

# RECONOCIMIENTO DE IMÁGENES DE MONUMENTOS DEL PATRIMONIO CULTURAL DE MADRID MEDIANTE TÉCNICAS DE APRENDIZAJE PROFUNDO



TRABAJO FIN DE GRADO  
CURSO 2023-2024

AUTORES

FRANCISCO JOSÉ FERNÁNDEZ FERREIRO  
VÍCTOR PORRAS MARTÍNEZ

DIRECTOR

GONZALO PAJARES MARTINSANZ

COLABORADORA EXTERNA

CLARA ISABEL LÓPEZ GONZÁLEZ

GRADO EN INGENIERÍA DEL SOFTWARE  
FACULTAD DE INFORMÁTICA  
UNIVERSIDAD COMPLUTENSE DE MADRID

RECONOCIMIENTO DE IMÁGENES DE MONUMENTOS  
DEL PATRIMONIO CULTURAL DE MADRID MEDIANTE  
TÉCNICAS DE APRENDIZAJE PROFUNDO  
IMAGE RECOGNITION OF MONUMENTS OF MADRID'S  
CULTURAL HERITAGE USING DEEP LEARNING  
TECHNIQUES

TRABAJO DE FIN DE GRADO EN INGENIERÍA DEL SOFTWARE

AUTORES

FRANCISCO JOSÉ FERNÁNDEZ FERREIRO  
VÍCTOR PORRAS MARTÍNEZ

DIRECTOR

GONZALO PAJARES MARTINSANZ

COLABORADORA EXTERNA

CLARA ISABEL LÓPEZ GONZÁLEZ

**CONVOCATORIA: JUNIO**

GRADO EN INGENIERÍA DEL SOFTWARE  
FACULTAD DE INFORMÁTICA  
UNIVERSIDAD COMPLUTENSE DE MADRID

DÍA DE MES DE AÑO



## DEDICATORIA

**Francisco José Fernández Ferreiro**

A mi madre por todo su apoyo en estos  
años de carrera y a los “Bonobos” y los  
“Trolling” por ser el oxígeno en mis  
momentos de ahogo.

**Víctor Porras Martínez**

A mis padres y a mi hermano por su  
incondicional apoyo durante toda mi  
carrera universitaria, consiguiendo que no  
me rindiera.



## **AGRADECIMIENTOS**

A nuestro profesor y director Gonzalo Pajares Martinsanz, así como a Clara Isabel López González por sus esfuerzos y dedicación en brindarnos todo el apoyo necesario y orientación para llevar a cabo este trabajo. Por su paciencia clarificando nuestras dudas y su tarea por hacernos mejorar en todo lo posible.



## **RESUMEN**

En el presente trabajo se desarrolla una aplicación de reconocimiento de monumentos en la ciudad de Madrid mediante el uso de técnicas de reconocimiento de imágenes basadas en aprendizaje profundo.

Para ello se utilizan Redes Neuronales Convolucionales, concretamente AlexNet, GoogLeNet, SqueezeNet e InceptionV3. Dichas redes poseen la capacidad de ser entrenadas con una serie de parámetros para que aprendan características de imágenes de monumentos de la ciudad de Madrid, como parte de su patrimonio cultural. El objetivo es identificar un monumento, mediante el reconocimiento de imágenes basado en las mencionadas técnicas.

Antes de comenzar el entrenamiento el usuario puede configurar una serie de parámetros tales como el algoritmo de optimización, el tamaño del mini lote, el número de épocas o la tasa de aprendizaje. Tras el entrenamiento también se puede realizar una clasificación de imágenes mediante el modelo entrenado para observar los resultados.

Por último, la principal funcionalidad consiste en que una vez que el modelo ha sido entrenado por el usuario según la arquitectura de red y los parámetros elegidos, el modelo puede identificar un monumento a partir de una imagen de este, que constituye la entrada al modelo entrenado.

Adicionalmente, cuando el modelo reconoce un monumento, se le ofrece al usuario un contexto cultural e histórico para que pueda obtener información adicional acerca del monumento reconocido.

### **Palabras clave**

Aprendizaje Profundo, Reconocimiento de imágenes, Redes Neuronales Convolucionales, AlexNet, GoogLeNet, SqueezeNet, InceptionV3, Monumentos patrimonio de Madrid.





## **ABSTRACT**

The objective of this project has been the development of an application for recognizing monuments in Madrid using Deep Learning techniques.

Within the context of Deep Learning, the application makes use of Convolutional Neural Networks (CNN) such as AlexNet, GoogLeNet, SqueezeNet and InceptionV3. These models have the capability to be trained with a set of parameters so that they can learn characteristics from a series of images of monuments of Madrid as part of Madrid's cultural heritage and identify a monument through image recognition.

Before starting the training, users can configure several parameters such as the solver method for optimization, the mini-batch size, the number of epochs, or the learning rate. After training, image classification can also be performed by the model to observe the results.

Finally, the primary use of the application is that once the model has been trained by the user according to the net architecture and the chosen parameters, the model can identify a specific monument by analyzing an input image of it.

Additionally, when a model recognizes a monument, the user is provided with cultural and historical context, allowing them to obtain additional information about the recognized monument.

### **Keywords**

Deep Learning, Image Recognition, Convolutional Neural Networks, AlexNet, GoogLeNet, SqueezeNet, InceptionV3, Heritage monuments of Madrid.

# ÍNDICE DE CONTENIDOS

Capítulo 1 - Introducción .....	1
1.1 Estructuración de la memoria .....	1
1.2 Antecedentes .....	2
1.2.1 Turismo y monumentos .....	3
1.2.2 Avance tecnológico, Aprendizaje Profundo y Redes Neuronales Convolucionales .....	9
1.3 Motivación .....	11
1.4 Objetivos.....	12
1.5 Plan de trabajo .....	13
1.6 Alcance y limitaciones .....	20
Capítulo 2 - Métodos conceptuales y técnicas aplicadas .....	23
2.1 Redes Neuronales Convolucionales (CNN).....	23
2.2 Operaciones en Redes Neuronales Convolucionales .....	23
2.2.1 Métodos de optimización .....	24
2.2.2 Funciones de activación no lineal (ReLU y LReLU) .....	25
2.2.3 Normalización.....	26
2.2.4 Convolución .....	26
2.2.5 Softmax .....	28
2.2.6 Agrupamiento (Pooling) .....	28
2.2.7 Dropout .....	29
2.3 Modelos de Redes Neuronales Convolucionales .....	30
2.3.1 AlexNet.....	30

2.3.2 GoogLeNet .....	32
2.3.3 SqueezeNet .....	34
2.3.4 InceptionV3 .....	35
Capítulo 3 - Diseño y desarrollo de la aplicación.....	41
3.1 Arquitectura .....	41
3.2 Interfaz de usuario .....	42
3.2.1 Fase de entrenamiento .....	44
3.2.2 Fase de clasificación .....	46
3.2.3 Fase de reconocimiento .....	48
3.2.4 Interfaz de ayuda.....	49
3.3 Gestión del proyecto .....	50
3.3.1 Reuniones de planificación .....	51
3.3.2 Reuniones semanales .....	51
3.3.3 Reuniones de retrospectiva .....	51
3.3.4 Tablero de tareas .....	51
3.4 Herramientas y recursos .....	52
3.4.1 Herramientas .....	52
3.4.2 Banco de Imágenes .....	54
Capítulo 4 - Resultados obtenidos .....	57
4.1 Influencia del hardware, datos en el entrenamiento y resultados .....	57
4.2 Entrenamiento y clasificación con GoogLeNet.....	59
4.3 Entrenamiento y clasificación con AlexNet .....	65
4.4 Entrenamiento y clasificación con SqueezeNet.....	69
4.5 Entrenamiento y clasificación con InceptionV3.....	73
4.6 Errores comunes.....	78

Capítulo 5 - Conclusiones y trabajo futuro.....	81
5.1 Conclusiones .....	81
5.2 Trabajo Futuro .....	82
Introduction.....	86
Conclusions and future work .....	97
Contribuciones Personales .....	101
Bibliografía.....	108
Apéndice A - Manual de Usuario.....	112

## ÍNDICE DE FIGURAS

Figura 1-1. Fuente de Cibeles (Fuente Pixabay 2023) .....	4
Figura 1-2 Fuente de Neptuno (Fuente Pixabay 2023) .....	5
Figura 1-3. Estatua del Oso y el Madroño (Fuente Pixabay 2023) .....	6
Figura 1-4. Puerta de Alcalá (Fuente Pixabay 2023) .....	7
Figura 1-5. Puerta de Toledo (Fuente Pixabay 2023) .....	8
Figura 1-6. Estatua de Felipe III (Fuente Openverse 2023) .....	9
Figura 1-7 Logo de la aplicación .....	15
Figura 1-8. Mockup de la página de inicio .....	16
Figura 1-9. Mockup de la página de entrenamiento.....	16
Figura 1-10 Mockup de la página de información y ayuda .....	17
Figura 2-1. Gradiente descendente .....	24
Figura 2-2. Funciones: (a) ReLU; (b) LReLU.....	26
Figura 2-3. Ejemplo de convolución 2-D.....	27
Figura 2-4. Max pooling, no zero-padding (V), $s = 2$ .....	29
Figura 2-5. Max pooling, zero-padding (S), $s = 2$ .....	29
Figura 2-6. Dropout.....	30
Figura 2-7. Modelo de red AlexNet.....	31
Figura 2-8. Módulo Inception con incorporación de unidades ReLU .....	33
Figura 2-9. Modelo completo GoogLeNet (inception V1) .....	34
Figura 2-10. Modulo fire.....	35
Figura 2-11. Red inception-V4 .....	36
Figura 2-12. Módulos: (a) stem; (b) Inception-A; y (c) Reducción-A .....	37
Figura 2-13. Módulos: (a) Inception-B; y (b) Reducción-B .....	38

Figura 2-14. Módulo Inception-C.....	38
Figura 3-1. Diagrama de casos de uso.....	42
Figura 3-2. Vista de la página principal de la aplicación.....	43
Figura 3-3. Vista de la página de Ayuda e información.....	44
Figura 3-4. Vista de la fase de entrenamiento.....	46
Figura 3-5. Pesos primera capa convolucional.....	47
Figura 3-6. Matriz de confusión de validación.....	47
Figura 3-7. Predicción de muestras en fase de clasificación.....	48
Figura 3-8. Vista de la fase de reconocimiento.....	49
Figura 3-9. Interfaz de ayuda.....	50
Figura 4-1. Analizador modelo red GoogLeNet.....	60
Figura 4-2. Ejemplo de imágenes seleccionadas para el entrenamiento.....	60
Figura 4-3. Resumen de entrenamiento óptimo GoogLeNet.....	62
Figura 4-4. Resultados entrenamiento fallido GoogLeNet.....	63
Figura 4-5. Imágenes clasificadas con GoogLeNet.....	64
Figura 4-6. Matriz de confusión para el modelo GoogLeNet.....	65
Figura 4-7. Analizador modelo red AlexNet.....	65
Figura 4-8. Resultado de entrenamiento óptimo AlexNet.....	67
Figura 4-9. Imágenes clasificadas con AlexNet.....	68
Figura 4-10. Matriz de confusión para el modelo AlexNet.....	69
Figura 4-11. Analizador modelo red SqueezeNet.....	70
Figura 4-12. Resultado de entrenamiento óptimo SqueezeNet.....	72
Figura 4-13. Imágenes clasificadas con SqueezeNet.....	73
Figura 4-14. Matriz de confusión para el modelo SqueezeNet.....	73
Figura 4-15. Analizador modelo red InceptionV3.....	74

Figura 4-16. Resultado de entrenamiento óptimo InceptionV3 .....	76
Figura 4-17. Imágenes clasificadas con InceptionV3.....	77
Figura 4-18. Matriz de confusión para el modelo SqueezeNet.....	77





## ÍNDICE DE TABLAS

Tabla 2-1. Parámetros aprendidos por el modelo AlexNet.....	32
---	----

# Capítulo 1 - Introducción

En este primer capítulo introductorio, se comienza explicando cómo se ha estructurado el propio documento, para facilitar al lector su comprensión. Tras la explicación de la estructura, se comentan los fundamentos que han formado la base del trabajo que se ha realizado durante meses. Se analizan y explican los antecedentes, que de alguna forma motivan el trabajo, centrándose en el turismo y los monumentos escogidos que forman parte del bagaje patrimonial de la ciudad de Madrid. Dado que el objetivo es el reconocimiento de imágenes mediante técnicas avanzadas de aprendizaje profundo, también se incide en el avance tecnológico y las metodologías existentes en el ámbito de la Inteligencia Artificial y concretamente dentro del Aprendizaje Profundo (*Deep Learning*) con un papel crucial en el trabajo desarrollado, basado en las Redes Neuronales Convolucionales (*Convolutional Neural Networks, CNN*). También se explica la motivación existente detrás de la idea llevada a cabo, explicando la necesidad de comprender el proceso de reconocimiento de imágenes y su aplicación en el contexto del patrimonio cultural y turístico de la ciudad de Madrid y sus monumentos. De igual modo, se desglosan los diferentes objetivos perseguidos para la realización satisfactoria de este proyecto. Finalmente, se expone detalladamente el plan de trabajo secuencial seguido para el desarrollo del proyecto. Y, por último, se añade un pequeño apartado centrado en el alcance del proyecto y sus limitaciones, junto con una serie de consideraciones a modo de conclusión y trabajo futuro.

## 1.1 Estructuración de la memoria

Este apartado tiene el objetivo de facilitar al lector la comprensión adecuada de la estructura y el enfoque seguido para desarrollar el proyecto y plasmarlo en el presente documento. Durante el desarrollo de esta memoria, se ha tomado como esquema de referencia la plantilla proporcionada por la propia facultad, ciñéndonos a la normativa establecida y buscando realizar adiciones que aporten valor y enriquezcan el documento.

Como se puede ver en el índice de contenidos, la memoria está compuesta por cinco capítulos principales. El primero sirve a modo de introducción para clarificar los antecedentes y el contexto motivacional del trabajo, incluyendo también el plan de trabajo que se ha seguido. El segundo capítulo tiene un enfoque fuertemente teórico centrado en explorar los métodos y técnicas utilizadas en el reconocimiento de imágenes que lleva a cabo la aplicación desarrollada en el ámbito del Aprendizaje Profundo (AP). El tercer capítulo muestra ilustrativamente el diseño de la aplicación, ofreciendo las explicaciones necesarias para comprender su estructura y funcionamiento. En el cuarto capítulo se presentan los resultados obtenidos, realizando un análisis de estos y mostrando los mismos sobre los diferentes modelos de AP usados. Por último, el quinto capítulo consiste en una síntesis de las conclusiones obtenidas y del trabajo futuro.

Adicionalmente se han incorporado diferentes ilustraciones a lo largo de toda la memoria, con el propósito de enriquecer la lectura y comprensión de esta. Además de tener el objetivo de aligerar la lectura, estas ilustraciones buscan ofrecer información adicional que facilite la comprensión de los conceptos.

Asimismo, se ha realizado una subdivisión de algunos apartados en diferentes subapartados con el objetivo de mantener un flujo narrativo, garantizando coherencia y una secuencia lógica a lo largo del documento, que soporta el diseño de la aplicación.

## **1.2 Antecedentes**

A continuación, se exponen los antecedentes necesarios para comprender la relevancia y el contexto de investigación llevados a cabo para la realización de este trabajo. Se comienza por considerar la importancia del turismo en España y en su capital, Madrid, así como el patrimonio cultural del cual forman parte los monumentos elegidos para su reconocimiento a través de esta aplicación. El enfoque elegido tiene su base en el avance tecnológico que se ha vivido en los últimos años y la creciente importancia del AP y dentro de éste de las CNN necesarios para llevar a cabo el reconocimiento de imágenes.

### **1.2.1 Turismo y monumentos**

Tras la crisis vivida por el COVID-19 y su pandemia, los datos estadísticos respecto a los niveles de turismo mundiales en los últimos años indicaban que los niveles de turismo alcanzarían y superarían las cifras anteriores a la pandemia. En 2023, España superó la cifra de 85 millones de visitantes de procedencia internacional, superando en un 1,9% las cifras establecidas en 2019 como referencia prepandemia según La Moncloa (2024). España es un país que suscita un gran interés a todos los viajeros del mundo, ya sea por su clima, su gastronomía, sus paisajes o por los numerosos puntos de interés que tiene a lo largo de toda la península ibérica.

Y dentro del territorio español, el destino más visitado es su capital. Madrid es la ciudad que ha obtenido más visitantes durante el último año, seguida de cerca por Barcelona. Madrid se coloca como la tercera ciudad en el tercer puesto mundial de los destinos urbanos más visitados en el año 2023 (Viajes National Geographic, 2023). Por tanto, Madrid es una ciudad donde el turismo representa un papel fundamental. Sin embargo, es posible que muchos de los visitantes, especialmente los que descubren la ciudad por primera vez, desconozcan gran parte de los monumentos emblemáticos.

En este contexto pensamos en la necesidad de ofrecer una herramienta eficaz que facilite a los turistas identificar dichos monumentos, pudiendo así conocer su nombre oficial y permitiéndoles posteriormente, si lo desean, obtener más información sobre el monumento. Ya que una vez conocido el nombre específico se puede acceder a su información y contexto mediante internet. De esta forma, la aplicación desarrollada ofrece una solución contextual a la necesidad de conocimiento del monumento y acceso a información adicional de interés.

Con el fin de validar la solución conceptual, de entre todos los diferentes monumentos emblemáticos de la ciudad de Madrid, que forman parte de su patrimonio cultural, hemos escogido seis, que nos parecen representativos y llamativos. La aplicación queda abierta, de forma que en un futuro se podrían añadir una gran lista de monumentos existentes en toda la ciudad. A continuación, se presenta un desglose con los seis monumentos escogidos a modo de indicación sobre el contenido de la

información turística que se podría facilitar tras su identificación por la aplicación, y que incluye un pequeño contexto de cada uno de ellos siguiendo la referencia Turismo de la ciudad de Madrid (2024),

➤ Fuente de Cibeles

Diseñada por el arquitecto Ventura Rodríguez, la Fuente de Cibeles (Figura 1-1) fue construida entre 1777 y 1782. Situada en el centro de la plaza que le da su propio nombre, es un monumento icónico de la ciudad. La fuente representa a la diosa Cibeles, símbolo de la Tierra, agricultura y fecundidad. La diosa es tirada en un carro por dos leones, que representan los personajes mitológicos Hipómenes y Atalanta. Esculpidos en mármol y piedra por el escultor francés Roberto Michel. En su inicio la fuente no era un mero símbolo artístico, sino que servía de utilidad para abastecer agua mediante dos caños. Uno de ellos llevaba el agua hasta las casas, mientras que el otro abastecía al público general.

Adicionalmente este es un lugar icónico para los seguidores del equipo de fútbol del Real Madrid, ya que es el lugar dónde se celebran sus títulos, al igual que sucede con la Selección Española de fútbol.

