tabla 4.2.

Elemento	Característica
Procesador	Intel Xeon E5-2630 v4 @ 2.20GHz (4 núcleos)
Memoria	128GB
GPUs	3 x Titan XP
Discos	SSD 500Gb, 2 x HDD 2TB

Tabla 4.2: Características del equipo

Para interactuar con la máquina, se ha utilizado una conexión SSH <sup>15</sup> así como un cliente Putty <sup>16</sup>, para la modificación de los *notebooks* a través de Jupyter *notebooks* <sup>17</sup>, dentro del ecosistema de Anaconda <sup>18</sup>.

---

<sup>15</sup>El *security shell* es un protocolo que permite el acceso remoto a través de un canal seguro <https://www.openssh.com/>.

<sup>16</sup>Putty es una implementación libre de SSH para Windows <https://www.putty.org/>.

<sup>17</sup>Se trata de una aplicación que permite editar y lanzar *notebooks* a través del buscador, <https://jupyter.org/>.

<sup>18</sup>Anaconda es una distribución libre, accesible a través del buscador, que es ampliamente utilizada en la ciencia de datos <https://www.anaconda.com/>.

## 4.2. Github

Github es una compañía que ofrece repositorios Git<sup>19</sup> en la nube. Estos repositorios se han utilizado en el proyecto para:

1. Control de versiones.
2. Seguimiento de las fases del proyecto.
3. Documentación de las reuniones y de los problemas a resolver.

El repositorio del proyecto es accesible desde <https://github.com/Ponsoda/tfm-iris-recognition>.

## 4.3. Python

Se ha utilizado el lenguaje de programación **Python** para las distintas fases del proyecto.

Entre las principales librerías utilizadas se encuentran:

- [imageio](#) - leer y escribir imágenes.
- [keras](#) - manejo de las redes neuronales.
- [matplotlib](#) - visualización de datos e imágenes.
- [numpy](#) - trabajo con las imágenes a nivel de arrays.
- [os](#) - acceso a los directorios.
- [opencv](#) - trabajo con las imágenes.
- [scikit-image](#) - transformación de las imágenes y el uso de dataset.
- [shutil](#) - copia de directorios.
- [tensorflow](#) - modificación de las redes neuronales.

---

<sup>19</sup>Sistema de control de versiones <https://git-scm.com/>.

## Redes neuronales

En el desarrollo del proyecto, se ha utilizado dos redes neuronales. En primer lugar, durante el proceso de adecuación del código procedente de [2], se ha utilizado una red neuronal basada en U-Net<sup>20</sup> y accesible desde el repositorio del *paper* [9]. Esta red neuronal, ya había sido específicamente entrenada para la segmentar el iris en imágenes oculares. Al tratarse de una red neuronal ya adaptada para dicho fin, se pudo utilizar directamente para la fase de extracción.

Para el proceso de *fine-tuning*, se ha utilizado la red neuronal VGG16, entrenada con el *dataset* de ImageNet para el reconocimiento de objetos, ya descrita en la sección de [Introducción](#).

## Visual Studio Code

Tanto para la redacción de la memoria con  $\text{\LaTeX}$  como para el la creación de los *notebooks* a nivel local, se ha utilizado [Visual Studio Code](#).

Se trata de un IDE<sup>21</sup> con licencia *open-source*, desarrollado por Microsoft que permite funciones de desarrollo, como la edición de código o su depuración.

## 4.4. Scrum

Scrum es un marco que ayuda a la organización de los equipos entorno a un proyecto, en base a Sprints de una determinada duración y permite una retroalimentación continua del proyecto, de forma que se asegura que el equipo trabaje en consonancia.

En el caso de este proyecto, se ha adecuado la metodología de Scrum para cuadrar reuniones semanales o bisemanales con los tutores del proyecto, estableciendo los sprints y los objetivos de cada sprint utilizando la plataforma Github.

---

<sup>20</sup>Red neuronal para la segmentación de imágenes biomédicas desarrollada por la Universidad de Freiburg <https://arxiv.org/abs/1505.04597>.

<sup>21</sup>Entorno de desarrollo integrado.





---

## Aspectos relevantes del desarrollo del proyecto

---

El proyecto se ha dividido en dos fases principales. La primera fase ha consistido en la asimilación, adecuación y optimización del código perteneciente a [2] para utilizarlo posteriormente en las fases de pre-procesamiento, estas son la segmentación y la normalización del iris. También se incluye en esta fase la aplicación de las técnicas de *data augmentation*. Por otro lado, en la segunda fase se ha desarrollado la adaptación de la red neuronal VGG16 a los cuatro conjuntos de imágenes derivadas del *dataset* creadas en la primera fase, estas son los conjuntos con/sin pre-procesamiento y con/sin *data augmentation*.

El objetivo de estas dos fases, es el de crear los cuatro *datasets* que se utilizarán en la fase de *fine-tuning* para crear los modelos finales, con los que se calculará la tasa de acierto relativa a su capacidad de clasificación. Estos cuatro datasets se reflejan en la tabla 5.3.

<i>Dataset</i>	Pre-procesamiento	<i>Data augmentation</i>
1		
2	x	
3		x
4	x	x

Tabla 5.3: *Datasets* utilizados para el *fine-tuning*.

## 5.1. Preparación del dataset

En los proyectos de inteligencia artificial, el procedimiento habitual para entrenar las redes neuronales es el de dividir los datos que se van a utilizar para entrenar a la red en dos subconjuntos. El subconjunto de mayor tamaño se utiliza para entrenar el modelo *per se*, mientras que el subconjunto de menor tamaño se utiliza en la validación del mismo.

En el caso de este proyecto caso, el *dataset* inicial ha sido dividido en un 70 % para la fase de entrenamiento, y un 30 % para calcular la tasa de acierto, siendo estos los porcentajes más comúnmente utilizados.

## 5.2. Pre-procesamiento de los datos

En la primera fase del proyecto, se ha llevado a cabo un pre-procesamiento del *dataset* para crear el conjunto de datos de imágenes del iris aisladas.