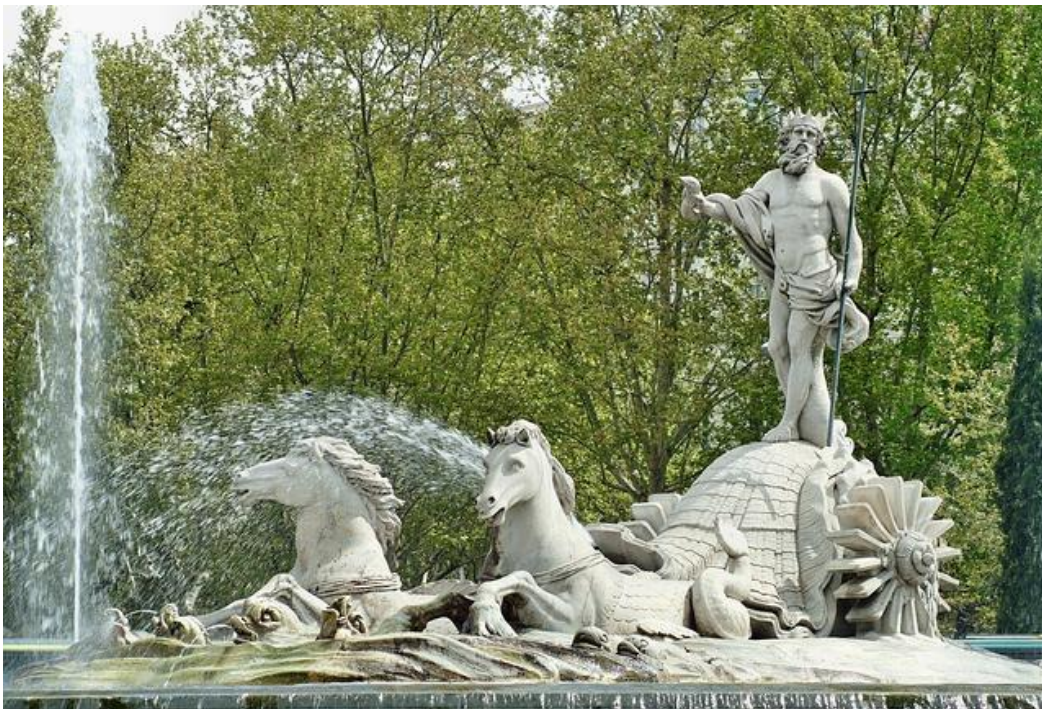


*Figura 1-1. Fuente de Cibeles (Fuente Pixabay 2023)*

➤ Fuente de Neptuno

Diseñada también por el arquitecto Ventura Rodríguez, la fuente de Neptuno (Figura 1-2) fue construida entre 1777 y 1786. Situada en la plaza de Cánovas del Castillo, es junto a Cibeles otra de las fuentes más majestuosas de Madrid. Representando al Dios del mar, Neptuno, sosteniendo su tridente. Sobre una base rocosa surge una carroza con forma de concha, tirada por dos hipocampos, símbolos del mar agitado y las tormentas. Inicialmente esculpida por Juan Pascual de Mena, no pudo acabar la obra debido a su fallecimiento. Y documentos acreditan que la obra fue terminada por su discípulo José Arlas.

Esta obra también es utilizada como un lugar icónico por otro equipo de fútbol de la capital, el Atlético de Madrid, que celebra sus títulos en este lugar emblemático.



*Figura 1-2 Fuente de Neptuno (Fuente Pixabay 2023)*



➤ Estatua del Oso y el Madroño

Esta estatua (Figura 1-3) es obra del escultor español Antonio Navarro Santafé, construida en el año 1967. Situada en una zona emblemática de la capital como es la Puerta del Sol, el lugar donde se encuentra el kilómetro cero de las carreteras radiales del país. Representando de forma realista las armas heráldicas de Madrid con el madroño superando en altura al oso, que apoya sus manos sobre el tronco y dirige sus fauces a uno de los frutos. Esta estatua realizada en piedra y bronce pesa aproximadamente veinte toneladas y alcanza los cuatro metros de altura. Además, posee una base formada por un pedestal cúbico escalonado, realizado en granito.



*Figura 1-3. Estatua del Oso y el Madroño (Fuente Pixabay 2023)*

➤ Puerta de Alcalá

Este monumento (Figura 1-4) es una de las cinco antiguas puertas reales que daban acceso a la ciudad, diseñada y construida por el arquitecto italiano Francesco Sabatini. Su construcción tuvo lugar entre los años 1769 y 1778, por mandato de Carlos



III para sustituir una puerta anterior del siglo XVI. Situada al lado del Parque del Retiro, en ella confluyen las calles de Alcalá, Alfonso XII y Serrano. Consiste en un arco de triunfo de granito de estilo neoclásico, siendo el primero en construirse en Europa tras la caída del Imperio Romano. Las dos fachadas que la componen poseen decoraciones diferentes. Siendo más simple la cara interior que la exterior, que es presidida por el escudo real y posee una mayor riqueza decorativa y que antiguamente era lo que veían aquellos que entraban a Madrid.



*Figura 1-4. Puerta de Alcalá (Fuente Pixabay 2023)*

#### ➤ Puerta de Toledo

Esta puerta (Figura 1-5) fue diseñada por el arquitecto neoclásico español Antonio López Aguado. Construida entre los años 1813 y 1827. Situada en la glorieta Puerta de Toledo, esta es otra de las cinco antiguas puertas reales que daban acceso a la ciudad. La obra sustituyó a otras puertas anteriores del siglo XVI situadas en las proximidades. Su arco de triunfo es de estilo neo-romano y está compuesto por tres arcos. Dos laterales de estructura cuadrada y uno central de medio punto. Contiene vanos flanqueados por medias columnas con capiteles jónicos en el arco central y

pilastras jónicas en los otros dos. Fue erigida como conmemoración de la Guerra de la Independencia española tras la ocupación francesa.



*Figura 1-5. Puerta de Toledo (Fuente Pixabay 2023)*

➤ Estatua de Felipe III

Esta estatua ecuestre (Figura 1-6) fue iniciada por el escultor italiano Juan de Bolonia y finalmente terminada por su discípulo Pietro Tacca. Construida en el año 1616 fue un regalo del gran duque de Florencia para el rey de España. La estatua representa al rey Felipe III con la cabeza visible, vestido con media armadura, mientras que en su pecho cuelga el collar con la Orden del Toisón de Oro. En su mano derecha sostiene el bastón de mando, mientras que con su mano izquierda sostiene las riendas del caballo simbolizando de esta manera su poder sobre las riendas del Estado. También se puede apreciar que el caballo tiene su pata delantera izquierda levantada, otorgando dinamismo a la figura.



Figura 1-6. Estatua de Felipe III (Fuente Openverse 2023)

### **1.2.2 Avance tecnológico, Aprendizaje Profundo y Redes Neuronales Convolucionales**

Desde los orígenes de la informática hasta la actualidad, el avance tecnológico experimentado por esta disciplina ha sido increíble. Cambiando por completo la forma en la que vivimos, estudiamos, trabajamos y nos comunicamos.

Comenzando en la década del 1940 con el inicio de la computación, que marca el prelude de este avance tecnológico sin precedentes, continuando hasta la década del 1960 con la aparición revolucionaria de Internet, de forma que en años posteriores cambió radicalmente la manera en la que accedemos y compartimos información, para desembocar finalmente en la época actual en la que vivimos.

Nos encontramos inmersos en una época donde la Inteligencia Artificial y sus distintas aplicaciones han adquirido una importancia significativa. Desde asistentes virtuales como Google Assistant o Siri, hasta sistemas de recomendación como los algoritmos utilizados por Netflix para recomendar series y películas o los de Amazon para recomendar productos, pasando por el procesamiento de lenguaje natural como Google Translate o los *chatbots* de atención al cliente que usan empresas para resolver

dudas. Y por supuesto, el procesamiento de imágenes como Google Lens o los filtros de realidad aumentada o faciales existentes en las redes sociales. A medida que avanzamos en el tiempo, la tecnología continúa progresando con nosotros, facilitando nuestras vidas y ayudándonos a descubrir nuevas experiencias.

Una de las ramas más importantes de la Inteligencia Artificial es el Aprendizaje Profundo, que es una herramienta fundamental para llevar a cabo el procesamiento de imágenes en distintos ámbitos de trabajo y aplicaciones. Su evolución ha estado marcada por los cambios en los que las máquinas interpretan y comprenden la información visual, abriendo nuevas posibilidades para reconocer patrones y clasificar imágenes.

Diferenciándose de enfoques más tradicionales que dependían en gran parte de la ingeniería manual, el Aprendizaje Profundo permite que el modelo descubra automáticamente las características relevantes de los datos de entrada que se proporcionan. Esta capacidad ha significado una revolución en la manera en la que abordamos el reconocimiento de imágenes, ofreciéndonos un nivel de precisión y generalización inéditos hasta la fecha.

Por último, las Redes Neuronales Convolucionales (CNN) son el epicentro del Aprendizaje Profundo. En lugar de depender de la ingeniería manual, las capas convolucionales de estas redes aplican filtros permitiendo la detección de patrones exactos que permitan reconocer e identificar las imágenes dadas. Inicialmente, y teniendo en cuenta la arquitectura de estos modelos de redes, en las primeras capas se detectan características más simples, mientras que en las siguientes subcapas se combinan para poder reconocer características más complejas. Se ha demostrado su eficacia a la hora de extraer dichas características de las imágenes proporcionadas. Esto se realiza mediante algoritmos de entrenamiento mejorados permitiendo que puedan llevar a cabo la clasificación de imágenes de manera excepcional. Es precisamente en lo que respecta al reconocimiento de imágenes, donde se ubica el presente trabajo para el reconocimiento de monumentos del patrimonio cultural de la ciudad de Madrid.

### 1.3 Motivación

El principal impulso para la investigación y desarrollo de este trabajo surge de la necesidad de comprender el concepto de reconocimiento de imágenes y su funcionamiento. Así como afrontar los desafíos asociados en la identificación y clasificación precisa de monumentos en Madrid.

Primeramente, el contexto turístico explicado anteriormente y su gran importancia en la ciudad donde residimos, con los diferentes monumentos emblemáticos que posee. Es un contexto que brinda la oportunidad de aplicar tecnologías mencionadas anteriormente para facilitar y mejorar la experiencia turística y ayudar a reconocer y saber más sobre estos lugares icónicos.

Nuestro principal interés se centra en la tecnología de la Inteligencia Artificial, su rama del Aprendizaje Profundo y en particular las Redes Neuronales Convolucionales. Marcan la oportunidad de poder aprender, entender y trabajar con estas tecnologías, permitiendo alcanzar nuestro objetivo. Pudiendo descubrir su funcionamiento por el camino, logrando obtener resultados óptimos para abordar el problema de la identificación de monumentos.

Además, la creciente evolución e interés actual por la Inteligencia Artificial que ya se utiliza en diversos campos laborales y cotidianos, subraya la importancia y el impacto social que está teniendo esta tecnología. Nuestro trabajo no tiene el único propósito de resolver una necesidad local específica, sino también entender y contribuir al avance general de la tecnología y la aplicación de la Inteligencia Artificial para el reconocimiento visual de imágenes.

Por tanto, la motivación subyacente al presente trabajo es una fusión que radica en la riqueza cultural de Madrid y la contribución a los visitantes que desconozcan la ciudad. Queriendo formar parte de las tecnologías como el Aprendizaje Profundo y su creciente importancia en la Inteligencia Artificial. Con este trabajo no solo se proporcionará una aplicación para el reconocimiento de monumentos, sino que se abrirá nuevas posibilidades innovadoras para ayudar a los turistas a conocer la riqueza cultural de Madrid, mediante el uso de reconocimiento de imágenes.

## 1.4 Objetivos

Este trabajo tiene como principal objetivo el desarrollo de una aplicación de escritorio que sea utilizada para identificar y clasificar las imágenes de diferentes monumentos de Madrid, que forman parte de su patrimonio. Tiene como finalidad resaltar la riqueza cultural de la ciudad y sus diferentes monumentos emblemáticos y ayudar a aquellas personas que desconocen estos lugares icónicos.

El objetivo es desarrollar una aplicación conceptual que pueda ser usada fácilmente por un usuario promedio y también por uno más experimentado con conocimientos sobre las tecnologías que la aplicación utiliza.

Por ello, parte del objetivo es incluir un manual de usuario (Apartado de Información y Ayuda) que desglose las diferentes funciones de la aplicación y trate de explicar a modo de introducción, los diferentes conceptos y campos que se pueden modificar para realizar el entrenamiento del modelo y la clasificación de las imágenes proporcionadas.

Adicionalmente, una vez que un usuario ha introducido una imagen y la aplicación la haya clasificado, proporcionando el nombre del monumento, se ofrece un contexto histórico y cultural para saciar la curiosidad de los usuarios que tengan interés por los diferentes monumentos.

Para poder entender el objetivo principal de este trabajo, se proporciona a continuación un desglose esquemático que reúne todos los objetivos específicos planteados a la hora de realizar este proyecto:

- Adquirir conocimientos mediante la comprensión de la tecnología de la Inteligencia Artificial y su aplicación mediante el Aprendizaje Profundo.
- Conocer y aprender el funcionamiento de modelos de Aprendizaje Profundo, focalizándose en las CNN y explorando su aplicación en el reconocimiento de imágenes.

- Desarrollar una aplicación minimalista e intuitiva para usuarios con diferentes niveles de experiencia. Desde aquellos con poca familiaridad en las tecnologías, hasta quienes posean nociones más avanzadas.
- Integrar todas las tecnologías funcionales bajo una interfaz amigable, realizando tanto pruebas individuales como de integración.
- Incluir un manual de usuario que permita entender el funcionamiento de la aplicación y permita a los usuarios con menos nociones sobre las tecnologías usadas, aprender un concepto introductorio de ellas y su funcionamiento.
- Experimentar con diferentes modelos de CNN observando sus resultados y comparándolos para poder entender el funcionamiento de cada modelo.
- Ayudar a los usuarios con la identificación de monumentos de Madrid, de manera precisa y eficiente.
- Brindar a los usuarios la posibilidad de conocer el contexto histórico y cultural de los monumentos, fomentando la riqueza cultural de Madrid.
- Generar una documentación detallada del trabajo, que recoja la arquitectura de la aplicación, el proceso de desarrollo, los modelos utilizados y los resultados obtenidos. Facilitando la comprensión del proyecto.

## 1.5 Plan de trabajo

A continuación, se presentan los diferentes pasos seguidos secuencialmente para la realización de este trabajo:

- **Estudio y elección del lenguaje de programación**

Para llevar a cabo el desarrollo de la aplicación se realizó un estudio de dos posibles lenguajes en los cuales realizarlo. Por un lado, Python y por otro Matlab (2023). Ambos permiten la implementación de reconocimiento de imágenes utilizando modelos CNN.

Python, mediante el uso de *TensorFlow*, era una opción atractiva por su amplia comunidad que ofrece distintos recursos. También por su versatilidad en la integración de diferentes tecnologías.

Matlab, mediante el uso de los diferentes módulos que permite como por ejemplo la *Deep Learning Toolbox* o la *Toolbox de Computer Vision*, también resultaba una opción interesante, al llevar más tiempo siendo utilizada y proporcionar facilidades.

Finalmente se optó por utilizar Matlab por su amplia gama de herramientas especializadas, su eficiencia y su rendimiento. Además, nos ofrecía la facilidad para consultar su documentación de ayuda y la opción de diseñar la interfaz gráfica mediante *Matlab App Designer*. Nos parecía una opción robusta, fiable y accesible.

- **Estudio y selección de las Redes Neuronales Convolucionales**

Para poder realizar el procesamiento de imágenes, necesitábamos disponer de modelos de Redes Neuronales Convolucionales. Existen muchos y diferentes tipos de estos modelos de redes, de entre todo el abanico disponible se han escogido las cuatro siguientes: AlexNet, GoogLeNet, SqueezeNet, InceptionV3.

La elección de estas cuatro redes se basa en que son consideradas como algunas de las redes más eficaces y populares en el campo del Aprendizaje Profundo. Por ello, nos parecían opciones sólidas para experimentar y probar. Escogimos cuatro para poder estudiar sus diferentes resultados y comprobar su rendimiento.

- **Búsqueda y selección de imágenes**

Una vez decidimos los monumentos que iban a participar en la aplicación, tuvimos que buscar imágenes suficientes como para poder entrenar los modelos. Para ello recurrimos a una búsqueda en Internet, tratando de seleccionar fotografías de cada monumento. Las fotografías fueron seleccionadas asegurándonos abarcar diferentes ángulos y posiciones, que permitieran al modelo reconocer el monumento desde cualquier perspectiva. Y de este modo conseguir que el usuario final pudiese obtener un reconocimiento exacto sin importar el ángulo de su foto proporcionada. El tamaño de las imágenes no era un factor a tener en cuenta, ya que en el entrenamiento del modelo se realiza un redimensionamiento de las imágenes a tamaños específicos.



Para ello se ha hecho uso de los recursos descritos en la sección 3.4.2, recurriendo a bancos de datos de uso libre, siendo Pixabay (2023). Banco de imágenes. Disponible on-line: <https://pixabay.com/es/> (Accedido Septiembre 2023). y Openverse (2023) los finalmente elegidos.

- **Diseño conceptual de la aplicación**

Antes de comenzar el desarrollo, tomamos un tiempo para reflexionar sobre cómo queríamos estructurar la aplicación. Detallando lo que se quería ofrecer al usuario y la manera en la que se iba a diseñar. Utilizando una pizarra llevamos a cabo una serie de *mockups* que posteriormente digitalizamos para tomar como ejemplo.

Antes de realizar los mockups se decidió darle un nombre a la aplicación que fuese breve pero descriptivo, resultando finalmente "MonuMad", que fusiona las palabras "monumento" y "Madrid" asociadas a su uso. Utilizando la herramienta creativa Adobe Illustrator (2023) se realizó el diseño del logo mostrado en la Figura 1-7. Se optó por un logo con una disposición horizontal y lineal donde la palabra "MonuMad" va seguida de la silueta de un oso y un madroño, elementos icónicos de la ciudad de Madrid. Seleccionamos los colores azul y blanco, que nos parecían colores relajantes que posteriormente nos ayudarían a diseñar la aplicación con un estilo similar.



*Figura 1-7 Logo de la aplicación*

En la Figura 1-8 se muestra el diseño realizado para la página de inicio de la aplicación. Con el título de la aplicación encabezando la página y los distintos botones principales consistentes en un botón para el entrenamiento del modelo, otro para la clasificación de imágenes del modelo y otro para clasificar (descubrir) un monumento. También en la parte inferior, aparece un apartado para que el usuario pueda elegir el modelo de CNN a utilizar. Y un enlace de "información y ayuda" donde se ubica el manual de usuario y la información adicional.

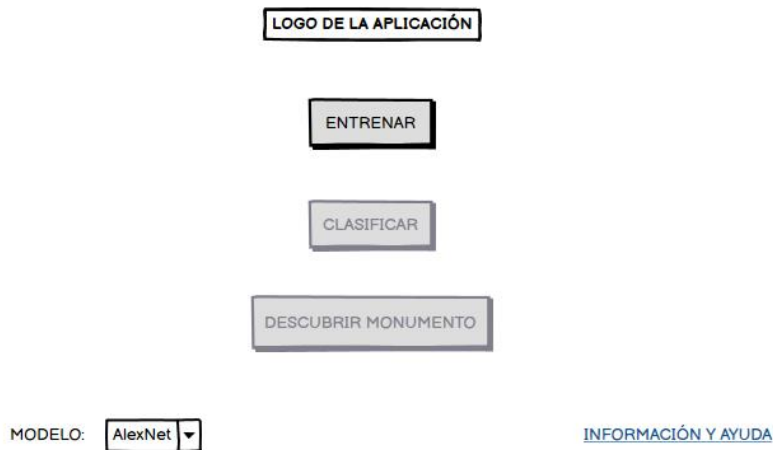


Figura 1-8. Mockup de la página de inicio

La Figura 1-9 refleja el diseño conceptual realizado para la página de entrenamiento de un modelo del tipo CNN. En ella aparecen los distintos parámetros que el usuario puede modificar para llevar a cabo el entrenamiento. Y en la parte inferior aparecen dos botones, uno para entrenar el modelo con los parámetros establecidos y otro para volver a la pantalla de inicio.