### Adecuación del código previo

Como se ha mencionado anteriormente, el código desarrollado en [2] para la segmentación y naturalización del iris en una imagen ocular ha sido adaptado para el proyecto.

Para ello, en una primera fase se ha asimilado el código completo del trabajo, de forma que se comprendiese a fondo el mismo, para seguidamente, en una segunda fase, se ha pasado a limpiar el código y adecuarlo a las necesidades del proyecto.

Este código permite la creación del segundo dataset utilizado en el proyecto, que siguiendo con la idea de que el iris es la mejor parte del ojo para identificar a una persona, lo aísla para entrenar a la red neuronal y evitar el ruido que pueda causar la imagen completa.

### Aplicación del *data augmentation*

El proceso de *data augmentation* permite ampliar el número de imágenes y la robustez de los modelos entrenados con los dos *datasets* anteriormente mencionados. Estos son las imágenes oculares completas y las imágenes con el iris aislado.

Por lo tanto, este proceso utiliza las técnicas de ruido gaussiano y transformaciones afines, explicadas en la sección 3.2 para crear dos nuevos conjuntos de datos.

Para la aplicación del ruido gaussiano, se han aplicado aleatoriamente valores de la desviación estándar de 2.5, 5 y 7.5.

Por su lado, las transformaciones afines se han aplicado de forma independiente al ruido gaussiano, es decir, una no es excluyente de la otra, ni las transformaciones afines son excluyentes entre ellas mismas.

Como resultado del *data augmentation*, los *datasets*, que originalmente contaban con 756 imágenes, cuentan tras este proceso con 1158 imágenes.

### Creación de el *pipeline*

Con el fin de mejorar el manejo de las diferentes secciones de código elaboradas, se ha optado por la utilización de una pipeline que permitiera controlar fácilmente la secuencia de ejecución del código así como la configuración del mismo.

Para ello, se ha encapsulado cada uno de los procesos del proyecto en funciones independientes y se ha creado un diccionario donde se ha establecido la configuración inicial.

Este diccionario era el único elemento de entrada y salida en la pipeline y permitía que en cada proceso se pudiese modificar o ampliar el diccionario, haciendo posible el cambio del orden de los procesos en el pipeline.

Los criterios de la configuración de la pipeline se muestran en la siguiente tabla 5.4. El *pipeline* se explica más detalladamente en el Anexo D.3.

## 5.3. Adaptación de la red neuronal

La adaptación de la red neuronal VGG16<sup>22</sup> es una parte central del proyecto, puesto que el objetivo principal del proyecto es comparar como comporta la red neuronal al ser adaptada por uno de los datasets desarrollados en el proyecto, en cuanto a tasa de acierto en la clasificación de individuos a través de sus imágenes oculares.

La adaptación de la red neuronal se ha aplicado a los cuatro *datasets*, estos son, por un lado, los dos *datasets* sin pre-procesamiento (uno de ellos con *data augmentation*) y, por otro lado, a los dos datasets a los que se les ha aplicado el pre-procesamiento (de nuevo, a uno de ellos también se le han aplicado técnicas de *data augmentation*).

---

<sup>22</sup>Se ha elegido esta red neuronal puesto que en [2] era la que mejores resultados de clasificación devolvía de forma promedio.

Nombre	Configuración
Raíz	Establecimiento del directorio raíz
Tamaño dataset	Tamaño de los <i>datasets</i> de entrenamiento y validación
<i>Data augmentation</i>	Tipo de <i>data augmentation</i> a aplicar
Imágenes y gráficas	Mostrar imágenes y gráficas, configurado individualmente para cada elemento.
Modelo	Nombre y ubicación donde guardar la red neuronal.
Epochs	Número de epochs por cada red neuronal entrenada.
<i>Batch</i>	Tamaño del <i>batch</i> .
<i>Random seed</i>	Tamaño del <i>random seed</i> .
Peso	Peso utilizado para el entrenamiento la red neuronal.
<i>Fine-tuning</i>	<i>Dataset</i> ha utilizar para el <i>fine-tuning</i> .

Tabla 5.4: Configuración del *pipeline*

## Aplicación del *fine-tuning*

En la utilización de técnicas de fine-tuning para la adaptación de la red neuronal VGG16 a los datasets del proyecto, se han seguido tres fases.

### Primera fase

En una primera fase, se ha definido una primera red neuronal cuyo objetivo es el de determinar las características que mejor definen a las imágenes, siendo la base del nuevo modelo.

Para definir esta red neuronal, se ha eliminado la capa final, que corresponde a la clasificación de la imagen, a una red neuronal que entrenada con ImageNet, menos la capa final, puesto que el modelo será utilizado como modelo base. El modelo se establece en este punto como no entrenable para que no se re-entrene en las siguiente fase, de forma que el modelo resultante funcione como un *inference model*<sup>23</sup>.

<sup>23</sup>Este modelo aprovecha el conocimiento de una red neuronal ya entrenada para clasificar imágenes pero interfiere en el resultado final, que es modificado por un nuevo *dataset*, que suele ser de un tamaño no lo suficientemente grande para entrenar la red neuronal desde cero.

### Segunda fase

En la segunda fase, se ha creado una nueva capa a partir del modelo base, que es capaz de clasificar a las imágenes en base a las imágenes oculares y sus etiquetas.

### Tercera fase

En esta tercera fase, se pasa a entrenar el modelo completo, descongelando el modelo base, pero manteniéndolo como *inference model* para evitar que se vuelva a entrenar, por lo que la extracción de características se hará sobre el nuevo *dataset* pero las *hidden layers* utilizadas serán principalmente las de VGG16.

## Clasificación de imágenes

Las imágenes se han clasificado utilizando los cuatro modelos creados en la sección anterior. Para ello, utilizando como dataset de entrada, el 30 % reservado en la primera fase.

En el caso las imágenes normalizadas, antes de ser clasificadas, se les aplica la fase de pre-procesamiento 3.2, de forma que estas se ajusten a la entrada requerida por el modelo.