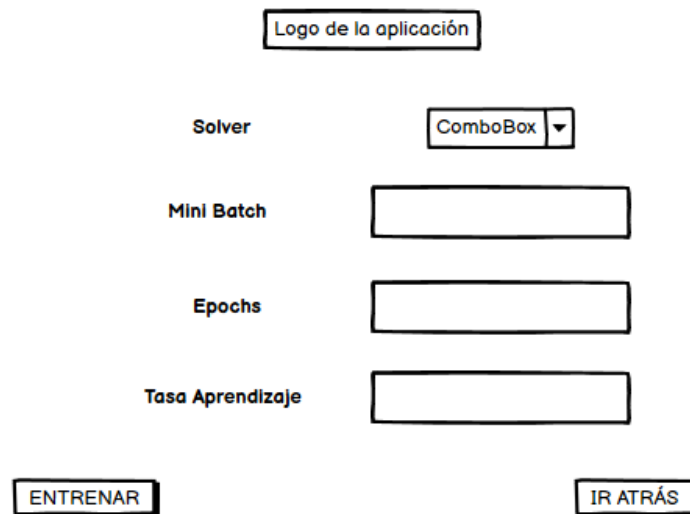
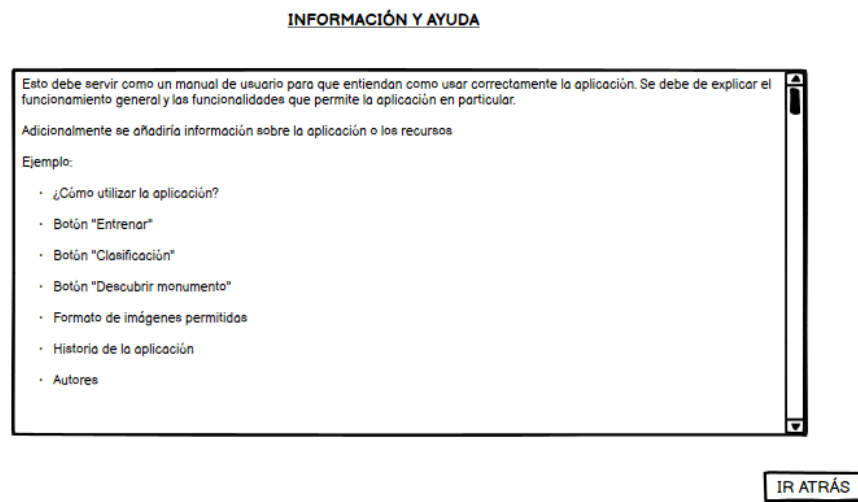


Figura 1-9. Mockup de la página de entrenamiento

Por último, en la Figura 1-10 se muestra el diseño realizado para la página de ayuda. Esta consiste en un texto explicativo con una serie de preguntas que podría plantearse el usuario y las respuestas que se ofrecen. Además, se incluye un apartado con información adicional y explicaciones sobre la aplicación. Esta página sirve para

contener el manual de usuario. También se incluye el botón de “Ir atrás” que permitiría volver a la página de inicio.



*Figura 1-10 Mockup de la página de información y ayuda*

Una vez analizados los elementos constitutivos de la aplicación, se dispuso del diseño en forma presencial. También realizamos anotaciones de las funcionalidades concretas que tendría cada botón. Posteriormente se decidió digitalizarlos usando la herramienta Balsamiq Wireframes (2023), especializada en la creación de *mockups*. De esta manera era factible tener los diseños en los ordenadores propios, de forma clara, para poder usarlos de guía a la hora del desarrollo. Durante el planteamiento de esta fase, se tuvo en cuenta que este diseño podría sufrir pequeños cambios o adiciones que mejorarían la aplicación y que podrían ocurrir más adelante.

- **Desarrollo de los módulos y funcionalidades de la aplicación**

A continuación, se detallan los diferentes módulos más relevantes desarrollados con una breve explicación de sus funcionalidades:

- Redimensionamiento de imágenes

Dado que los modelos CNN necesitan las imágenes en unas dimensiones específicas, se desarrolló un módulo (ConvertirSizesRedes.m) que se encarga de llevar a cabo el redimensionamiento. Obteniendo como resultado las imágenes en las dimensiones 224×224×3 usadas por GoogLeNet, 227×227×3 usadas por

AlexNet y SqueezeNet y  $299 \times 299 \times 3$  usadas por InceptionV3. Todas ellas dispuestas tras el redimensionado en sus respectivas carpetas individuales.

- Entrenamiento de los modelos con parámetros configurables

Antes de realizar la clasificación de imágenes, es necesario entrenar los modelos. Para llevar a cabo el entrenamiento de cada uno de los modelos se desarrollaron tres módulos (Training224.m, Training227.m y Training299.m) que utilizan las imágenes redimensionadas previamente. Estos módulos requieren diferentes parámetros (tipo de optimizador, tamaño del minibatch, número de épocas, tasa de aprendizaje inicial) que son modificables posteriormente mediante la interfaz, si se desea. De este modo se puede adaptar el entrenamiento a demanda del usuario.

- Clasificación de imágenes

Una vez entrenado el modelo, se pueden clasificar las imágenes para identificar una imagen dada ofreciendo los resultados de esta. Para esto se ha desarrollado un módulo (Classify.m) que lleva a cabo este proceso y muestra las diferentes ventanas para ilustrar los distintos resultados.

- Reconocimiento de un monumento

Por último, una vez que el modelo está entrenado, se puede realizar la principal funcionalidad que sería ofrecer una imagen de entrada para que el modelo reconozca dicha imagen e identifique el monumento que contiene. Para ello se desarrolla el módulo (Search.m) que toma la imagen dada por el usuario y la clasifica utilizando el modelo CNN preentrenado, mostrando el monumento contenido en la imagen.

- **Desarrollo de la interfaz gráfica de la aplicación**

Para realizar la interfaz gráfica de la aplicación aprovechamos la herramienta que brinda Matlab (2023) para el diseño de interfaces, llamada Matlab App Designer. Esta herramienta es bastante intuitiva de utilizar. Mediante un panel situado en el lado

izquierdo de la pantalla se pueden seleccionar los diferentes elementos de la página web (botones, imágenes, desplegables, campos numéricos o de texto, etc.). Simplemente deben ser arrastrados hacia el cuadrado central que representa la aplicación. Una vez arrastrado el elemento que se ha seleccionado, asignándole la posición dentro de la aplicación, se pueden modificar ciertos ajustes de los elementos y asignar funcionalidades asociadas.

- **Estudio de los resultados obtenidos**

Una vez desarrollada y operativa la aplicación, se realizaron diferentes pruebas. Probando diferentes combinaciones en los ajustes del entrenamiento de cada red. Lo que permitió analizar el comportamiento con diferentes configuraciones de cada modelo. Esto nos sirvió para llevar a cabo un análisis y comparación de las diferentes CNN, pudiendo entender más su comportamiento y funcionamiento.

- **Búsqueda del contexto histórico y cultural de los monumentos**

Para obtener información acerca del contexto histórico y cultural de los monumentos recurrimos a plataformas oficiales locales especializadas en turismo. Por ejemplo, la página de Turismo de la ciudad de Madrid (2024) ofrece una galería de imágenes con un texto descriptivo. Además, incluye un mapa con la localización del monumento y otros datos de interés como los posibles métodos de transporte cercanos a la zona.

También utilizamos la Plataforma oficial de la comunidad de Madrid (2024) que ofrece una descripción detallada del monumento y su introducción histórica proporcionando el contexto.

Adicionalmente, para ofrecer más información y detalles acudimos a la enciclopedia en línea de Wikipedia. Su acceso gratuito y su modelo colaborativo ofrece información de valor para aquellos usuarios que quieran conocer más detalles acerca del monumento, además de la posibilidad de acceder a otras páginas de información relacionadas con el lugar.

La combinación de estas tres fuentes nos permite brindar al usuario un amplió recurso de datos, asegurando que la información abarcara el contexto histórico, así

como, el cultural. Enriqueciendo de esta manera su experiencia en relación con la aplicación.

- **Desarrollo del manual de usuario**

Por último, una vez finalizada la aplicación, realizamos el apartado de información y ayuda que sirve de manual de usuario. En este apartado se explica el funcionamiento de la aplicación de manera detallada y también se proporciona cierta información acerca de los parámetros modificables en la aplicación y algunos datos de miscelánea.

La sección de información y ayuda incluye las respuestas a las siguientes cuestiones:

- ¿Qué es MonuMad?
- ¿Qué monumentos hay disponibles?
- ¿Cómo usar MonuMad?
- Explicación del botón “Entrenar”
- Explicación del botón “Clasificar”
- Explicación del botón “Identificar”
- Formato de imágenes permitido
- Historia de la aplicación
- Autores de la aplicación

## **1.6 Alcance y limitaciones**

El alcance de la aplicación ha sido limitado en cuanto al número de monumentos considerados (seis), en relación con todos los existentes en la ciudad de Madrid. Se ha escogido la clasificación de monumentos ya que, como se explicó en el apartado de antecedentes, Madrid es una ciudad española con amplia repercusión turística. La elección de únicamente seis monumentos se debió al hecho de ofrecer una

aplicación con carácter de desarrollo conceptual en el campo de la Inteligencia Artificial, primando sobre su extensión a un número mayor de monumentos.

No obstante, es importante destacar que el alcance de la aplicación se podría expandir potencialmente para incluir un catálogo más amplio de los monumentos de Madrid e incluso extenderse a nivel nacional. Sin embargo, para poder lograr eso se requeriría una colección significativa de imágenes para poder entrenar los modelos. Este paso limitaría sustancialmente el tiempo de desarrollo del proyecto, y por ello se decidió abarcar solo los seis monumentos explicados anteriormente para centrarnos en el desarrollo y perfeccionamiento de la aplicación planteada.





## **Capítulo 2 - Métodos conceptuales y técnicas aplicadas**

En este capítulo se explican teóricamente los métodos conceptuales y técnicas aplicadas utilizadas para llevar a cabo este proyecto. Se abordan lo que son las CNN para el reconocimiento de imágenes, las operaciones que realizan y la explicación de los diferentes modelos CNN seleccionados. La base conceptual para explicar estos conceptos se ha tomado del libro Pajares y col. (2021). Este recurso servirá como fundamento para una comprensión sólida y detallada de los conceptos.

### **2.1 Redes Neuronales Convolucionales (CNN)**

Las CNN constituyen el método específico en el desarrollo de este trabajo. Son especialmente efectivas en tareas de reconocimiento de patrones visuales y por ello son útiles para la aplicación desarrollada con el objetivo de reconocimiento de monumentos. Estas redes son capaces de identificar patrones complejos y jerarquías de características en imágenes. Teniendo la capacidad de aprender cuando son entrenadas.

Las CNN son métodos basados en modelos cuyas capas son de naturaleza convolucional. Además, en ellas se realizan diferentes operaciones básicas que se enumeran a continuación en el siguiente punto.

### **2.2 Operaciones en Redes Neuronales Convolucionales**

A continuación se detallan las diferentes operaciones que se realizan en las CNN, proporcionando una explicación sobre ellas y aportando ilustraciones gráficas para facilitar la comprensión de los conceptos.

### 2.2.1 Métodos de optimización

La optimización se refiere a la tarea de minimizar o maximizar una función  $f(x)$  modificando  $x$ . Pero en general la optimización alude a la minimización, ya que la maximización equivale a la minimización de  $-f(x)$ .

La función para minimizar se conoce como la función objetivo o criterio y es utilizada para evaluar una solución candidata, por ejemplo, los pesos en una red neuronal. Cuando se está minimizando, según Goodfellow y col. (2016) se le denomina también función de pérdida (loss function). Esta función es necesaria para poder disponer de un error que retropropagar durante la etapa de aprendizaje. Además, debe ser diferenciable, de modo que mediante dicha función se compara la salida de la red neuronal (predicción) con la etiqueta del objetivo verdadera (ground-truth). Esta función de pérdida es una medida de lo bueno que es un modelo en predecir el resultado esperado.

Cuando se calcula la pérdida, se debe mejorar el modelo. Esto se realiza propagando el error hacia atrás por la estructura del modelo, de modo que en las redes neuronales son los pesos del modelo los que terminan ajustándose.

Se puede pensar la función de pérdida como una montaña ondulada cuya pendiente en descenso es como deslizarse por la montaña para llegar al punto más bajo de la misma. De aquí surge el concepto de optimización mediante el gradiente. Y uno de los métodos más utilizados para encontrar el punto mínimo de la función es el denominado "descenso de gradiente", ilustrado en la Figura 2-1.

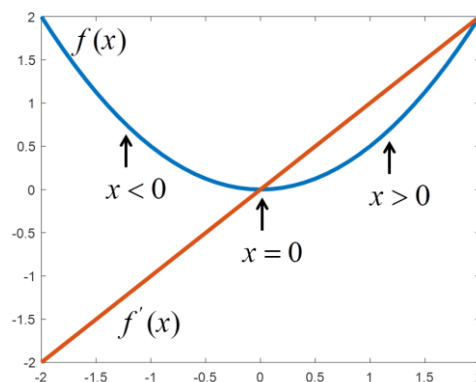


Figura 2-1. Gradiente descendente

### 2.2.2 Funciones de activación no lineal (ReLU y LReLU)

Siendo una función de activación clásica la función sigmoide o sigmoidal definida como sigue:

$$f(a, x, c) = \frac{1}{1 + e^{-a(x-c)}} \quad (2.1)$$

Dependiendo del signo del parámetro  $a$ , la función sigmoide se abre hacia la izquierda o hacia la derecha, siendo apropiada para representar conceptos tales como “muy grande” o “muy negativo”.

La función sigmoide que proyecta salidas de números reales de entrada al intervalo  $[0,1]$  posee dos problemas que afectan a la eficiencia del entrenamiento:

- 1) Saturación del gradiente: el valor de la función de activación se aproxima a los extremos 0 o 1. Entonces el gradiente de la función tiende a 0 y esto repercute en el ajuste de los pesos de las redes.
- 2) Pesos positivos de forma continua: el valor medio de la función de salida no es 0, provocando que los pesos tiendan a ser positivos.

Por esto se introduce la función Unidad Lineal Rectificada (ReLU) que presenta las características de tener el gradiente no saturado y tener una baja complejidad computacional. Aunque soluciona los problemas, esta función presenta la desventaja de que la neurona ReLU puede morir cuando recibe un gradiente negativo alto durante la retropropagación. Esta desventaja puede ser evitada al inicializar cuidadosamente los pesos o utilizar ReLU con “fugas” (LReLU) similar a ReLU pero donde su salida es lineal multiplicada por un valor pequeño. Ambas funciones están ilustradas en la Figura 2-2.

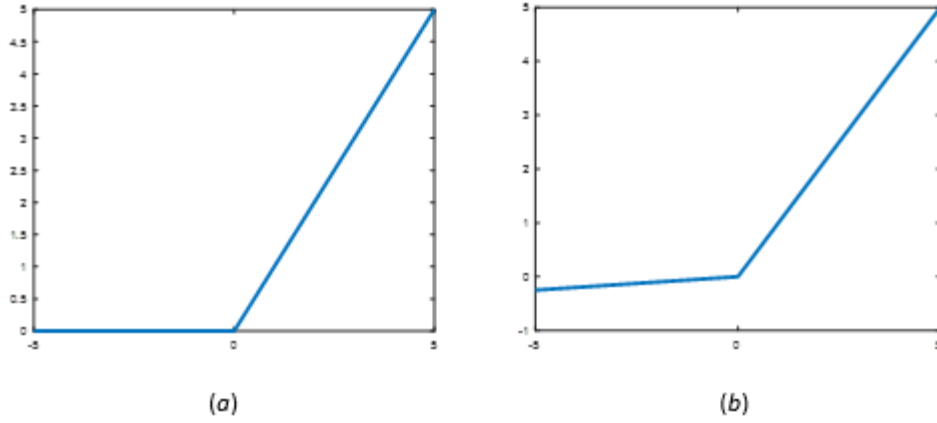


Figura 2-2. Funciones: (a) ReLU; (b) LReLU

### 2.2.3 Normalización

Este proceso ayuda a mejorar el rendimiento de las redes neuronales, utilizándose para equilibrar y mejorar la generalización de las respuestas de las neuronas. Ajustando la actividad de las neuronas para que trabajen de manera más eficiente, evitando problemas tales como la saturación.

Denotándose por  $a_{x,y}^i$  la actividad de una neurona obtenida por aplicación del núcleo  $i$  en la posición  $(x, y)$ . Luego se aplica la no linealidad ReLU, de manera que la actividad de la respuesta normalizada  $b_{x,y}^i$  viene dada por la expresión,

$$b_{x,y}^i = a_{x,y}^i / \left( k + \alpha \sum_{j=\max(0, i-n/2)}^{\min(N-1, i+n/2)} (a_{x,y}^j)^2 \right)^\beta \quad (2.2)$$

donde  $N$  es el número total de núcleos en la capa y las constantes  $k$ ,  $n$ ,  $\alpha$  y  $\beta$  son hiperparámetros cuyos valores se determinan usando un conjunto de validación. En Krizhevsky y col. (2012) se proponen los siguientes valores como los más apropiados  $k = 2$ ,  $n = 5$ ,  $\alpha = 10^{-4}$  y  $\beta = 0,75$ .

### 2.2.4 Convolución

Esta operación involucra dos funciones con valores reales como argumento. Se puede pensar en ello como un proceso de mezcla o promedio ponderado. Por ejemplo,

teniendo una fuente de luz variable y un sensor que mide su intensidad en diferentes momentos, la convolución ayudaría a obtener una señal más limpia y relevante ponderando las lecturas de manera específica.

En el contexto del Aprendizaje Profundo la convolución se utiliza para procesar datos como imágenes, mediante una combinación de una entrada (*input*) con un núcleo (*kernel*) de convolución. Esto ayuda a extraer las características más importantes de los datos y procesarlos eficientemente.

Hay que destacar la relevancia de lo que se conoce como correlación cruzada, expresada en la siguiente expresión 2D como sigue:

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i + m, j + n) K(m, n) \quad (2.3)$$

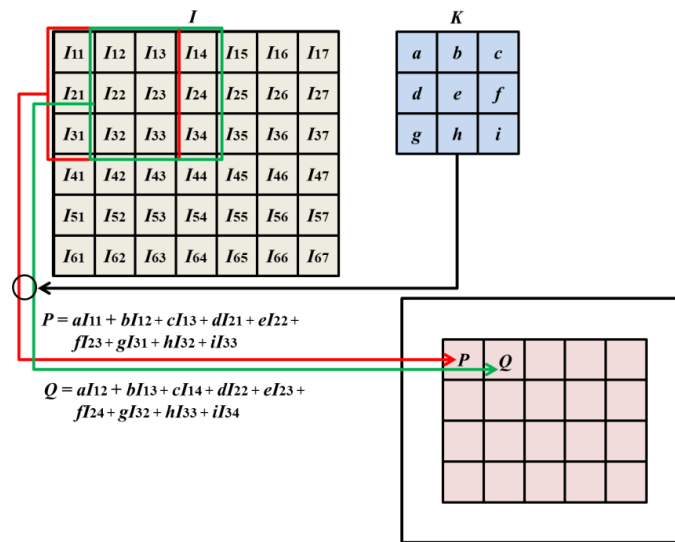


Figura 2-3. Ejemplo de convolución 2-D

La Figura 2-3 muestra un ejemplo de convolución con el núcleo  $K$ , aplicado sobre un tensor 2-D que bien puede representar una imagen  $I$ . La convolución es representada con solapamiento total del núcleo y obteniendo una imagen cuyos bordes externos no tienen valores, obteniendo una imagen de menor dimensión que la original como resultado. Es decir, teniendo una imagen  $I$ , tendríamos el resultado de convolución  $S(i,j)$  para un determinado píxel de la imagen. Dónde  $P$  y  $Q$  serán las salidas generadas después de aplicar el filtro del núcleo  $K$ . No obstante, si se desea obtener una imagen

de la misma dimensión lo que hay que hacer es ampliar la imagen original en dos filas (arriba y abajo) y dos columnas (izquierda y derecha) con ceros, procesándola con el núcleo solapado. A esta operación se la conoce como “relleno con ceros” o *zero-padding*.