### Tasa de acierto

La tasa de acierto representa el número de veces que el modelo a determinado correctamente la clase de la imagen.

En la siguiente tabla 5.5, se puede observar la tasa de acierto que han tenido los modelos, a la hora de relacionar las imágenes con los individuos. El cálculo de este valor se explica detalladamente en el Anexo D.3.

Modelo	Sin normalización		Con normalización	
	Con <i>data augmentation</i>	Sin <i>data augmentation</i>	Con <i>data augmentation</i>	Sin <i>data augmentation</i>
<i>Accuracy</i>	0.94	0.93	0.72	0.68

Tabla 5.5: Tasa de acierto de los modelos.

Estos resultados muestran, en primer lugar, que el *data augmentation* no supone sino un decremento del *accuracy*, tanto en los casos donde se

normaliza el iris como en los que no. Esto puede deberse a que, aunque el modelo sí tenga una mayor robustez, al calcularse este parámetro utilizando datos sin ninguna modificación, que tienen un gran parecido a las imágenes originales, el *data augmentation* disminuya ligeramente los resultados. No obstante, la mejora de la robustez previsiblemente permitirá al sistema funcionar de forma más eficiente en un contexto no académico.

Por otro lado, en cuanto a los mejores resultados utilizando imágenes no normalizadas, se debe de tener en cuenta que, por la forma en la que funcionan las redes neuronales, en cuanto a la reducción de imágenes para quedarse con sus características más representativas, tiene sentido que al proporcionar más elementos representativos del individuo, y no solo el iris, la propia red neuronal haya sido capaz de encontrar características en la imagen que son más eficientes para su clasificación, y que, de alguna forma, son ajenos a la zona propiamente del iris.

Además, cabe de tener en cuenta que, tal como se explica en [Dataset CASIA-IrisV1](#), el *dataset* utilizado ha sido sometido a un preprocesamiento previo, en el que se eliminó la pupila para evitar que el brillo emitido por las cámaras para tomar las propias imágenes pudiera afectar a la misma. Por lo tanto, realmente la normalización practicada solo ha afectado a la parte exterior al iris, y esto puede haber reducido su efecto.

---

## Trabajos relacionados

---

La utilización del iris como elemento biométrico que permite el acceso a los dispositivos electrónicos es un tema de estudio candente en los últimos años, tal como se explica en la [Introducción](#).

A continuación, se describen algunos de los trabajos centrados en la identificación de individuos a través de su iris en los que se ha basado este proyecto.

### 6.1. Sistema clasificador de iris

En este trabajo final de grado [2] de la Universidad de Burgos, se desarrolla un sistema de clasificación basado en el iris. Para ello, se utiliza el *dataset* de CASIA-V1. Este se somete primero a segmentación, utilizando el método de Wildes, y luego a normalización.

Posteriormente, se extraen los atributos más identificativos de cada iris, utilizando una red que ya ha sido pre-entrenada para dicho fin [9]. Finalmente, se procede a la clasificación de individuos utilizando técnicas de *machine learning*.

Como se ha comentado anteriormente, el código para la carga de imágenes y los procesos de segmentación y normalización del iris, desarrollados en este proyecto, han sido utilizados de este trabajo como base de las mismas secciones en el *pipeline*. El *pipeline* se explica más detalladamente en el Anexo [D.3](#).

## 6.2. Iris Recognition Development Techniques: A Comprehensive Review

En este artículo [10], los investigadores de la Universiti Putra Malaysia hablan de siete pasos en los que se divide un sistema de reconocimiento del iris:

1. adquisición
2. preprocesamiento
3. segmentación
4. normalización
5. extracción de características
6. selección de rasgos únicos y característicos
7. clasificación.

Este *paper* también describe una falta de trabajos entorno a *datasets* de baja calidad y realza que los sistemas de reconocimiento del iris se vuelven poco efectivos cuando las imágenes tienen rotaciones or reflejos, algo que intentamos de aplicar a nuestro proyecto, con el *data augmentation* (sección 3.2).

El mismo artículo resume los distintos *dataset* utilizados para estos estudios de reconocimiento de iris en los últimos años, el tipo de ruido utilizado en cada uno de ellos, así como describe a grandes rasgos cual es el método seguido en ellos para realizar el reconocimiento de individuos a través de sus imágenes oculares.

Los más comunes son los tipos de segmentación tradicional, basado en la creación de círculos estandarizados, y aunque aparece un incremento del uso de las redes neuronales para la segmentación en los últimos años. También se describen distintas técnicas de normalización del iris y extracción de sus características, así como distintos enfoques en el cálculo del *accuracy*.



### 6.3. Iris Recognition Using Wavelet Transform and Artificial Neural Networks

En este artículo de Hadeel N Abdullah [1], perteneciente a la University of Technology de Iraq, se lleva a cabo un pre-procesamiento de un *dataset* de imágenes oculares que incluye la extracción del iris utilizando la transformada de *Hough*<sup>24</sup> y la normalización con *Daugman*, ya explicado en la sección 3.2.

Posteriormente, tras eliminar el ruido de la imagen, la extracción se realiza con transformaciones de Wavelet<sup>25</sup>. Finalmente, se crea una red neuronal, donde se utiliza el *mean-squared error* para calcular los pesos en la red.

### 6.4. Reliable pupil detection and iris segmentation algorithm based on SPS

En este *paper* [11] encontramos el desarrollo de técnicas de *deep learning* para el reconocimiento del iris basado en una *convolutional neural network* residual. utilizando una red pre-entrenada de ResNet50<sup>26</sup> y técnicas de *fine-tuning* con el *dataset* IIT Delhi<sup>27</sup>.

En este artículo, no se utiliza *data augmentation* ni se pre-procesan las imágenes antes de aplicar el *fine-tuning*. La red neuronal es entrenada con *cross-entropy loss function*.

---

<sup>24</sup>Tiene como premisa que los bordes del ojo se pueden considerar círculos y se extrae el iris utilizando dicha forma.

<sup>25</sup>Transforma los datos de las imágenes con la transformada de Wavelet.