Adicionalmente hay que destacar que también se podría realizar con una figura 3-D, de forma que el núcleo sería un cuboide que se desplace a través de las dimensiones alto, ancho y alto del mapa de características. Por ejemplo, teniendo una imagen RGB que posee tres canales para la capa de entrada, se utilizarían núcleos de tipo cuboide que tienen tres dimensiones.

En la Figura 2-3 los desplazamientos del núcleo son la posición a la siguiente, pero en vez de desplazarse a una posición de izquierda a derecha y de arriba a abajo, el desplazamiento podría ser de más de unidad, lo que es conocido como *stride*.

### 2.2.5 Softmax

Esta función *softmax* también conocida como función exponencial normalizada viene dada por la siguiente expresión,

$$softmax(\mathbf{x})_i = \frac{exp(x_i)}{\sum_{j=1}^n exp(x_j)} \text{ para } i = 1, \dots, n \text{ y } \mathbf{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n \quad (2.4)$$

Se utiliza en las últimas capas ocultas de una red neuronal para convertir un vector de valores reales en un vector de probabilidades normalizadas en el rango [0,1]. Cada componente del vector de salida refleja la probabilidad de pertenencia a una clase específica. Con esto la función asigna valores exponenciales y normaliza asegurando que la suma de probabilidades sea igual a 1, lo que facilita la interpretación y la toma de decisiones basada en la salida de la red.

### 2.2.6 Agrupamiento (Pooling)

En las redes neuronales, las capas denominadas de agrupamiento o pooling proporcionan una importante invariancia a pequeñas traslaciones de la entrada. La operación de *pooling* se aplica para reducir la dimensionalidad, de forma que en cada

paso, la posición de la ventana se actualiza de acuerdo con los desplazamientos (*strides*). Existen varias operaciones de este tipo, una de ellas es el máximo (*max*) que consiste en dividir la entrada en ventanas, produciendo como salida el máximo de la ventana. Otra operación es la media (*average, mean*) de la ventana en la que la salida es el resultado de esta operación sobre la ventana. Cuando la ventana se sitúa en los bordes de los elementos, la ventana puede quedar fuera de los elementos de entrada y la entrada se puede extender con valores de cero, lo que previamente se ha denominado como *zero-padding*.

En la Figura 2-4 se muestra un ejemplo ilustrativo de agrupamiento máximo (*max pooling*) sin relleno de ceros (*no zero-padding*) y desplazamiento  $s = 2$ . En este caso la ventana se desplaza de dos en dos posiciones en horizontal y vertical a partir del inicio, pero la columna de la derecha no interviene dado que las ventanas no se pueden solapar sobre ellas. Y en la Figura 2-5 se muestra la misma operación pero utilizando relleno con ceros (*zero-padding*). En este caso la ventana mediante su desplazamiento en saltos de dos en dos ya puede incorporar la columna de la derecha.

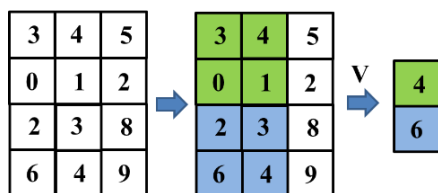


Figura 2-4. Max pooling, no zero-padding (V),  $s = 2$

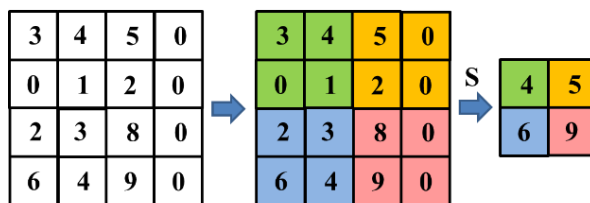


Figura 2-5. Max pooling, zero-padding (S),  $s = 2$

## 2.2.7 Dropout

Este es un mecanismo para paliar el efecto del sobreajuste en las redes neuronales. Consiste en anular determinado tipo de neuronas, incluidas sus salidas, para evitar el mencionado problema de sobreajuste. La selección de las unidades a anular

se realiza de forma aleatoria. De este modo, la red resultante es la que mantiene las unidades que sobreviven al dropout. En la Figura 2-6 se muestran nodos cancelados y con su peso de activación ( $w$ ).

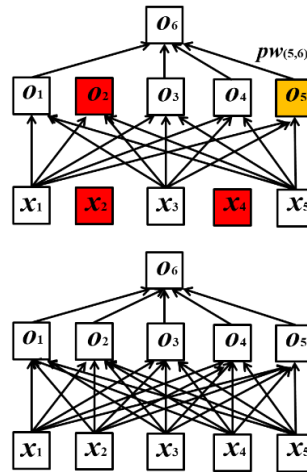


Figura 2-6. Dropout

## 2.3 Modelos de Redes Neuronales Convolucionales

A continuación, se explican y detallan los cuatro modelos elegidos en este proyecto para llevar a cabo el reconocimiento de imágenes, ilustrando el funcionamiento de cada uno de ellos.

### 2.3.1 AlexNet

La red AlexNet (ImageNet, 2020; Russakovsky y col., 2015.; Krizhevsky y col., 2012; BVLC AlexNet Model, 2020) es una de las redes más utilizadas en el ámbito CNN y consiguió ganar la competición de ImageNet LSVRC (2012) en el año 2012.

En la Figura 2-7 se puede observar la estructura de esta red, donde se ilustran las operaciones que se llevan a cabo, entre ellas la de convolución (conv), ReLU (relu), Normalización (norm), Pooling (pool), dropout (drop) o softmax. Además se adjuntan las dimensiones de los filtros, el stride ( $s$ ), el padding ( $p$ ) y el número de filtros ( $K$ ) en cada una de las capas.



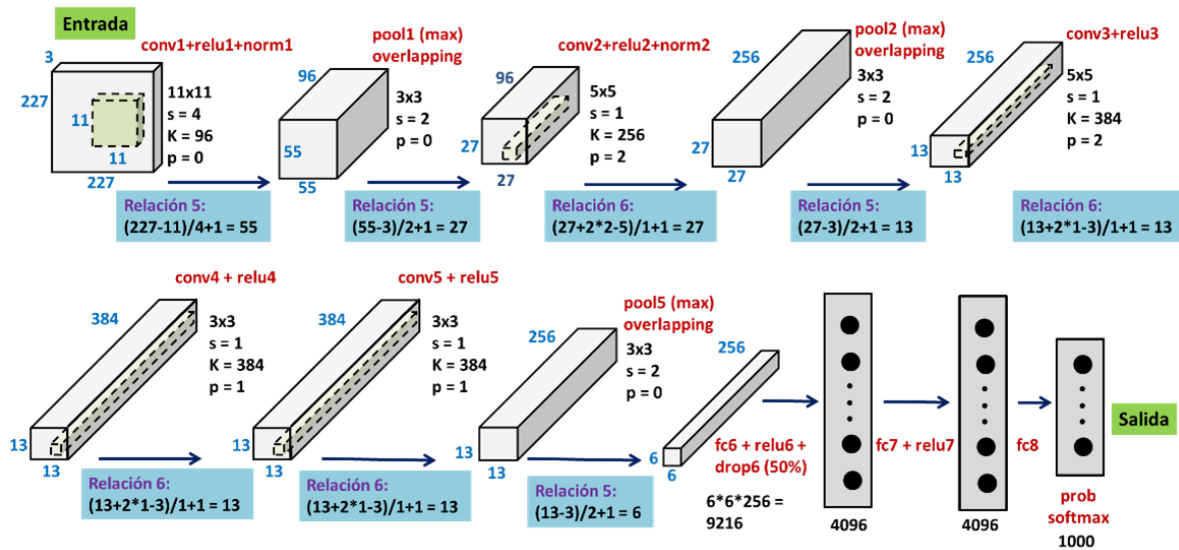


Figura 2-7. Modelo de red AlexNet

En la tercera columna de la Tabla 2-1. Parámetros aprendidos por el modelo AlexNet se muestran los parámetros (pesos) que se aprenden en dicho modelo, correspondiéndose con los pesos en las capas (primera columna) de convolución (conv1 a conv5) y las totalmente conectadas (*Fully connected*, fc6, fc7, fc8) independientemente de que se aplique dropout o no en ellas.

En algunas implementaciones, Matlab (2023), tras la función relu7 se realiza una operación de dropout del 50% cambiando las dimensiones de la salida de esta capa.

Nombre de la capa	Dimensión de salida	Pesos
conv1	55x55x96	W = 11x11x3x96 b = 1x1x96
conv2	27x27x256	W = 5x5x48x256 b = 1x1x256
conv3	13x13x384	W = 3x3x256x384 b = 1x1x384
conv4	13x13x384	W = 3x3x192x384 b = 1x1x384
conv5	13x13x256	W = 3x3x192x256 b = 1x1x256
fc6	1x1x4096	W = 4096x9216 b = 4096x1
fc7	1x1x4096	W = 4096x4096 b = 4096x1
fc8	1x1x4096	W = 1000x4096

		b = 1000x1
--	--	------------

Tabla 2-1. Parámetros aprendidos por el modelo AlexNet

### 2.3.2 GoogLeNet

Esta red fue la ganadora del concurso ImageNet LSVRC 2014 (también llamada como Inception V1) de Google (BVLC GoogLeNet Model 2020), inspirada en el trabajo de Szegedy y col. (2014). La red fue capaz de conseguir hasta una tasa de error de 6,67%, consiguiendo de este modo un resultado muy cercano al nivel humano, haciendo que los organizadores necesitaran la intervención de un experto. La evaluación por parte del experto era bastante difícil de realizar, requiriendo algún entrenamiento adicional para determinar la precisión de GoogLeNet. Pasados unos días, el experto humano (Andrej Karpathy) consiguió una tasa de error entre los 5 primeros del 5,1% (modelo único) y el 3.6% (en conjunto). La red usó una CNN inspirada en LeNet, pero implementó un elemento novedoso llamado módulo *inception*. Se utilizó la normalización por lotes (*batch*), distorsiones de imágenes y RMSProp como método de optimización. El método *inception* (*Inc*) se basa en varias convoluciones relativamente pequeñas que consiguen reducir significativamente el número de parámetros. La estructura del modelo GoogLeNet consiste en una CNN de 22 capas de profundidad, consiguiendo una reducción en el número de parámetros de 60 millones (AlexNet) a 4 millones. De esas 22 capas, 9 son del tipo *Inception* de diferentes categorías. La suma total de capas es de 144 donde cada una de las capas Inception contiene la arquitectura mostrada en la Figura 2-8, en las que se incorpora las unidades ReLU presentes en el módulo.

El modelo entero propuesto por Szegedy y col. (2014) es el mostrado en la Figura 2-9. En él aparecen las distintas capas: convolución (Conv) indicando las dimensiones de los filtros; Pooling, bien *average* (AvgPool) o máximo (MaxPool) con indicación en este caso también de las medidas de la ventana; Normalización (LRN, *Local Response Normalization*); capas completamente conectadas (FC) y por supuesto los módulos *Inception* (*Inc*), en este caso V1 que forman la parte nuclear de este tipo de redes. Además, en las capas de convolución y *Pooling* se señala también el desplazamiento (*stride*) reflejado entre paréntesis con el símbolo s seguido del correspondiente valor de

desplazamiento para el caso 'same' y  $V$  también con el respectivo valor de desplazamiento, en este caso de tipo 'valid'. En el esquema puede verse también dos redes extra, que son clasificadores auxiliares y están compuestos de un *Pooling* promediado de dimensión  $5 \times 5$  y  $s = 3$  de tipo  $V$  que proporciona salidas de tamaños  $4 \times 4 \times 512$  y  $4 \times 4 \times 528$  respectivamente. Se continúa con una capa de convolución  $1 \times 1$  con 128 filtros y una unidad ReLU. Le sigue una unidad dropout con capas totalmente conectadas finalizando con unidades *softmax* (Softmax0 y Softmax1). La salida final de la red es también una unidad *softmax* (Softmax2).

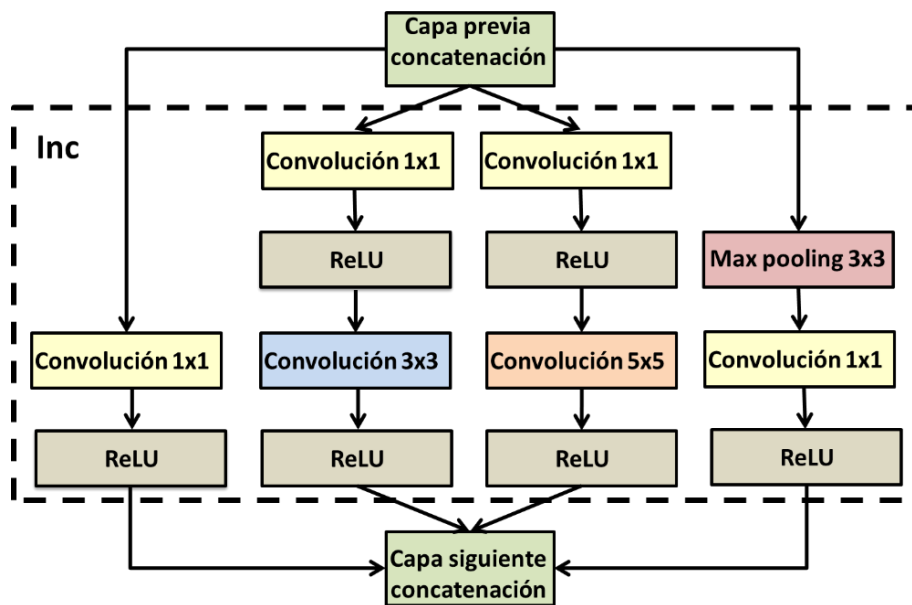


Figura 2-8. Módulo Inception con incorporación de unidades ReLU

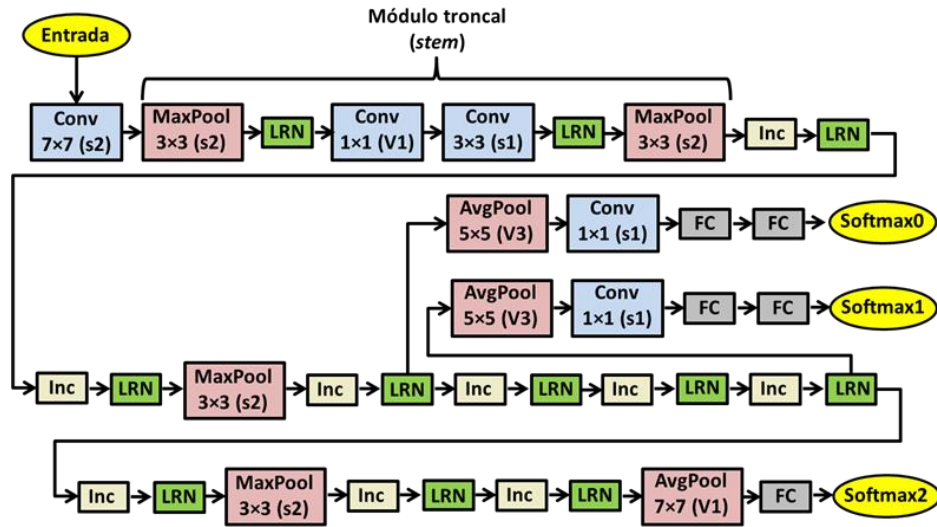


Figura 2-9. Modelo completo GoogLeNet (inception V1)

### 2.3.3 SqueezeNet

Este tipo de redes surge con la idea de conseguir modelos arquitectónicos con menos parámetros que otras redes con arquitecturas similares, consiguiendo resultados de precisión y desempeño similares. Están fundamentadas en lo que se conoce como *fire modules*, que constituyen los elementos base de este tipo de redes. Iandola y col. (2016) proponen tres estrategias para conseguir los objetivos de estas redes en base a su diseño modular:

- Estrategia 1: reemplazar los filtros de dimensión 3x3 con filtros de dimensión 1x1, suponiendo que estos tienen nueve veces menos parámetros que los primeros.
- Estrategia 2: disminuir el número de canales de entrada a los filtros 3x3, cuyos parámetros significan: número de canales de entrada x número de filtros. Por esto, disminuyendo el número de canales de entrada se consigue una reducción en este sentido. Esto se consigue mediante el uso de capas *squeeze*.
- Estrategia 3: submuestreo posterior en la red de modo que las capas de convolución poseen mapas de activación con una elevada dimensión. En una CNN, cada capa de convolución genera mapas de activación con

resoluciones espaciales controladas por factores como la dimensión de los datos de entrada y las capas elegidas para el submuestreo. Este submuestreo se lleva a cabo mediante desplazamientos (stride) mayores a 1, reduciendo las dimensiones de los mapas de activación y optimizando la arquitectura de la CNN.

Es decir, las estrategias 1 y 2 consisten en disminuir el número de parámetros mientras se preserva la precisión. Y mediante la estrategia 3 se trata de incrementar la precisión dado un número limitado de parámetros, para lo cual Landola y col. (2016) proponen el módulo *fire* mostrado en la Figura 2-10. Este módulo consta de una capa de convolución comprimida (squeeze) que tiene solo filtros de dimensión  $1 \times 1$  y una capa expandida (*expand*) con filtros de dimensión  $1 \times 1$  y  $3 \times 3$ .

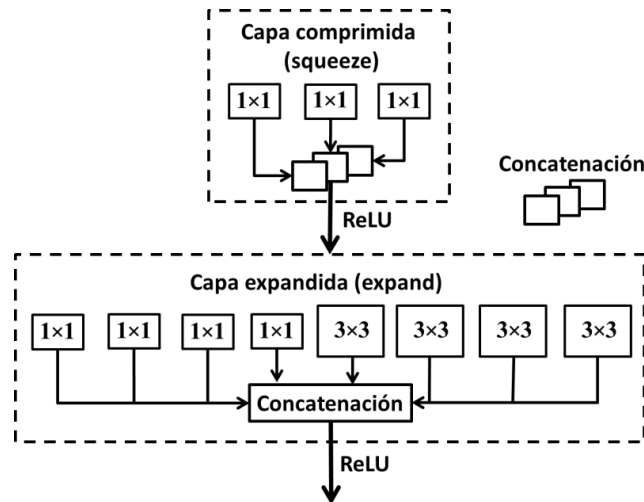


Figura 2-10. Módulo *fire*

### 2.3.4 InceptionV3

Esta red desarrollada por Google utiliza la arquitectura *inception* para las CNN. La arquitectura ha demostrado su eficiencia en diversas tareas de visión por computadora, como clasificación de imágenes y reconocimiento visual. Su diseño se centra en el uso de módulos *inception* que son conjuntos de operaciones de convolución de diferentes tamaños de filtro y agrupamiento.

Para ilustrar esta red se utiliza *inceptionv4* mediante la Figura 2-11, la última versión de esta CNN, que aunque es un poco más avanzada y mejorada sigue utilizando el mismo esquema. En ella se ven conceptos utilizados por las anteriores CNN ya mencionados como *pooling*, *dropout* y *softmax*.

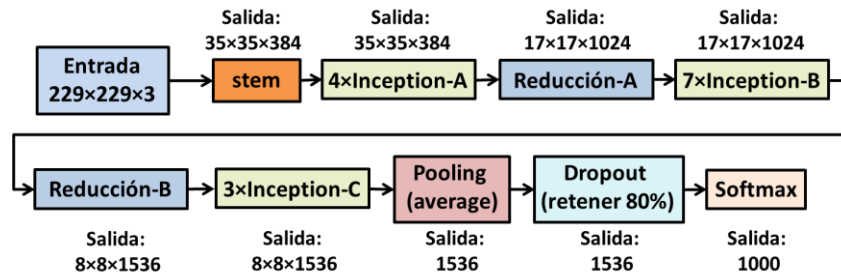


Figura 2-11. Red inception-V4

La Figura 2-12 muestra el diseño del módulo troncal (*stem*) que utiliza esta red, junto al módulo *Inception-A* que proporciona salidas con dimensiones 35x35 y el módulo *Reducción-A*, que pasa de dimensiones 35x35 a 17x17 mediante la asignación de los siguientes valores para el número de filtros en el banco correspondiente:  $k = 192$ ,  $l = 224$ ,  $m = 256$  y  $n = 384$ .