<sup>26</sup>Red neuronal con 50 capas, entrenada para reconocer imágenes de entre mil categorías.

<sup>27</sup>Accesible desde [https://www4.comp.polyu.edu.hk/~csajaykr/IITD/Database\\_Iris.htm](https://www4.comp.polyu.edu.hk/~csajaykr/IITD/Database_Iris.htm)

## 6.5. An experimental study of deep convolutional features for iris recognition

En este artículo [12], se hizo uso del *dataset* CASIA 10k<sup>28</sup> y la arquitectura VGG16, para realizar una *PCA*<sup>29</sup> y extraer los elementos más característicos de las imágenes. En una segunda fase, los investigadores utilizan algoritmos de *machine learning* para clasificar las imágenes, consiguiendo unos altos porcentajes de acierto.

## 6.6. Deep Learning-Based Feature Extraction in Iris Recognition: Use Existing Models, Fine-tune or Train From Scratch?

En este artículo [3], los investigadores de la Universidad de Notre Dame se preguntan si vale la pena, teniendo un *dataset* de imágenes oculares de un tamaño considerable, entrenar a una red neuronal desde cero o aplicar técnicas de *fine-tuning* sobre una red ya pre-entrenada.

Para contestar a esta pregunta, utilizan técnicas de *deep learning* para clasificar imágenes de ojos del *dataset* CASIA 10k sin aplicar técnicas de *data augmentation* sobre él. Para la segmentación y normalización del iris se utiliza OSIRIS<sup>30</sup>.

Para la parte de *fine-tuning*, se utiliza la red pre-entrenada ResNet50 y posteriormente, se realiza la clasificación con SVM<sup>31</sup>. La conclusión que se extrae del artículo es que, al menos en este caso, el clasificador funciona mejor al adaptar una red neuronal con *fine-tuning* antes que crear la nuestra propia, aun siendo el *dataset* lo suficientemente grande para hacerlo.

---

<sup>28</sup>Accesible desde <http://www.nlpr.ia.ac.cn/pal/CASIA10K.html>

<sup>29</sup>*Principal Components Analysis* es un método de reducción de dimensiones.

<sup>30</sup>Herramienta de código abierto utilizada para el reconocimiento del iris <https://github.com/tohki/iris-osiris>.

<sup>31</sup>Support vector machine es un método de aprendizaje supervisado.

---

# Conclusiones y Líneas de trabajo futuras

---

En este apartado se explican las conclusiones del proyecto, así como se establecen las posibles líneas de trabajo futuras.

## 7.1. Conclusiones

Como conclusión de este proyecto, se puede determinar que la utilización de técnicas de *fine-tuning* para llevar a cabo todo el proceso de clasificación, resultan, en este caso de estudio, más eficientes cuanto mayor es el tamaño de la imagen, y no cuando se aísla la zona del iris, como se podía pensar al entender que el iris es, con diferencia, la parte más característica del ojo.

Por otro lado, la utilización de técnicas de *data augmentation* con el fin de aumentar el tamaño de los *datasets* y permitir la creación de unos modelos más robustos, no ha influido positivamente en las tasas de acierto de los modelos, sino todo lo contrario.

## 7.2. Líneas de trabajo futuras

Las líneas de trabajo futuras se podrían determinar con la utilización de estas técnicas con distintos *datasets*, utilizando imágenes que no estén específicamente preparadas para este tipo de estudios. Además, sería interesante probar distintos modelos para el *fine-tuning* de los distintos *datasets* así como la utilización de redes neuronales para llevar a cabo el proceso de segmentación del iris.

El objetivo de todo ellos sería el conseguir un modelo lo suficientemente robusto que permitiese su utilización en un programa de escritorio, con una primera fase de *fine-tuning* con imágenes del ojo del usuario y una segunda fase donde este se utilizase como método de seguridad para el acceso a ciertos documentos de los aparatos electrónicos.

# Apéndice



## Apéndice A

---

# Plan de Proyecto Software

---

### A.1. Introducción

La planificación del proyecto ha sido una parte esencial, puesto que ha permitido la coordinación entre estudiante y tutores de forma online.

### A.2. Planificación temporal

Para la planificación temporal se ha utilizado la metodología SCRUM basada en sprints, generalmente de una o dos semanas.

#### Sprint 1

- Creación de un repositorio para el control de las versiones y el seguimiento del trabajo.
- Investigación sobre la materia de estudio (antecedentes, casos de uso, estado del arte)
- Investigación sobre posibles dataset de iris para el estudio.
- Investigación sobre la metodología que mejor se adapte al proyecto.
- Estudio sobre el uso de *data augmentation* sobre *datasets* de iris.
- Investigación sobre el uso de pipelines para la mejora de la reproducibilidad del trabajo.
- Investigación sobre los transformes de scikit-learn

## Sprint 2

- Investigación sobre *pipelines* que se ejecuten de forma condicional.
- Investigación sobre posibles redes neuronales para la segmentación del iris.
- Investigación para el uso de *data augmentation* en el paquete Keras.

## Sprint 3

- Clarificación de objetivos del proyecto.
- Pruebas con la utilización de ruido gaussiano con *standard deviation* aleatorio
- Investigación sobre las transformaciones afines como método para el *data augmentation*.
- Implementación del pipeline.

## Sprint 4

- Pruebas con las transformaciones afines de Keras.

## Sprint 5

- Pruebas con el transformador de Keras.

## Sprint 6 y 7

- Aplicación de *fine tuning* con redes neuronales.

## Sprint 8

- Reconfigurar el pipeline para que el *data augmentation* solo se aplique al conjunto de entrenamiento.
- Prevención de imágenes no segmentadas correctamente para prevenir que entren en el proceso de normalización.



- Incorporación del modelo de *deep learning* al *pipeline*.
- Mejora de la configuración del *pipeline*.

## Sprint 9

- Creación de los modelos de clasificación.

## A.3. Estudio de viabilidad

En este apartado se redacta la viabilidad del proyecto, tanto económica como legal, que potencialmente habría costado el proyecto.