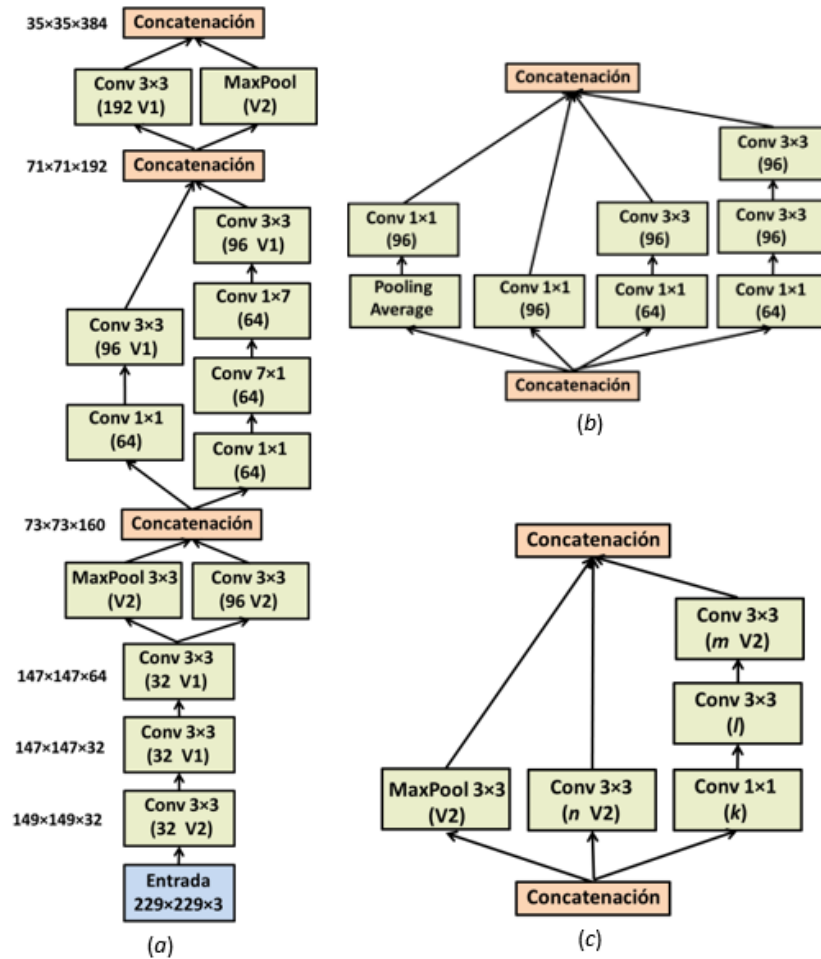


Figura 2-12. Módulos: (a) stem; (b) Inception-A; y (c) Reducción-A

Por último, en la Figura 2-13 se muestran los correspondientes módulos *Inception-B* cuya salida es de dimensiones  $17 \times 17$ , además aparece el módulo *Reducción-B* que pasa de dimensiones  $17 \times 17$  a  $8 \times 8$ . Y en la Figura 2-14 se muestra el módulo *Inception-C* cuya salida es de dimensiones  $8 \times 8$ .

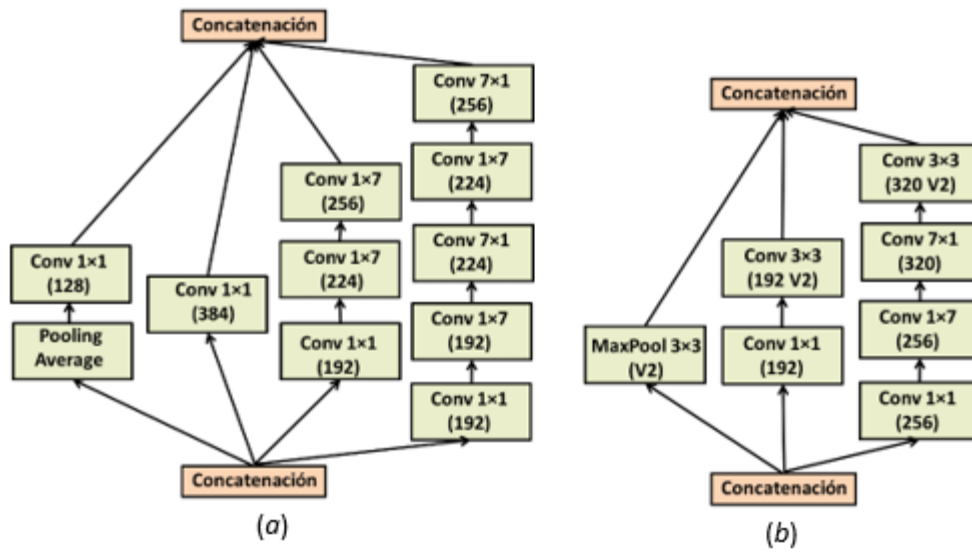


Figura 2-13. Módulos: (a) Inception-B; y (b) Reducción-B

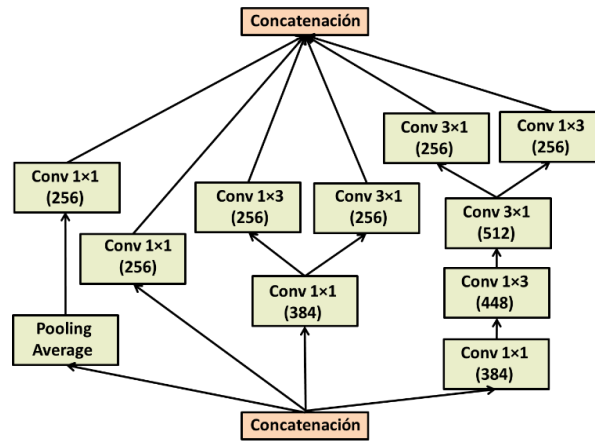


Figura 2-14. Módulo Inception-C







## Capítulo 3 - Diseño y desarrollo de la aplicación

En este capítulo se aborda el diseño y el desarrollo de la aplicación que se ha llevado a cabo, empezando por explicar detalladamente la arquitectura que posee y cómo es su interfaz de usuario. Dentro de la interfaz de usuario aparecen las opciones de entrenamiento y clasificación. Posteriormente se explica cómo se ha realizado la gestión del proyecto, así como las herramientas y recursos utilizados para poder llevar a cabo el desarrollo completo.

### 3.1 Arquitectura

Se ha dividido la arquitectura de la aplicación en dos capas principales: presentación y negocio. Además, se realiza guardado y carga de datos de manera local.

- **Presentación:** en esta capa se incluye un conjunto de menús y botones que permiten al usuario interactuar con la aplicación, así como, modificar los parámetros. Permitiendo de esta manera realizar un amplio uso de la aplicación.
- **Negocio:** es el conjunto de módulos y *scripts* que son llamados mediante la capa de presentación con los datos introducidos por el usuario. Es la encargada de realizar todas las operaciones pertinentes para devolver un correcto resultado.

También, se gestionan los datos que se guardan de manera local en la estructura de carpetas que se ha determinado. Es necesario que haya una gestión de los datos, dado que la aplicación debe poder leer y cargar los modelos entrenados, así como los datos correspondientes a la clasificación. Los datos se guardan en formato *.mat* ya que es un estándar de Matlab y facilita enormemente la posterior carga de estos.

En la Figura 3-1 se muestra un diagrama de casos de uso, donde se representan las diferentes posibles acciones que puede realizar tanto un usuario cualquiera como el administrador de la aplicación. En dicho diagrama se muestran dos roles: el de administrador y el de usuario. El usuario puede hacer uso de todas las funciones de las

que dispone el menú principal y sus submenús, como por ejemplo entrenar nuevos modelos o clasificar una imagen determinada.

Mientras que el administrador podrá realizar todas las acciones del usuario y además, podrá ampliar la aplicación, añadiendo nuevas imágenes y redimensionando las mismas.

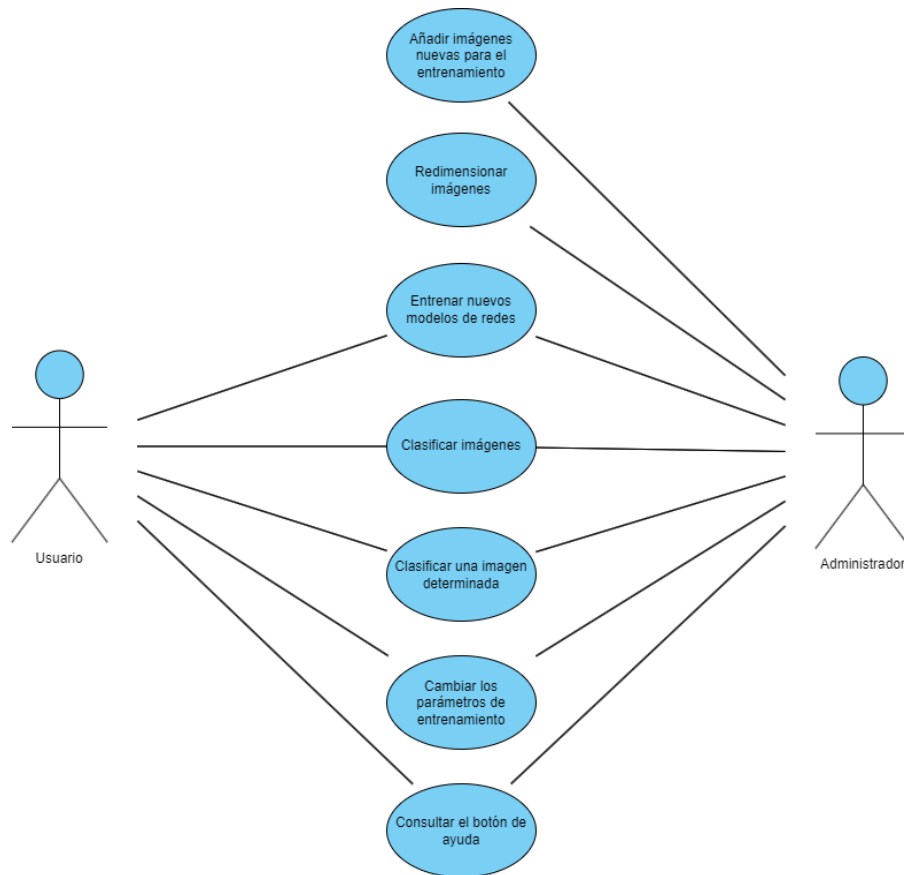


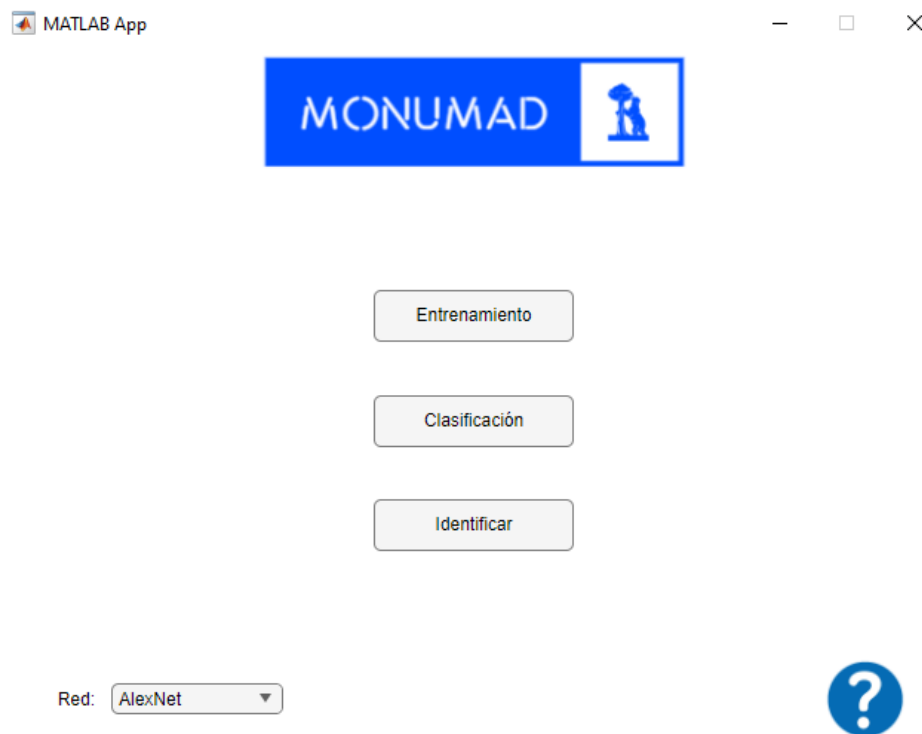
Figura 3-1. Diagrama de casos de uso

## 3.2 Interfaz de usuario

La interfaz principal de usuario que se muestra en la Figura 3-2 que corresponde a la vista inicial que aparece al abrir la aplicación. En la parte superior de la pantalla se puede observar el logo diseñado para la misma, con el nombre de MonuMad. En la parte central se muestra una columna con tres opciones, que consisten en la fase de entrenamiento del modelo representada por el botón de Entrenamiento, la fase de clasificación de imágenes del modelo representada por el botón Clasificar y la fase de

reconocimiento de una propia imagen para identificar un monumento representada por el botón Identificar. En la parte inferior de la pantalla, a la izquierda se ubica el selector del modelo que se quiere utilizar. Este selector muestra los cuatro modelos CNN disponibles: AlexNet, Inception, GoogLeNet y SqueezeNet. Mientras, en la parte inferior derecha se muestra un icono de interrogación, el cual abre la página de ayuda que se puede observar en la Figura 3-3. La cual consiste en una página con diferentes preguntas que forman un manual de usuario que explica el funcionamiento de la aplicación y un pequeño apartado de información adicional.

Hay que destacar que si el modelo no está entrenado previamente las opciones de clasificación y de identificación se mostrarán desactivadas.



*Figura 3-2. Vista de la página principal de la aplicación*

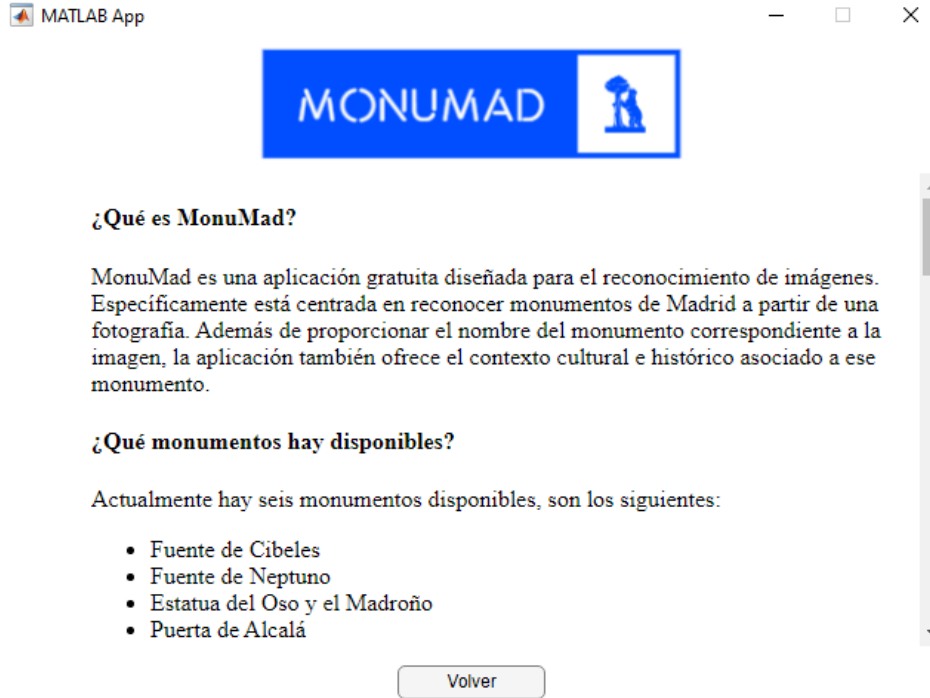


Figura 3-3. Vista de la página de Ayuda e información

### 3.2.1 Fase de entrenamiento

La fase de entrenamiento permite al usuario entrenar el modelo seleccionado utilizando las imágenes predefinidas que ya están redimensionadas listas para ser usadas por el modelo. Estas imágenes se alojan en las carpetas identificadas como DATASET299x299, DATASET227x227 y DATASET224x224, según las dimensiones requeridas por los distintos modelos. Como se muestra en la Figura 3-4, el usuario puede personalizar la fase de entrenamiento mediante dos tipos de opciones. En la parte izquierda de la pantalla se encuentran las opciones de entrenamiento que consisten en:

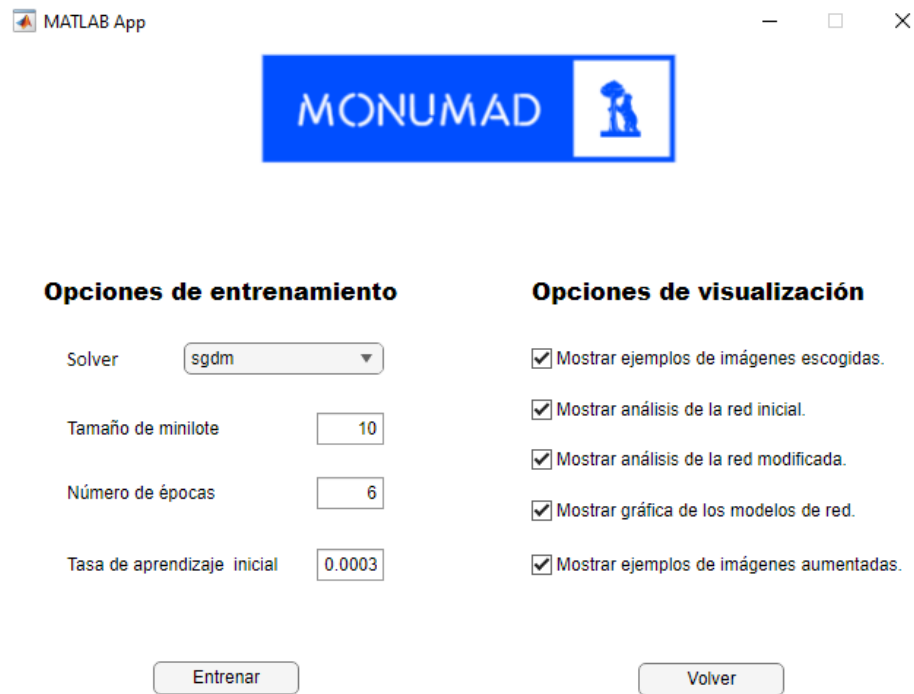
- **Selección del optimizador (solver)**
  - **SGDM (Stochastic Gradient Descent with Momentum):** este optimizador es una variante del descenso de gradiente estocástico que incorpora un momentum, lo que ayuda a acelerar la convergencia del algoritmo de optimización. Utiliza la información

de los gradientes anteriores para actualizar los pesos de la red neuronal.

- **Adam (Adaptive Moment Estimation):** este optimizador combina conceptos del descenso de gradiente estocástico y el método del momento adaptativo. Ajusta la tasa de aprendizaje de forma adaptativa para cada parámetro lo que lo hace especialmente eficaz en problemas con gradientes de gran varianza o ruido.
- **RMSProp (Root Mean Square Propagation):** este algoritmo adapta la tasa de aprendizaje para cada parámetro de forma individual, utilizando una media ponderada de los cuadrados de los gradientes para ajustar la tasa de aprendizaje. Esto hace que sea más estable en ciertos problemas.
- **Tamaño de minilote:** determina la cantidad de muestras que se utilizan en cada iteración durante el entrenamiento. Cuanto mayor sea el tamaño del lote, menos iteraciones se requerirán para completar el proceso de entrenamiento, pero también requerirá más recursos computacionales y memoria lo que puede ralentizar el proceso de entrenamiento.
- **Número de épocas:** es el número de veces en el que todo el conjunto de entrenamiento ha pasado por el modelo. A un mayor número de épocas, el entrenamiento tardará más pero también obtendrá un refinamiento mayor en el ajuste de pesos del modelo, existiendo el riesgo de generar sobreajuste si es demasiado alto.
- **Tasa de aprendizaje inicial:** determina la magnitud de variación de los pesos del modelo durante la optimización para su ajuste.

Por otro lado, en la parte de la derecha de la pantalla el usuario puede elegir entre varias opciones de visualización, para elegir qué pantallas adicionales se abrirán durante el entrenamiento. Por último, en la parte inferior de la pantalla se encuentran dos botones: el botón Entrenar, que inicia el entrenamiento con las opciones

seleccionadas y el botón Volver, que retornará al usuario a la página principal de la aplicación.



The screenshot shows the MATLAB App window for MONUMAD. The title bar includes the MATLAB App icon and window controls. The main header is a blue bar with the text "MONUMAD" and a logo of a person sitting at a desk. Below the header, there are two columns of settings. The left column, titled "Opciones de entrenamiento", contains four settings: "Solver" set to "sgdm", "Tamaño de minilote" set to "10", "Número de épocas" set to "6", and "Tasa de aprendizaje inicial" set to "0.0003". The right column, titled "Opciones de visualización", contains five checked checkboxes: "Mostrar ejemplos de imágenes escogidas.", "Mostrar análisis de la red inicial.", "Mostrar análisis de la red modificada.", "Mostrar gráfica de los modelos de red.", and "Mostrar ejemplos de imágenes aumentadas.". At the bottom, there are two buttons: "Entrenar" on the left and "Volver" on the right.

Opciones de entrenamiento	Opciones de visualización
Solver: sgdm	<input checked="" type="checkbox"/> Mostrar ejemplos de imágenes escogidas.
Tamaño de minilote: 10	<input checked="" type="checkbox"/> Mostrar análisis de la red inicial.
Número de épocas: 6	<input checked="" type="checkbox"/> Mostrar análisis de la red modificada.
Tasa de aprendizaje inicial: 0.0003	<input checked="" type="checkbox"/> Mostrar gráfica de los modelos de red.
	<input checked="" type="checkbox"/> Mostrar ejemplos de imágenes aumentadas.

Entrenar Volver

Figura 3-4. Vista de la fase de entrenamiento

### 3.2.2 Fase de clasificación

Una vez el modelo ha sido entrenado, se puede pulsar el botón “Clasificación” mostrado en la Figura 3-2 para realizar la clasificación de imágenes por parte del modelo. Al hacer clic, se abrirán directamente las pantallas pertenecientes a la fase de clasificación. Mostrándose una pantalla con los pesos aprendidos de la primera capa convolucional que se puede observar en la Figura 3-5.



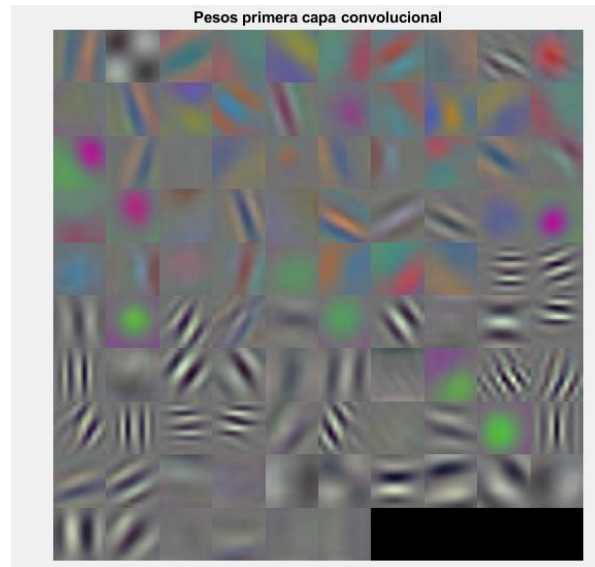


Figura 3-5. Pesos primera capa convolucional

También una pantalla que muestra la matriz de confusión de validación, reflejada en la Figura 3-6. Con ella se puede evaluar el rendimiento de un modelo de clasificación supervisada durante la fase de validación. En cada celda de la matriz se muestra el número de instancias de la clase que fueron clasificadas correcta o incorrectamente por el modelo. Sirve para evaluar la precisión en la fase de clasificación y ajustar el modelo en consecuencia si se desea.

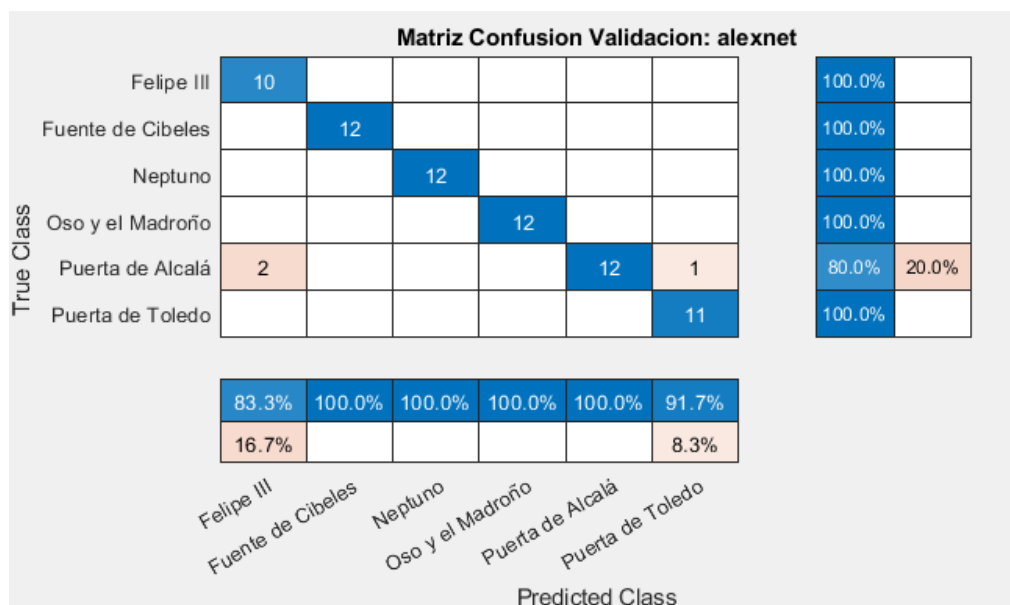


Figura 3-6. Matriz de confusión de validación

Por último, en la Figura 3-7 se muestra la pantalla que representa una muestra de ocho imágenes pertenecientes al *dataset*, las cuales son seleccionadas aleatoriamente durante el entrenamiento junto a la predicción que ha realizado el modelo acerca del monumento identificado. Se añade un porcentaje que refleja la confianza o certeza que el modelo tiene en su predicción.



Figura 3-7. Predicción de muestras en fase de clasificación

### 3.2.3 Fase de reconocimiento

Para llevar a cabo esta fase, es necesario pulsar sobre el botón Identificar. Automáticamente se abrirá el explorador de archivos para que el usuario escoja la imagen a procesar. Una vez que el reconocimiento ha sido llevado a cabo, la pantalla muestra el resultado dado por el modelo como se muestra en la Figura 3-8. La vista muestra a la izquierda la imagen proporcionada por el usuario que ha sido procesada. Y a su derecha se muestra el título del monumento identificado junto a una descripción del contexto histórico y cultural del monumento en cuestión. Adicionalmente, en la parte inferior derecha aparece un enlace a Wikipedia, que proporciona información adicional mediante una página web. Por último, en la parte inferior izquierda se

incorpora nuevamente un botón Volver, para regresar a la página principal de la aplicación.



Figura 3-8. Vista de la fase de reconocimiento

### 3.2.4 Interfaz de ayuda

Para acceder a la interfaz de ayuda, el usuario tiene que pulsar el icono de interrogación mostrado en la interfaz principal, Figura 3-2. Con esto se abrirá la interfaz de la Figura 3-9, que sirve como manual de usuario. En ella se responden a ciertas preguntas que el usuario se puede formular, bien acerca del funcionamiento de la aplicación o sobre conceptos de esta, además de proporcionar cierta información adicional. Las preguntas que incluye esta página de ayuda son las descritas anteriormente en la sección 1.5 del Plan de trabajo. También se incluye el propio botón Volver para regresar a la página principal.

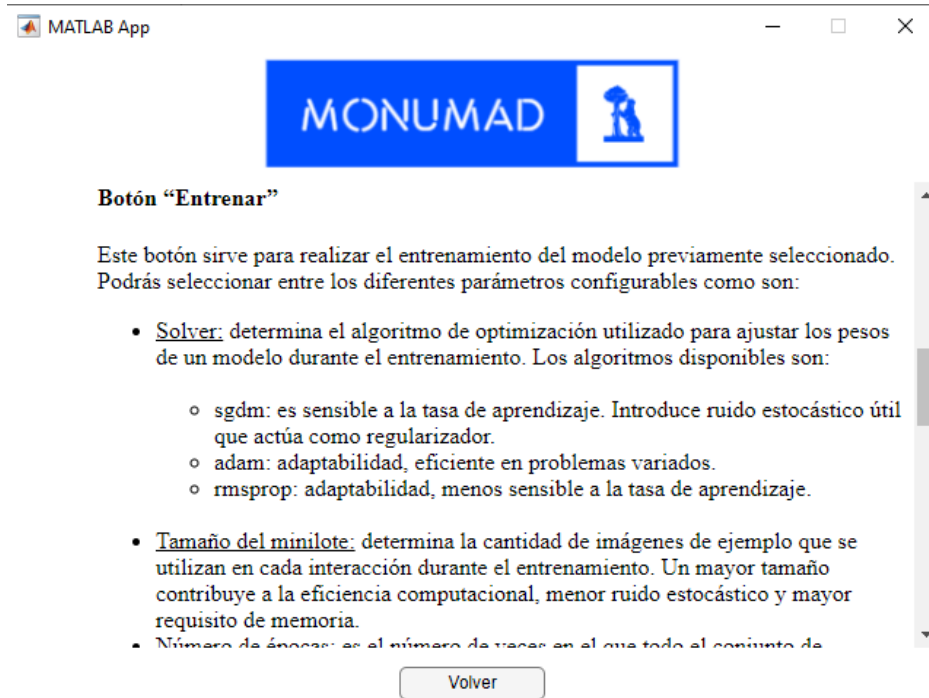


Figura 3-9. Interfaz de ayuda

### 3.3 Gestión del proyecto

Para llevar a cabo el desarrollo de la aplicación, no hemos seguido estrictamente una metodología concreta. Pero sí hemos aplicado ciertos conceptos inspirados en metodologías ágiles como Scrum o Kanban, y que hemos podido aprender en asignaturas realizadas en nuestros estudios previos. Optamos por utilizar estos mecanismos de metodologías ágiles porque favorecían la flexibilidad, colaboración y adaptabilidad durante todo el proyecto.

Realizamos una serie de reuniones que han ayudado a mantener un enfoque dinámico, posibilitando una puesta en común continua y permitiendo que realizáramos ajustes constantes, siempre en contacto y bajo supervisión de los directores del proyecto. Además, podíamos realizar una valoración de nuestros resultados, nuestro ritmo de trabajo y nuestras expectativas para continuar con el desarrollo.

### **3.3.1 Reuniones de planificación**

Estas reuniones se llevaron a cabo antes de iniciar el trabajo sobre diferentes tareas, con el objetivo de construir un plan detallado de las actividades a realizar y establecer ciertas metas a alcanzar en periodos cortos de tiempo, que generalmente eran dos o tres semanas, similar al concepto de *spring* utilizado en Scrum. Durante estas sesiones se establecían los puntos claves que queríamos abordar para ese periodo de trabajo, haciendo un reparto de tareas y proporcionando una guía que seguir para conseguir los objetivos establecidos.

### **3.3.2 Reuniones semanales**

Semanalmente realizábamos reuniones tanto presenciales como online, generalmente los martes de cada semana, con el objetivo de hacer una puesta en común y revisión del trabajo realizado hasta el momento y facilitar el intercambio de ideas. Estas sesiones nos ofrecían la posibilidad de compartir posibles problemas encontrados y debatir los diferentes puntos de vista para obtener una solución. Además, permitían que ambos componentes del equipo estuviésemos informados sobre los avances y próximos pasos a seguir fomentando la transparencia y la cohesión del equipo.

### **3.3.3 Reuniones de retrospectiva**

Una vez concluido el periodo de trabajo definido, se llevaba a cabo una reunión de retrospectiva para revisar si todos los objetivos marcados se habían cumplido y poder evaluar los resultados obtenidos. Durante estas sesiones pudimos exponer nuestro punto de vista sobre cómo había sido la realización del trabajo, así como proponer mejoras en la metodología y sugerir posibles acciones para aumentar la productividad en caso de ser necesario.

### **3.3.4 Tablero de tareas**

Inspirados en el concepto de las tarjetas del tablero Kanban, comenzamos a definir una serie de tareas que tuvieran un tamaño similar en cuanto a la duración de

desarrollo. Estas tareas eran introducidas en un tablero dado utilizando Jira Software (2024). Esto permitía visualizar de manera clara y organizada todas las tareas pendientes de nuestro trabajo y añadir otras nuevas al tablero. De esta manera, durante las reuniones de planificación podíamos seleccionar y asignar las diferentes tareas que se llevarían a cabo en el próximo periodo de trabajo definido. Del mismo modo, podíamos realizar un seguimiento general de cómo era el estado de nuestro proyecto y cómo avanzaba el desarrollo.

### 3.4 Herramientas y recursos

A continuación, se enumeran las diferentes herramientas y recursos utilizados durante todo el desarrollo de la aplicación. Igualmente se proporcionan detalles sobre el banco de imágenes utilizado.

#### 3.4.1 Herramientas

Se han utilizado dos grupos de herramientas, uno para el desarrollo propiamente dicho de la aplicación y otro para el proceso de gestión de esta.

Para el desarrollo de la aplicación se utilizaron:

- **Matlab (2023):** un entorno de programación de alto nivel para el análisis y visualización de datos. Utilizamos esta herramienta para llevar a cabo el desarrollo de nuestra aplicación, en su versión R2023b. Instalando diversos módulos como Image Processing Toolbox, Computer Vision Toolbox, Deep Learning Toolbox que permitían llevar a cabo el reconocimiento de imágenes y el uso de redes neuronales convolucionales.
- **Matlab App Designer:** herramienta de desarrollo gráfico proporcionada por Matlab. Utilizamos esta herramienta para diseñar la interfaz de usuario de la aplicación.
- **Git:** herramienta de control de versiones para el código fuente del proyecto. Se utilizó para gestionar las versiones de nuestro proyecto, manteniendo un historial de cambios.

- **GitHub Desktop:** interfaz gráfica de usuario para Git. Facilita la gestión y la colaboración en proyectos alojados en GitHub. Se utilizó complementariamente junto a Git, ya que permite realizar las mismas operaciones de una manera más sencilla e intuitiva.
- **Adobe Illustrator:** herramienta de diseño vectorial. Utilizada para diseñar el logo de nuestra aplicación.
- **Visual Paradigm:** herramienta de modelado y diseño UML. Fue utilizada para diseñar los diagramas de casos de uso de nuestra aplicación.

Para el proceso de gestión del proyecto se utilizaron las siguientes herramientas y recursos:

- **Discord:** plataforma de comunicación que ofrece chat de texto, voz y vídeo. Fue utilizada para llevar a cabo las reuniones en línea, facilitando la comunicación.
- **WhatsApp:** aplicación de mensajería instantánea que permite comunicación a través de mensajes de texto, voz y multimedia. Ha sido utilizada como una herramienta informal para discutir ideas, planificar, organizar y mantener la comunicación durante todo el proceso.
- **Google Drive:** servicio de almacenamiento en la nube que permite almacenar, compartir y colaborar en diferentes archivos. Ha sido utilizada principalmente para compartir contenido en relación con la memoria, como diferentes versiones, organizar ideas, compartir bibliografía y fuentes.
- **Microsoft Word:** software de procesamiento de texto. Utilizado como herramienta principal para redactar y dar diseño a esta memoria.
- **Gmail:** servicio de correo electrónico que permite enviar y recibir mensajes electrónicos de manera rápida y segura. Ha sido utilizado para enviar información relevante relacionada con nuestro trabajo y para mantener la comunicación con nuestro director del trabajo de fin de grado.

### **3.4.2 Banco de Imágenes**

Uno de los recursos fundamentales, a la vez que necesario es el conjunto de imágenes requerido para validar tanto las fases del proceso, por un lado, el entrenamiento con las imágenes utilizadas para tal fin, junto con las de validación y, por otro lado, las imágenes de test.

Para seleccionar las imágenes en su conjunto se han explorado varias fuentes de datos, comenzando con el banco de datos de imágenes proporcionado por los servicios centrales de la Biblioteca de la Universidad Complutense (Biblioteca UCM, 2023). Dentro de esta página destacan herramientas diversas, entre las que se encuentra el buscador Search Creative Commons (Search CC, 2023), que proporciona imágenes bajo las diferentes modalidades de la mencionada licencia Creative Commons sin propósito comercial.

No obstante, como se ha mencionado previamente, para el desarrollo de este trabajo se ha optado por seleccionar las imágenes proporcionadas por Pixabay (2023), que es un banco de imágenes y vídeos de alta calidad publicados libres de derechos de autor, bajo la licencia Creative Commons CC0 tal y como se establece en los términos del servicio que Pixabay proporciona. Igualmente se ha utilizado Openverse (2023), otro banco de datos que permite descubrir y utilizar fotografías, también con licencia abierta Creative Commons CC0 y por tanto de dominio público según indican sus términos de servicio. En este trabajo se usan las imágenes con carácter docente, no comercial, sin modificar ni adaptar su contenido original, encajando, por tanto, dentro de los términos de la licencia establecidos por Pixabay y Openverse.

Estos bancos de datos contienen miles de diferentes imágenes, si bien, para el propósito que nos ocupa, se han seleccionado 40 imágenes de cada una de las categorías de monumentos utilizados. Con este número relativamente bajo de imágenes, en relación al número disponible, y desde el punto de vista de los modelos CNN, se consiguen resultados satisfactorios, tal y como se ha descrito previamente. Esto es debido a que los modelos utilizados están entrenados con el banco de datos Imagenet (2020), caracterizado por poseer 1.000 clases de objetos y contener 1.281.167



imágenes de entrenamiento, 50.000 imágenes de validación y 100.000 imágenes de prueba, de suerte que los modelos poseen sus pesos ajustados para este conjunto de imágenes, y al utilizar el conjunto propio sólo necesita ajustes relativamente ligeros, en lo que se conoce como transferencia de aprendizaje (*transfer learning*) y por tanto, lo que se está haciendo es utilizar modelos pre-entrenados para ajustarlos al nuevo problema de clasificación de monumentos.



## Capítulo 4 - Resultados obtenidos

En este capítulo se explican las diferentes pruebas realizadas con los cuatro modelos de CNN disponibles en la aplicación, detallando los resultados obtenidos y también los valores óptimos de los diferentes parámetros modificables. Adicionalmente se explica la influencia del hardware, junto con datos sobre estos entrenamientos y algunos errores comunes aparecidos durante el proceso. Conviene reseñar que para la obtención de los resultados se ha utilizado el mencionado *dataset* de forma que como para cada monumento se dispone de una muestra de 40 imágenes, el tamaño total del *dataset* es de 240. Este número de imágenes total se distribuye de forma que el 70% se utiliza para el entrenamiento, el 10% para validación durante el entrenamiento y el 20% restante para el test de clasificación, en todos los casos con la selección aleatoria de las imágenes.