### Viabilidad económica

En primer lugar, se encuentran los cálculos económicos del proyecto, que se pueden dividir en gastos de *software*, *hardware* y de personal.

#### *Software*

Los recursos utilizados para la realización gratuitos y de código abierto, por lo que no se prevé ningún gasto en este apartado.

#### *Hardware*

Para la realización del proyecto, se ha contado con una máquina de la Universidad de Burgos, cuyas características se reflejan en la tabla 4.2. En base a las características, el precio para esta máquina se establece entre los 2.500 y los 3.000 euros, aunque se ha de tener en cuenta que la amortización del mismo va más allá de la realización de este proyecto.

#### **Personal**

Finalmente, se ha de tener en cuenta que el proyecto se ha llevado a cabo por el estudiante y dos tutores.

Considerando que el proyecto se ha llevado a cabo en aproximadamente en 6 meses, se considerará que el estudiante trabajaría a tiempo completo durante ese tiempo, con un sueldo de 1500 euros al mes, cuyo gasto total para la universidad sería de unos 2400 euros al mes. Tabla A.1

Gasto	Cantidad en euros
Base de cotización	1500
Coste seguridad social empresa	898
Total universidad	2398

Tabla A.1: Gastos de contratación del estudiante.

Por su parte, los dos profesores estarían contratados por ese mismo periodo de tiempo, pero trabajando a media jornada., con un salario de 1800 euros al mes por la media jornada, cuyo gasto para la universidad sería de unos 2753 euros al mes por cada tutor. Tabla A.2.

Gasto	Cantidad en euros
Base de cotización	1800
Coste seguridad social empresa	953
Total universidad	2753

Tabla A.2: Gastos de contratación de los tutores (por tutor).

## Gasto total

Por lo tanto, el gasto total se establece en, como se desglosa en la siguiente tabla A.3.

Gasto	Cantidad en euros
<i>Hardware</i>	2500
Contratación estudiante	14400
Contratación tutores	33036
Total	49936

Tabla A.3: Gastos totales del proyecto.

## Viabilidad legal

Como se ha comentado anteriormente, el *software* utilizado es gratuito y libre, y está sujeto a las licencias recogidas en la tabla A.4.

Librería	Versión	Licencia
Anaconda	2.1.1	Libre
Imageio	2.19.3	BSD
Jupyter Notebook	4.11	BSD
Keras	2.7	MIT
Matplotlib	3.1	BSD
Numpy	1.19.5	BSD
OpenCV	4.6	BSD
Pandas	1.4.3	BSD
Python	3.10.5	PFS
Scikit-Learn	1.0	BSD
TensorFlow	2.9.1	Apache

Tabla A.4: Licencias de las librerías utilizadas.

El *dataset* utilizado tiene fines de investigación, pero no se permite la distribución alterada ni la misma y su uso no referenciado.

Finalmente, se han utilizado los iconos de <https://www.flaticon.com/> para la elaboración de las gráficas. La licencia de los mismos no permite comercializarlos o modificarlos.



## Apéndice *B*

---

# Especificación de Requisitos

---

### B.1. Introducción

En este apartado se especifican los objetivos y requisitos del proyecto.

### B.2. Objetivos generales

Los objetivos generales son, el ser capaz de asimilar el trabajo de [2], modificar el código para hacerlo reutilizable y, finalmente, ser capaz de utilizar técnicas de *fine-tuning* con el *dataset* de imágenes sobre una red neuronal ya pre-entrenada.

### B.3. Catalogo de requisitos

Los requisitos del proyecto son los siguientes.

1. Optimizar del código previo.
2. Crear los modelos para la clasificación de imágenes oculares.
  - a) Dividir el *dataset* en entrenamiento y test.
  - b) Aplicar de *data augmentation*.
  - c) Segmentar de las imágenes.
  - d) Normalizar de las imágenes.
3. Aplicar *fine-tuning* a los distintos *datasets*.

4. Crear una configuración donde el usuario pueda modificar los parámetros y visualizar los distintos *outputs*.
5. Determinar qué proceso de *fine-tuning* proporciona una tasa de acierto más alta.

## B.4. Especificación de requisitos

### Optimizar el código previo

Asimilar y optimizar el código previo para que sea funcional. Para llevar a cabo este apartado es necesario contar con el código y el *dataset*, ambos accesibles desde [https://github.com/jaa0124/iris\\_classifier/](https://github.com/jaa0124/iris_classifier/).

### Crear los modelos para la clasificación de imágenes oculares

La clasificación de las imágenes oculares se lleva a cabo utilizando modelos adaptados utilizando distintos *datasets*. Su creación se lleva a cabo en las siguientes fases:

#### Dividir el *dataset* en entrenamiento y test

División del *dataset* de CASIA en entrenamiento y test, para poder validar los modelos. Los requisitos previos son el 1.

#### Aplicar de *data augmentation*

Aplicación de *data augmentation* sobre dos de los modelos. Los requisitos previos son el 1 y el 2a. Este paso solo se aplica a dos de los *datasets*.

#### Segmentar de las imágenes

Segmentación de las imágenes donde el iris será aislado. Los requisitos previos son 1, 2a y 2b. Este paso solo se aplica a dos de los *datasets*.

#### Normalizar de las imágenes

Normalización de las imágenes donde el iris será aislado. Los requisitos previos son 1, 2a, 2b y 2c. Este paso solo se aplica a dos de los *datasets*.

**Aplicar *fine-tuning* a los distintos *datasets***

Aplicación de *fine-tuning* sobre los datasets. Los requisitos previos son Los requisitos previos son 1, 2a, 2b, 2c y 2d. Dependiendo del *dataset*, alguno de los pasos previos puede no ser necesario.

**Crear una configuración donde el usuario pueda modificar los parámetros y visualizar los distintos *outputs***

Creación de una *pipeline* y establecimiento de su configuración para permitir al usuario adaptar el código a sus necesidades. Todos los pasos anteriores son requisitos previos.