### 4.1 Influencia del hardware, datos en el entrenamiento y resultados

Previo a la explicación de las pruebas y resultados, hay que destacar que el rendimiento y los resultados obtenidos pueden variar dependiendo del hardware utilizado. Primero de todo, es importante remarcar que un modelo puede ofrecer diferentes precisiones en dos entrenamientos con los mismos parámetros usados debido a la aleatoriedad inherente al proceso. Esto es debido a que, durante el entrenamiento, los pesos de la red neuronal se inicializan de manera aleatoria lo que puede afectar a la convergencia y al resultado final. Esto también se puede deber a que las imágenes se distribuyen de forma aleatoria en distintos lotes, siendo estos diferentes en cada proceso de entrenamiento.

Además, el hardware utilizado para el entrenamiento puede impactar significativamente en la eficiencia y velocidad del proceso. Por ejemplo, un computador con más cantidad de RAM y un procesador potente puede manejar las operaciones y cálculos de manera más rápida resultando en un tiempo de entrenamiento más reducido. Por el contrario, si un computador posee menos RAM y un procesador menos potente, debido a la necesidad de procesar datos en lotes más

pequeños o realizar operaciones de manera más lenta, afectará a que el entrenamiento se lleve a cabo en una cantidad mayor de tiempo.

En nuestro caso, durante la realización de las pruebas uno de los dos integrantes del grupo poseía un ordenador más potente que el otro. El computador menos potente poseía 8GB de RAM y un procesador i5-5200U de 2,2GHz del 2015, mientras el ordenador más potente contaba con 16GB de RAM y un procesador AMD Ryzen 5 5625U de 2,3GHz del 2022, es decir, el ordenador más potente contaba con el doble de RAM y un procesador más moderno y con mejores prestaciones y por ello obtenía entrenamientos de manera más eficaz, al poder realizarlos en una menor cantidad de tiempo.

Por último, hay que tener en cuenta otro factor importante, que es el tamaño de los datos, para encontrar los valores óptimos para los diferentes parámetros modificables tales como el tamaño del minilote, el número de épocas o la tasa de aprendizaje. En nuestro caso, al existir 6 monumentos identificables tenemos un total de 6 clases. Como se ha indicado previamente, cada monumento posee una muestra de 40 imágenes, el tamaño total del *dataset* es de 240. Sobre la teoría, esto puede influir de la siguiente manera:

- Tamaño del minilote: si el número de clases es grande, puede requerirse un mayor tamaño de minilote para abarcar la diversidad de clases y reducir la varianza en el gradiente. Igualmente, si el *dataset* es grande permite que el tamaño del minilote sea mayor sin comprometer la variabilidad de los datos. Sin embargo, con un *dataset* pequeño es posible que haya que reducir el tamaño del minilote para evitar el sobreajuste.
- Número de épocas: con un mayor número de clases y un tamaño de *dataset* más grande es posible que se necesite un mayor número de épocas para que el modelo aprenda las características distintivas de cada clase de manera efectiva, pudiendo así capturar las complejidades y variaciones de los datos. Por el contrario, un *dataset* de menor tamaño puede facilitar que el modelo converja más rápidamente implicando menos épocas.

- Tasa de aprendizaje: un mayor número de clases puede requerir ajustes más cuidadosos en la tasa de aprendizaje para garantizar la convergencia. Y con un tamaño de *dataset* grande una tasa de aprendizaje alta puede provocar oscilaciones o un sobreajuste más fácilmente.

## 4.2 Entrenamiento y clasificación con GoogLeNet

Antes de que se inicie el entrenamiento, la aplicación permite abrir una serie de ventanas durante el proceso. Como por ejemplo el analizador del modelo de la red seleccionada en su estado inicial o modificada y un conjunto de imágenes seleccionadas aleatoriamente del *dataset* que serán utilizadas en el entrenamiento. También un ejemplo de imágenes aumentadas o una gráfica de los modelos de red.

En la Figura 4-1 se puede observar el analizador del modelo correspondiente a la red GoogLeNet antes del entrenamiento. Se muestran las diferentes capas junto con sus conexiones. El número de capas (*layers*) en este caso es 144. También muestra el número de pesos a aprender por el modelo (*total learnables*) en su caso 5.9 millones. Adicionalmente se muestra el número de avisos (*warnings*) y errores (*errors*) que en este caso es 0, indicando que el modelo está listo para iniciar el proceso de entrenamiento.

Matlab además ofrece dos figuras consistentes en un conjunto de imágenes seleccionadas aleatoriamente del *dataset*, que serán utilizadas para el entrenamiento. En la Figura 4-2 se puede observar una de estas figuras, constando de dieciséis imágenes sacadas del *dataset*, que mezclan diferentes monumentos. Estas imágenes seleccionadas participarán en el ajuste de pesos durante el entrenamiento.

# Analysis for trainNetwork usage

Name: lgraph

Analysis date: 17-Apr-2024 12:22:30

5.9M  
total learnables

144  
layers

0 warnings

0 errors

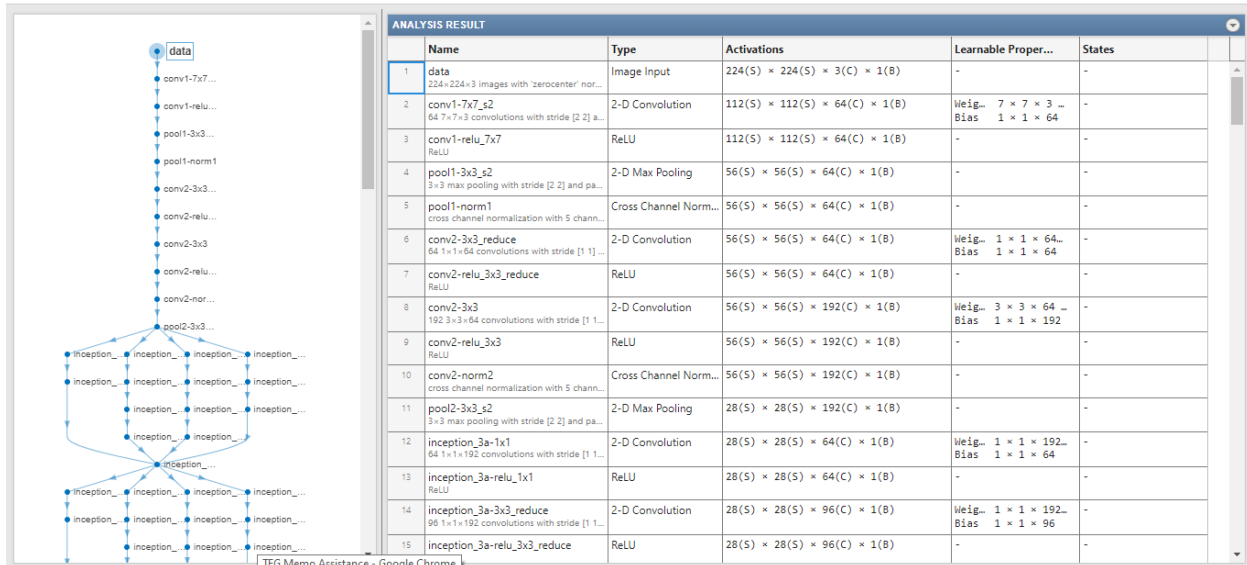


Figura 4-1. Analizador modelo red GoogLeNet

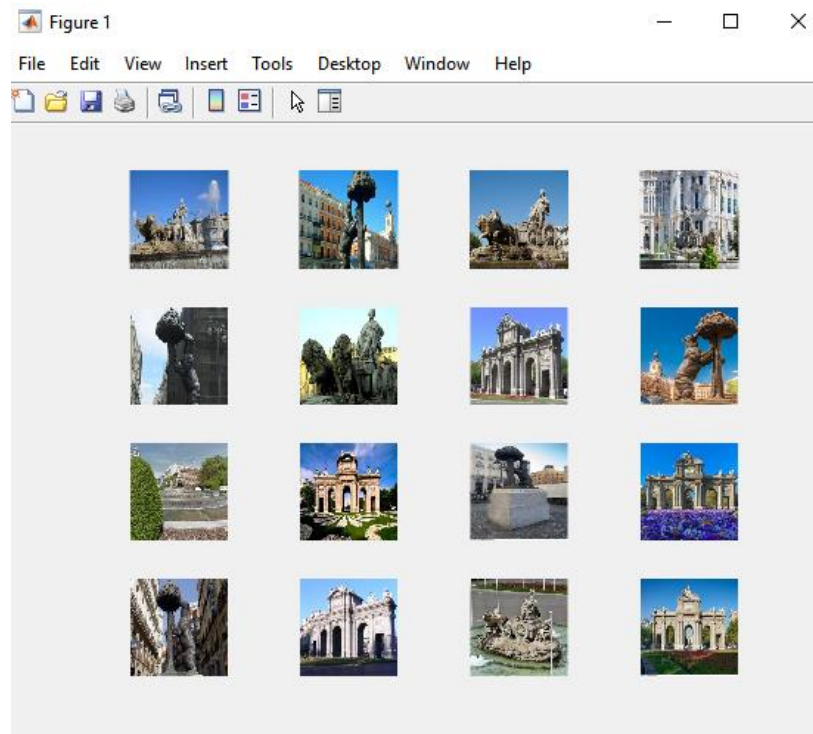


Figura 4-2. Ejemplo de imágenes seleccionadas para el entrenamiento

Tras diversas pruebas realizadas, encontramos que la mejor combinación de parámetros para obtener unos resultados óptimos es la siguiente:

- Tamaño del minilote: 16
- Número de épocas: 4
- Tasa de aprendizaje: 0,001 (para el optimizador SGDM), 0,0001 (para los optimizadores adam y RMSProp)

En la Figura 4-3 se muestra el resumen del entrenamiento con la combinación de parámetros explicada anteriormente. En este entrenamiento se alcanzó una precisión de validación del 97,22%. En las cuatro épocas se realizaron un total de 40 iteraciones. El proceso de entrenamiento duró 4 minutos y 9 segundos, utilizando el computador menos favorable de los dos integrantes. En el gráfico superior marcado en azul se puede observar cómo durante la primera época se produce una gran mejora en la curva de aprendizaje, posteriormente en la segunda época se produce una pequeña bajada que es corregida en las dos últimas épocas alcanzando el pico máximo en el final de la cuarta época. Durante todo el entrenamiento se puede observar cómo el porcentaje de precisión de validación no cae por debajo del 90% una vez que ha transcurrido la primera época. En otros entrenamientos con estos parámetros se ha llegado a conseguir una precisión del 100%, que sería el caso más ideal posible; no obstante, tras diferentes pruebas, los entrenamientos con estos parámetros rara vez bajan del 90% de precisión, lo cual nos indica que son entrenamientos muy satisfactorios que cumplen con las expectativas. Hay que tener en cuenta que el mejor ajuste se correspondería con valores del 100%. Por otra parte, conviene señalar que se muestra la evolución de dos gráficas, una para el entrenamiento (azul continua *smoothed*) y otra para la validación (negra discontinua), observándose que ambas progresan en paralelo, sin distanciarse grandemente entre sí, lo que significa que el modelo está generalizando bien, es decir el ajuste permite clasificar imágenes no vistas.

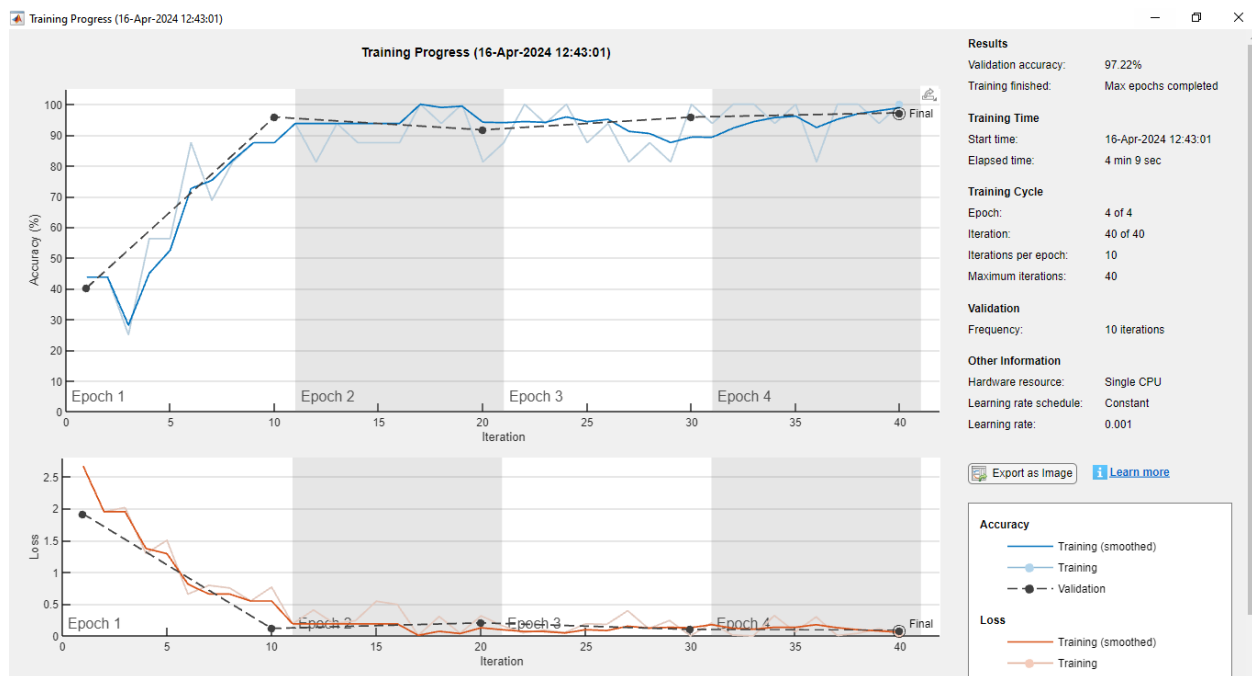


Figura 4-3. Resumen de entrenamiento óptimo GoogLeNet

El gráfico inferior marcado en naranja indica la función de pérdida (*loss*), en el que podemos ver cómo rápidamente comienza a decaer y a partir de la iteración 10 baja del 0.5 aproximándose en las siguientes iteraciones poco a poco al 0, que es el valor ideal. Con esto podemos medir el error de las muestras y con esa aproximación hacia el valor ideal podemos comprobar que el entrenamiento se ha desarrollado de manera satisfactoria. Señalar, en este sentido, que el error ideal debería ser nulo. Por otra parte, se muestra la evolución de dos gráficas correspondientes también a entrenamiento y validación, las cuales progresan también en paralelo hacia la convergencia, lo cual vuelve a garantizar un progreso adecuado de generalización.

La selección de estos parámetros se debe a que hemos probado otro tipo de combinaciones con los que se han obtenido resultados significativa o ligeramente peores. Por ejemplo, se ha probado a utilizar una tasa de aprendizaje inferior como 0,0001 o un mayor número de épocas como 6 u 8, buscando una mayor precisión a costa de aumentar más el tiempo de entrenamiento, si bien los resultados fueron ligeramente peores que esta combinación, consiguiendo precisiones alrededor del 90% e inferiores.



Experimentando con una tasa de aprendizaje ligeramente mayor obtuvimos el problema mostrado en la Figura 4-4, donde se puede observar cómo el entrenamiento no finaliza adecuadamente, deteniéndose en mitad de la segunda época, con una precisión de tan solo el 16,67%. En el gráfico de la función de pérdida se puede observar cómo tras la décima iteración el valor se dispara a valores muy elevados, causando que se convierta en algo imposible de calcular (NaN: *Not a Number*). Esto sucede debido a que la tasa de aprendizaje es demasiado elevada y se producen excesivas actualizaciones en los pesos provocando que el modelo diverja en vez de convergir.

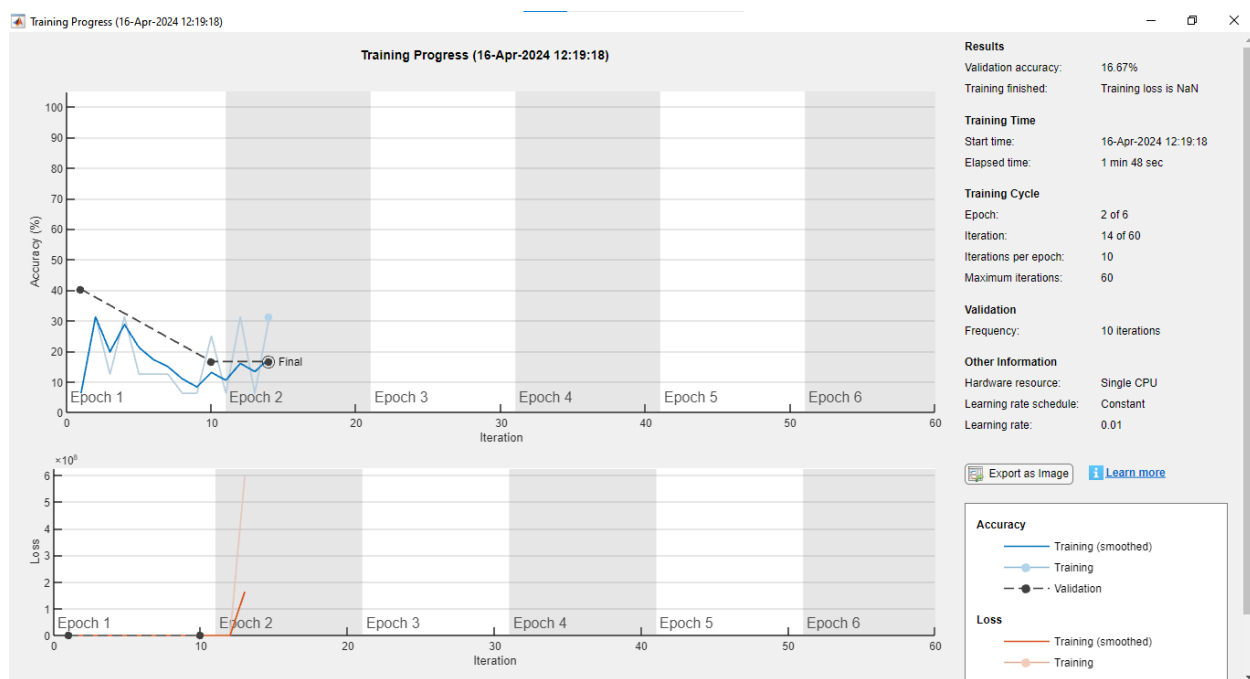


Figura 4-4. Resultados entrenamiento fallido GoogLeNet

Una vez llevado a cabo el entrenamiento, procedemos a realizar la clasificación usando el modelo de la red GoogLeNet entrenado. Cuando esta se realiza, aparece una ventana con la predicción que ha realizado el modelo. Como se puede ver en la Figura 4-5, se muestran un total de ocho imágenes, con el porcentaje de predicción realizado. En tres de estas imágenes el modelo hace una predicción correcta con un porcentaje de seguridad del 100%, en cuatro imágenes ofrece un porcentaje de probabilidad de más del 99% y por último en una de las imágenes ofrece un porcentaje del 91,6%. En las ocho imágenes escogidas el modelo predice correctamente el

monumento de cada una, demostrando así que el entrenamiento ha sido satisfactorio y el modelo es capaz de realizar una clasificación correcta.