**Determinar qué proceso de *fine-tuning* proporciona una tasa de acierto más alta**

Determinación de los valores más altos de la tasa de acierto en los distintos modelos.





## *Apéndice C*

---

# Especificación de diseño

---

### C.1. Introducción

Se detalla el diseño del proyecto.

### C.2. Diseño de datos

#### ***Dataset***

Tal como se ha explicado en la memoria, se ha utilizado el *dataset* CASIA-V1.

La estructura de datos es la siguiente, por persona y sesión. Imagen [C.1](#).

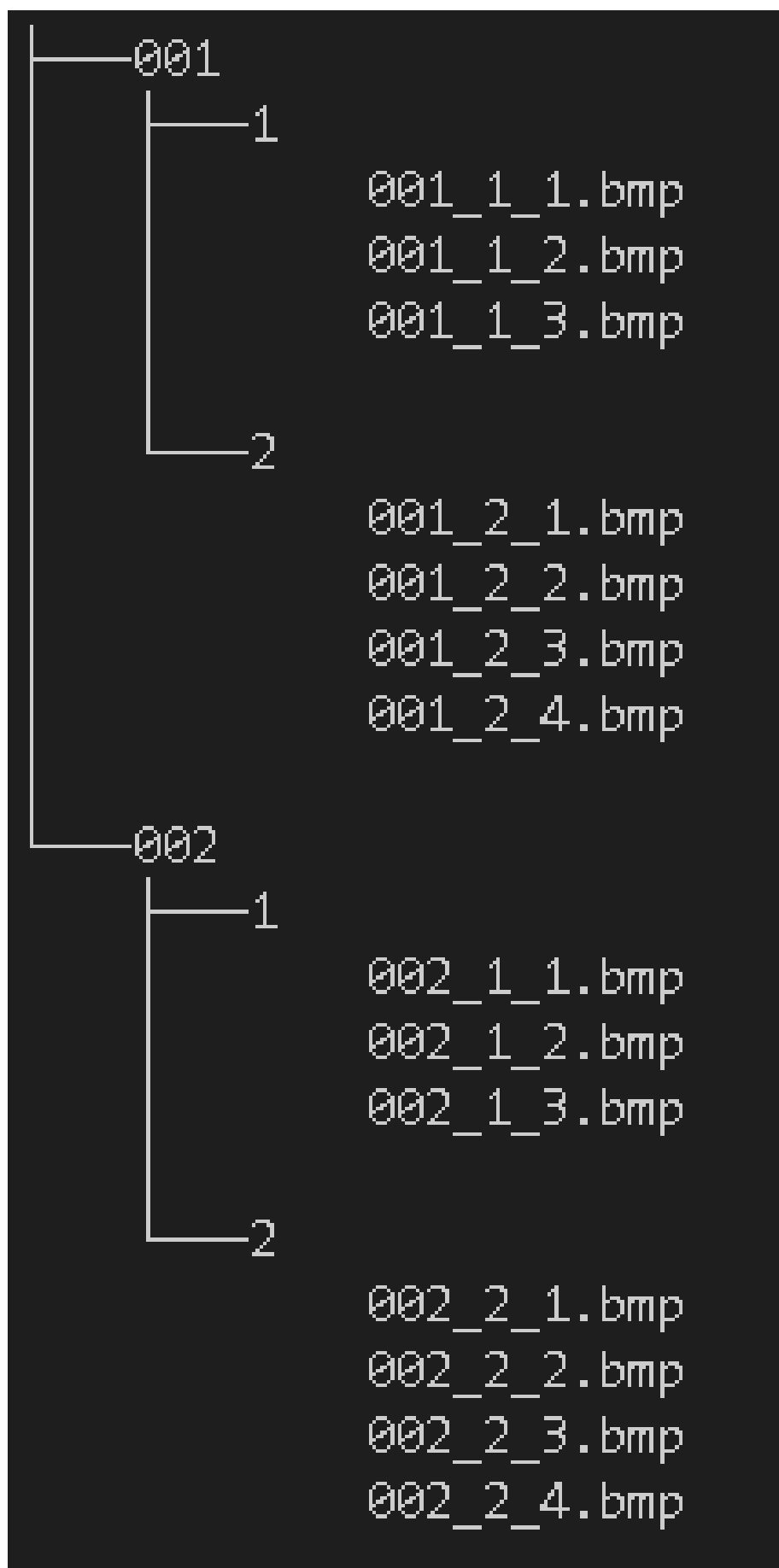


Figura C.1: Estructura de directorios de CASIA-V1.

### ***Notebooks***

El código del proyecto se ha reducido a dos *notebooks*:

- El primero es el, `9-pipelines_v7.ipynb`, en el cual se ha reducido todas las secciones del proyecto, encapsuladas en una *pipeline*.
- El segundo es el, `Accuracy.ipynb`, donde se lleva a cabo el cálculo de la tasa de acierto de los modelos.

## **C.3. Diseño procedimental**

1. Carga de datos
2. *Data augmentation*
3. Segmentación y clasificación
4. *Fine-tuning*
5. Tasa de acierto

## **C.4. Diseño arquitectónico**

Con la utilización de un *pipeline* configurable, se ha intentado que el código sea reutilizable y fácil de manejar.



## *Apéndice D*

---

# Documentación técnica de programación

---

## D.1. Introducción

En este apartado se determinará la forma de proceder para poder replicar el proyecto.

## D.2. Estructura de directorios

- 06\_Models contiene los modelos finales
- img contiene las imágenes utilizadas en el propio repositorio
- memoria, incluye la memoria
  - tex, contiene los archivos tex
  - img, contiene las imágenes del proyecto
- src, incluye el código del proyecto con distintas versiones

## D.3. Manual del programador

Los *notebooks* del proyecto se pueden explorar en cualquier plataforma compatible con Jupyter notebooks.

## **Notebook 9-pipelines\_v7.ipynb**

El *notebook* 9-pipelines\_v7.ipynb contiene el código principal del proyecto. A continuación se detallan sus partes.

### **Configuración**

La configuración del *pipeline* se muestra a continuación. En ella, se establece primero una configuración general, es decir, que es aplicable a las distintas secciones. En ella se han configurado las rutas en las que se guardan los distintos *datasets*. Si todo el código es ejecutado, solo el '*root-dir*' se mantendrá, y el resto se irán sobrescribiendo cuando se cree el propio *dataset*. No obstante, la capacidad de establecer todas las rutas en la configuración es lo que nos permite cambiar el orden de las secciones en el *pipeline*.

- *tratardDataset*
  - *general\_train\_size*, tamaño del *dataset* que será utilizado para el entrenamiento [Valor del 0-1].
  - *show\_first*, mostrar primera imagen [boolean].
- *dataAugmentation*
  - *gaussianNoise*, aplicación del ruido gaussiano [boolean].
  - *stdGN*, valores de ruido gaussiano a aplicar en las imágenes [valores menores de 10].
  - *afinTransformation*, aplicación de transformaciones afines [boolean].
- *segmentation*
  - *redNeuronal*, nombre de la red neuronal a utilizar [extensión .h5].
  - *verImagenV1*, ver la imagen producida [boolean].
- *CNN\_classification*
  - *dataset\_dir*, directorio a utilizar ['raw', para la imagen ocular completa o 'normalizado', para la imagen del iris aislada].
  - *verImagenV1*, ver la imagen producida [boolean].
  - *CNN\_weights*, establecer de donde utilizar los pesos para la red neuronal [imagenet, None, o se le pasa un directorio con los pesos].
  - *train\_size*, tamaño del *dataset* de entrenamiento [0-1].

- *test\_size*, tamaño del *dataset* de testeo [0-1].
- *batch\_size*, tamaño del *batch* [debe de ser menor del tamaño del *dataset*].
- *epochs1*, establecer el tamaño del *epochs* para entrenar al primer modelo [10-100].
- *plt\_accuracy1*, mostrar gráfica con el *accuracy* del primer modelo [boolean].
- *epochs2*, establecer el tamaño del *epochs* para entrenar al segundo modelo [8-80].
- *plt\_accuracy2*, mostrar gráfica con el *accuracy* del segundo modelo [boolean].
- *epochs3*, establecer el tamaño del *epochs* para entrenar al tercer modelo [8-50].
- *plt\_accuracy3*, mostrar gráfica con el *accuracy* del tercer modelo [boolean].
- *results\_array*, mostrar el *array* de resultados [boolean].
- *save\_model*, guardar el modelo [boolean].
- *save\_model\_name*, nombre con el que guardar el modelo [texto].

```
confi_dict = {  
    'general':{  
        'root_dir':r"/home/root_folder",  
        'dataset_dir': "CASIA-IrisV1",  
        'dataset_unif_dir': r"./CASIA-IrisV1_unif",  
        'dataset_unif_dir_aug':r"./CASIA-IrisV1_unif_aug",  
        'dataset_unif_segv2_edg_norm_dir' : r"./CASIA-  
            ↪ IrisV1_unif_segv2_edg_norm",  
        'dataset_unif_dir':r"./CASIA-IrisV1_unif_aug",  
        'dataset_unif_segv2_edg_norm_dir' : r"./CASIA-  
            ↪ IrisV1_unif_aug_segv2_edg_norm",  
        'dataset_unif_dir': r"./CASIA-IrisV1_reservado"  
    },  
    '1_tratarDataset':{  
        'general_train_size': 0.7,  
    },  
}
```

```

        'show_first' : False
    },

    '1.1_dataAugmentation':{
        'gaussianNoise' : True,
        'stdGN': [2.5, 5, 7.5],
        'afinTransformation': True
    },

    '2.1_segmentation':{
        'redNeuronal' : "Iris_unet_d5.h5",
        'verImagenV1' : False
    },

    # Dataset dir can be "normalizado" or "raw"

    '4_CNN_classification' :{
        'dataset_dir' : "normalizado",
        'CNN_weights' : "imagenet",
        'train_size' : 0.7,
        'test_size' : 0.3,
        'batch_size' : 10,
        'epochs1' : 50,
        'plt_accuracy1' : True,
        'epochs2' : 40,
        'plt_accuracy2' : True,
        'epochs3' : 25,
        'plt_accuracy3' : True,
        'results_array' : True,
        'save_model' : False,
        'save_model_name' : "models/normalizado_aug_modelv1"
    }
}

```

## Secciones

Las secciones con las que cuenta el *pipeline* son las siguientes:

1. *tratar\_dataset\_casia*



- Modificación de la configuración de archivos vista en el Anexo [C.2](#) a una configuración de directorio único.

## 2. *data\_augmentation*

- Aplicación de transformaciones afines.
- Aplicación del ruido gaussiano.

## 3. *segmentation*

- Genera las muestras que se le pasaran a la red pre-entrenada.
- Segmenta el iris.

## 4. *normalization*

- Establece los círculos correspondientes a los bordes del ojo.
- Binarización de las imágenes.
- Proyección de la sección del iris.

## 5. *extraction* (parte del código adaptado, no utilizado en este proyecto).

## 6. *clasification* (parte del código adaptado, no utilizado en este proyecto).

## 7. *clasificacionCNN*

- Adaptación del *dataset* al modelo de entrada de la red neuronal.
- División del *dataset* para el entrenamiento y el testeo.
- Construcción de los tres modelos.
- Muestra de resultados y de un ejemplo de clasificación.

## Ejecución del *pipeline*

Cada una de las secciones anteriores ha sido encapsulada en una única función, con la configuración del *pipeline* como único parámetro de entrada. Para ejecutar el *pipeline* se establece cada función dentro de un *FunctionTransformer* y se coloca en el orden requerido dentro de la función *Pipeline*. Finalmente, se ejecuta el *pipeline*, pasándole la configuración como parámetro.

```
_1_tratar_dataset_pip = FunctionTransformer(  
    ↪ tratar_dataset_casia)
```

```

_1_1_data_augmentation_pip = FunctionTransformer(
    ↪ data_augmentation)
_2_1_segmentation_pip = FunctionTransformer(segmentation)
_2_2_normalization_pip = FunctionTransformer(normalization)
_4_clasificationCNN_pip = FunctionTransformer(clasificationCNN
    ↪ )

iris_recognition_pipeline = Pipeline([('_1_tratarDataset',
    ↪ _1_tratar_dataset_pip),
    ('_1_1_dataAugmentation', _1_1_data_augmentation_pip),
    ('_2_1_segmentation', _2_1_segmentation_pip),
    ('_2_2_normalization', _2_2_normalization_pip),
    ('_3_1_extraction', _3_1_extraction_pip),
    ('_3_2_clasification', _3_2_clasification_pip),
    ('_4_clasificationCNN', _4_clasificationCNN_pip) ])

iris_recognition_pipeline.transform(confi_dict)

```

### ***Notebook Accuracy.ipynb***

El *notebook Accuracy.ipynb* permite el cálculo de la tasa de acierto de los modelos, utilizando los modelos creados en el apartado anterior, así como el *dataset* reservado para realizar el testeo.

### **Código**

El código del *notebook* está dispuesto de la siguiente forma:

- *model*, ruta del modelo a utilizar.
- *test\_path\_dir*, ruta del *dataset* de testeo.
- *size* y *\_\_parse\_image*, configuración para adaptar la imagen al formato requerido por el modelo.
- *realClass\_array*, *array* con la clase real de las imágenes.
- *predicted\_array*, *array* con la clase predicha de las imágenes.
- *m.update\_state*, obtiene las dos *arrays* anteriores de entrada y determina la tasa de acierto del modelo.

```

model = keras.models.load_model(r"root_folder/models/
    ↪ normalizado_aug_modelv1")
test_path_dir = r'root_folder/CASIA-
    ↪ IrisV1_unif_aug_segv2_edg_norm'
size = (224, 224)

def _parse_image(filename):
    img = tf.io.read_file(filename)
    img = tf.image.decode_bmp(img, channels = 3)
    img = tf.image.resize(img, size)
    img = tf.expand_dims(img, 0)
    return img

paths_array = np.array([x.__str__() for x in pathlib.Path(
    ↪ test_path_dir).rglob('*.bmp')])
realClass_array = np.array([x.split("/")[-1].split("_")[0] for
    ↪ x in paths_array]).astype(int)
predicted_array = np.array([])
for i in paths_array:
    result = model.predict(_parse_image(i))
    predicted_array = np.append(predicted_array, np.argmax(
        ↪ result) + 1)

m = tf.keras.metrics.Accuracy()
m.update_state(realClass_array, predicted_array)
m.result().numpy()

```

## D.4. Compilación, instalación y ejecución del proyecto

Pasos para la ejecución del proyecto:

- Clonar el repositorio del proyecto <https://github.com/Ponsoda/tfm-iris-recognition>.
- Descargar el *dataset* de CASIA-V1. En el proyecto se ha utilizado el *dataset* a partir de [https://github.com/jaa0124/iris\\_classifier/tree/master/notebooks/CASIA-IrisV1](https://github.com/jaa0124/iris_classifier/tree/master/notebooks/CASIA-IrisV1).

- Abrir el *notebook* `9-pipelines_v7.ipynb`.
- Determinar la ubicación del *dataset* de CASIA-V1 en la configuración del *pipeline*, así como el resto de parámetros y el directorio de salida para los modelos.
- Correr todas las celdas del proyecto en el orden definido en el proyecto para la creación de cada uno de los modelos.
- Abrir el *notebook* `Accuracy.ipynb`.
- Establecer la ubicación del *dataset* reservado para el testeo y del modelo a utilizar y lanzar todas las celdas del *notebook*.

## D.5. Pruebas del sistema

Las diferentes pruebas que se han ido realizando están accesible en el directorio en forma de *notebooks* de Jupyter que pueden ser descargados y ejecutados.

---

## Bibliografía

---

- [1] Dr Hadeel N Abdullah. Iris recognition using wavelet transform and artificial neural networks. page 13.
- [2] Johnson Bolívar Arrobo Acaro. Sistema clasificador de iris.
- [3] Aidan Boyd, Adam Czajka, and Kevin Bowyer. Deep learning-based feature extraction in iris recognition: Use existing models, fine-tune or train from scratch? Number: arXiv:2002.08916.
- [4] Aidan Boyd, Shivangi Yadav, Thomas Swearingen, Andrey Kuehlkamp, Mateusz Trokielewicz, Eric Benjamin, Piotr Maciejewicz, Dennis Chute, Arun Ross, Patrick Flynn, Kevin Bowyer, and Adam Czajka. Post-mortem iris recognition—a survey and assessment of the state of the art. 8:136570–136593.
- [5] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.
- [6] John G. Daugman. High confidence visual recognition of persons by a test of statistical independence. 15(11).
- [7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [8] Guoyang Liu, Weidong Zhou, Lan Tian, Wei Liu, Yingjian Liu, and Hanwen Xu. An efficient and accurate iris recognition algorithm based on a novel condensed 2-ch deep convolutional neural network. 21(11):3721.

- [9] Jus Lozej, Blaz Meden, Vitomir Struc, and Peter Peer. End-to-end iris segmentation using u-net. In *2018 IEEE International Work Conference on Bioinspired Intelligence (IWOBI)*, pages 1–6. IEEE.
- [10] Jasem Rahman Malgheet, Noridayu Bt Manshor, and Lilly Suriani Affendey. Iris recognition development techniques: A comprehensive review. 2021:1–32.
- [11] Shervin Minaee and Amirali Abdolrashidi. DeepIris: Iris recognition using a deep learning approach. Number: arXiv:1907.09380.
- [12] Shervin Minaee, Amirali Abdolrashidi, and Yao Wang. An experimental study of deep convolutional features for iris recognition. Number: arXiv:1702.01334.
- [13] Maciej Szymkowski, Piotr Jasiński, and Khalid Saeed. Iris-based human identity recognition with machine learning methods and discrete fast fourier transform. 17(3):309–317.