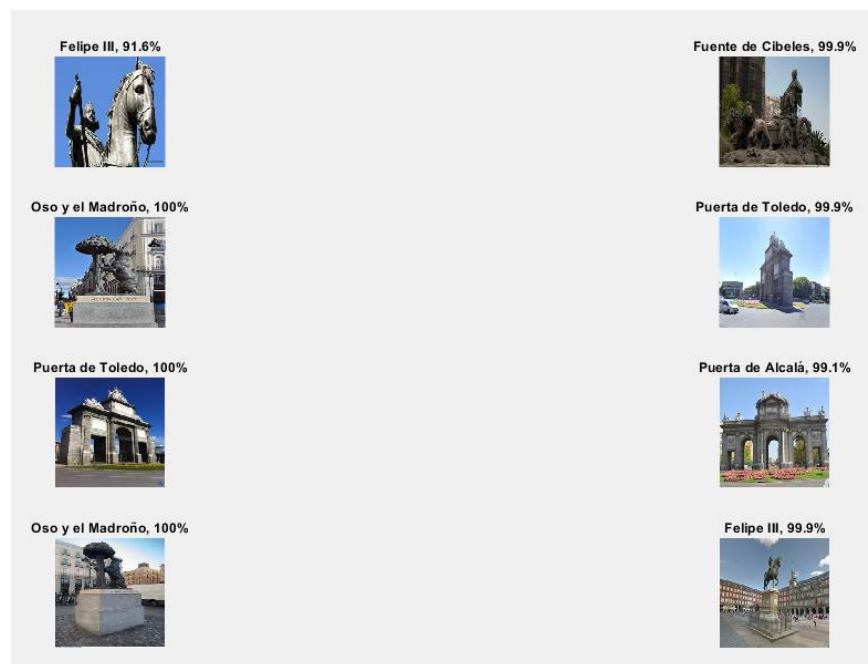


Figura 4-5. Imágenes clasificadas con GoogLeNet

Por otro lado, la clasificación nos ofrece también la matriz de confusión que se puede observar en la Figura 4-6. Sirve como método utilizado para evaluar el rendimiento de un modelo a la hora de realizar la clasificación. En las filas se representan las clases reales y las columnas representan las clases predichas por el modelo. Lo ideal es que se trace una única diagonal, lo cual indicaría que el modelo ha predicho correctamente todas las clases. En este ejemplo podemos ver cómo la diagonal está claramente representada, sólo hay tres posiciones fuera de la diagonal, lo que representaría fallos en la clasificación. Al ser solo tres instancias de todas las posibles, concluimos que se trata de un muy buen resultado y que el modelo está entrenado correctamente y puede clasificar de modo satisfactorio. Tanto en la parte inferior como en la parte derecha, se pueden observar los porcentajes correspondientes a cada clase, en este caso tres de las clases han recibido un resultado correcto al 100% y las otras tres restantes poseen un 92,3%, respaldando los correctos resultados mencionados anteriormente.

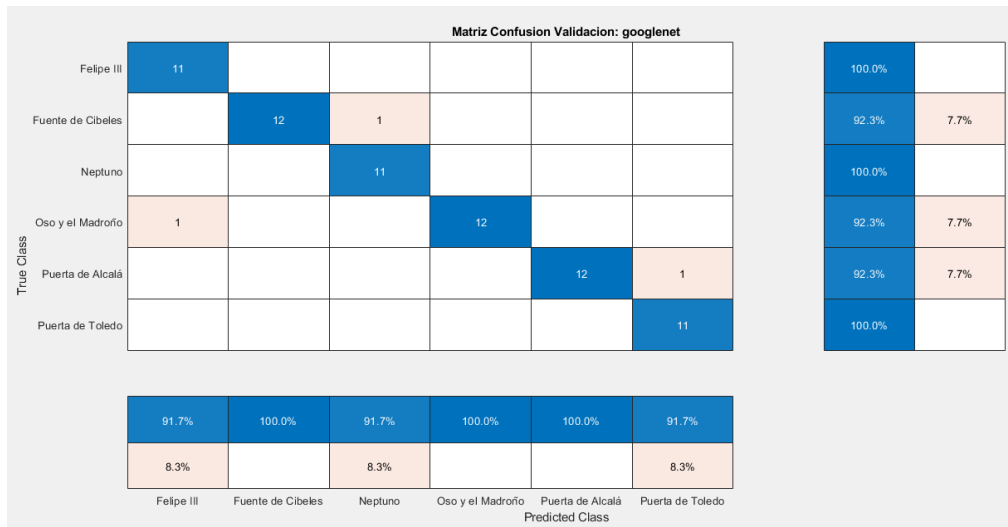


Figura 4-6. Matriz de confusión para el modelo GoogLeNet

### 4.3 Entrenamiento y clasificación con AlexNet

Observando el analizador del modelo de red que se refleja en la Figura 4-7, podemos apreciar algunas diferencias respecto a GoogLeNet. El analizador de AlexNet muestra que el número de capas es de 25, mientras que el número total de pesos a aprender es de 56.8 millones. También podemos volver a observar que no se presentan avisos ni errores, lo que nos indica que el modelo está listo para realizar el entrenamiento.

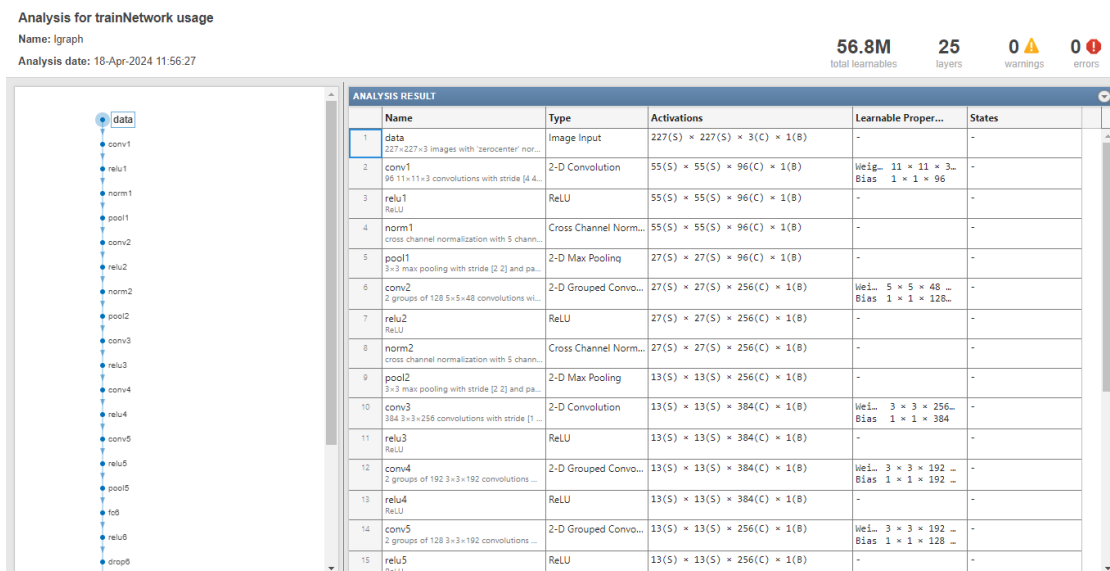


Figura 4-7. Analizador modelo red AlexNet

Durante las diferentes pruebas llevadas a cabo con este modelo, hemos podido apreciar que es un modelo que trabaja con mayor rapidez que GoogLeNet, pero sus precisiones empeoran algo en cada entrenamiento. Siendo los resultados de entrenamiento un poco menos precisos que los obtenidos con GoogLeNet. Finalmente se ha determinado que los parámetros óptimos para nuestras clases y tamaño de *dataset* son los siguientes:

- Tamaño del minilote: 10
- Número de épocas: 6
- Tasa de aprendizaje: 0,0003 (para SGDM, para adam y RMSProp se puede usar igualmente 0,0003 o también 0,0001)

En la Figura 4-8 se muestran los resultados obtenidos utilizando los parámetros óptimos mencionados. Como se puede observar, se ha obtenido un porcentaje de precisión de validación del 95,83% siendo este un muy buen resultado. Aunque el número de épocas es mayor que los parámetros usados en GoogLeNet y el tamaño de minilote también es inferior, este proceso ha obtenido una mejora de tiempo, completando el entrenamiento en 2 minutos y 50 segundos. En el gráfico superior que representa la curva de aprendizaje se puede observar cómo en la primera época se produce una gran mejora, que si bien sufre una pequeña bajada en la segunda época, la mejora es constante durante las cuatro últimas épocas. Del mismo modo, en el gráfico inferior que representa función de pérdida podemos observar cómo gradualmente el gráfico desciende hasta aproximarse al valor ideal de 0. Como curiosidad, en las iteraciones 30 y 40 se produce una subida en la función de pérdida que se corrige posteriormente ofreciéndonos buenos resultados. El proceso de entrenamiento hace un total de 102 iteraciones durante las 6 épocas.

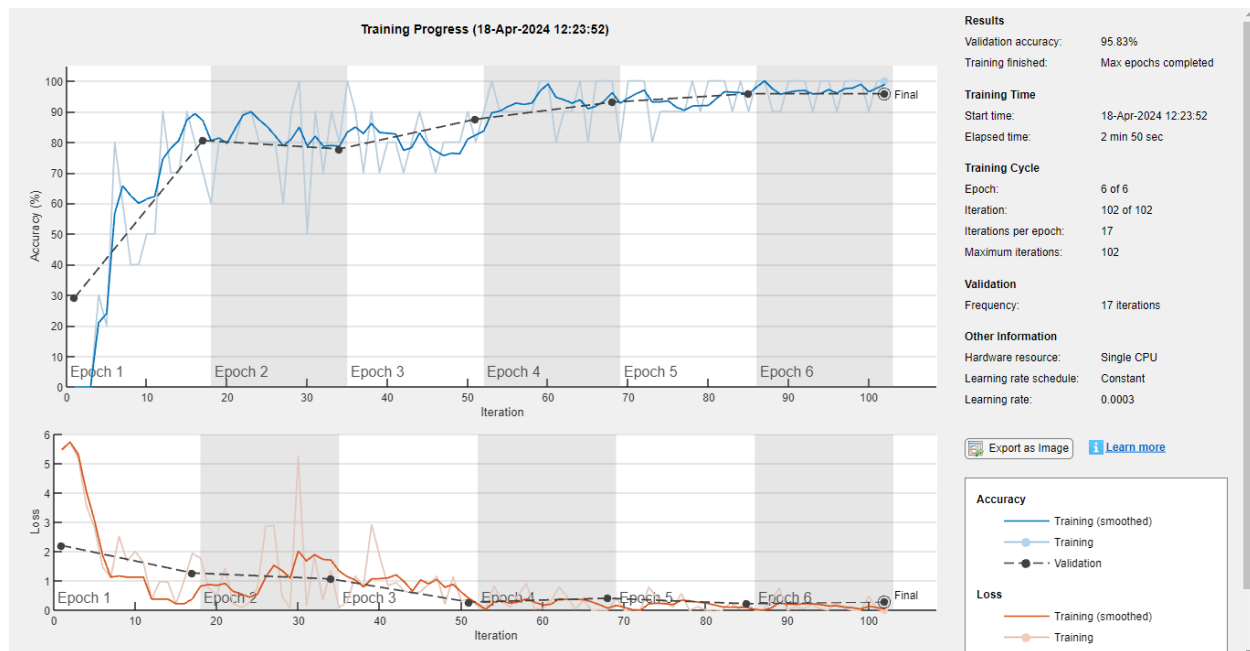


Figura 4-8. Resultado de entrenamiento óptimo AlexNet

Se han realizado diversas pruebas incrementando el número de épocas para tratar de aumentar la precisión, pero hemos descubierto que la precisión da resultados peores llegando incluso a caer por debajo del 80%. Esto es debido a que se produce sobreajuste o sobre-entrenamiento (*overfitting*), lo que produce que el modelo se adapta demasiado bien a los datos de entrenamiento específicos, pero no generaliza bien datos nuevos o no vistos, lo que provoca un rendimiento deficiente en datos de prueba o en situaciones del mundo real cuando un usuario introduzca nuevas imágenes antes no vistas. En este sentido, conviene señalar que tanto las curvas de precisión como de pérdida en entrenamiento y validación muestran algunas diferencias significativas entre ellas, aunque progresen de forma paralela. También hemos experimentado el mismo error que sucedía con GoogLeNet, y es que al aumentar la tasa de aprendizaje a 0,01 el entrenamiento no se completa debido a que la función de pérdida se vuelve incalculable (*NaN*).

A la hora de realizar el proceso de clasificación, hemos obtenido resultados satisfactorios nuevamente. En la Figura 4-9 se observa las ocho imágenes de prueba que el modelo ha clasificado. De estas ocho imágenes, siete de ellas tienen una predicción del 100% o extremadamente cercano a ello. Sin embargo, podemos ver cómo una

imagen posee un porcentaje de predicción del 68,3%, de igual modo el modelo predice que esta imagen corresponde al monumento de Neptuno, lo cual es correcto.

Si nos fijamos en la Figura 4-10 que representa la matriz de confusión para este modelo, podemos observar que los resultados son satisfactorios. Nuevamente se refleja una clara diagonal donde tan solo tres instancias están equivocadas. Esta vez, dos de las tres instancias equivocadas pertenecen a un mismo monumento. Lo que nos da unos resultados de 100% para cuatro clases, una clase con un porcentaje del 92,3% y la última clase con un 85,7%. A pesar de que esta última clase tiene el peor resultado es un muy buen porcentaje y por tanto consideramos que el modelo está entrenado correctamente y ofrece resultados satisfactorios a la hora de clasificar.



Figura 4-9. Imágenes clasificadas con AlexNet

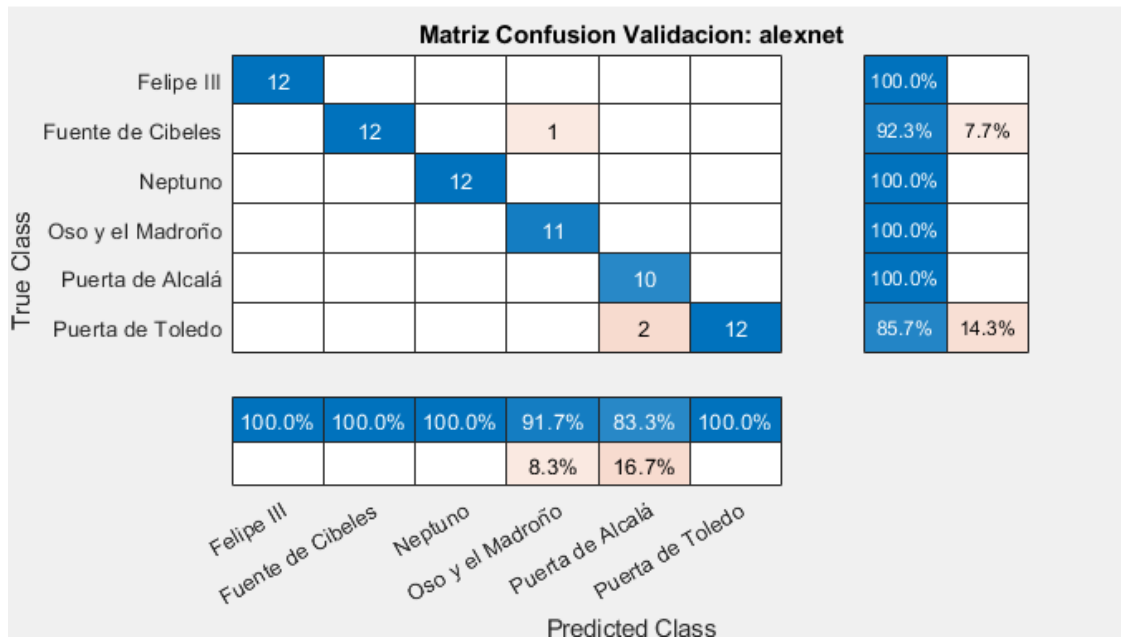


Figura 4-10. Matriz de confusión para el modelo AlexNet

#### 4.4 Entrenamiento y clasificación con SqueezeNet

Observando el analizador del modelo de red de SqueezeNet, que se muestra en la Figura 4-11, podemos observar algunas diferencias respecto a los diferentes modelos. En el caso de esta red hay un total de 68 capas, y un total de algo más de 725.000 pesos para aprender. Respecto a GoogLeNet y AlexNet, se puede ver que el número de pesos es drásticamente inferior, esto es debido a que SqueezeNet utiliza varias estrategias innovadoras, para intentar mantener un rendimiento competitivo reduciendo el número de pesos a aprender. Hemos podido descubrir con las diferentes pruebas realizadas que el modelo SqueezeNet es el más eficiente en cuanto a tiempo, logrando los entrenamientos en el menor tiempo posible, comparado con los demás. Por último, podemos observar cómo nuevamente no existen avisos ni errores, lo que refleja que el modelo está listo para el proceso de entrenamiento.

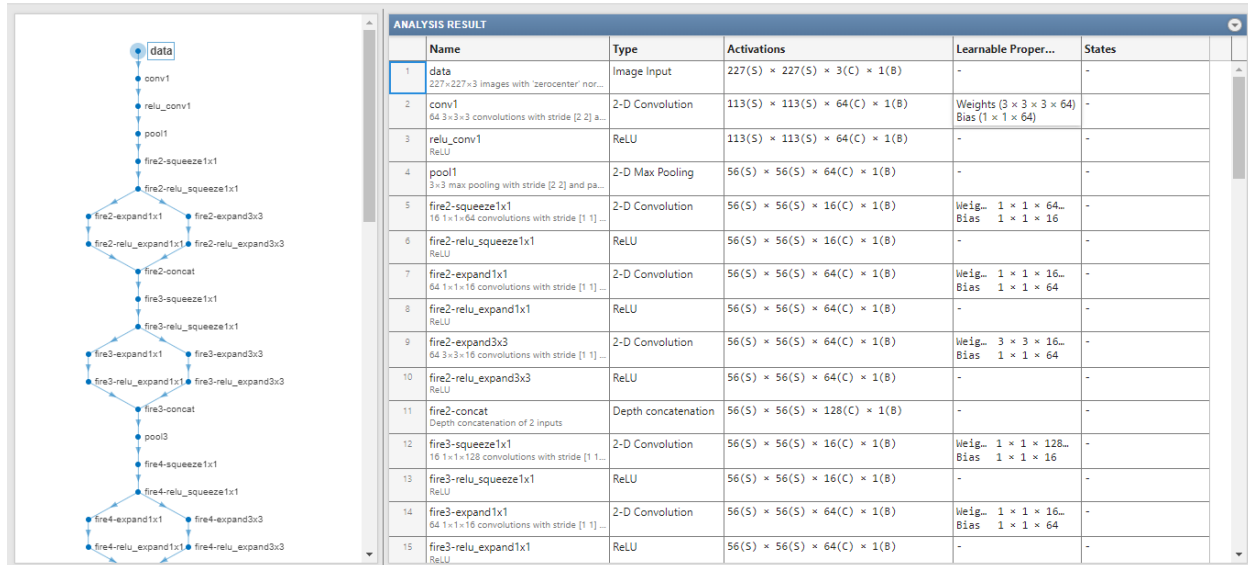


Figura 4-11. Analizador modelo red SqueezeNet

Durante las diferentes pruebas de entrenamiento llevadas a cabo usando este modelo, en general, hemos obtenido resultados satisfactorios constantemente. Sobre todo, probando parámetros similares a los que anteriormente han dado buenos resultados. Tras diferentes intentos, los mejores resultados han sido logrados utilizando los siguientes parámetros:

- Tamaño del minilote: 16
- Número de épocas: 8
- Tasa de aprendizaje: 0,0003 (para SGDM, para adam y RMSProp se puede usar igualmente 0,0003 o también 0,0001)

A pesar de que con este modelo es con el que más número de épocas utilizamos, un total de ocho, esta combinación de parámetros ha sido la que mejores resultados ha generado. Como podemos observar en la Figura 4-12, en este ejemplo de entrenamiento obtuvimos una precisión de validación del 97,22%, un resultado altamente positivo, que rozaba el porcentaje ideal del 100%. Además, como reflejamos, en general este modelo ha sido el más eficaz en cuanto a tiempo. En este caso de ejemplo, el entrenamiento se llevó a cabo en 2 minutos y 55 segundos, pero se lograron entrenamientos que bajaban del minuto utilizando el computador más potente.



Observando la gráfica superior de la curva de aprendizaje podemos ver que en las cinco primeras épocas se produce una mejora constante, acentuada sobre todo en las dos primeras. En la sexta época se produce una pequeña bajada que posteriormente en las dos últimas épocas se corrige alcanzando el pico máximo al final del entrenamiento. En el gráfico inferior de la función de pérdida se ve reflejado cómo la gráfica va decreciendo constantemente acercándose al valor ideal de 0. El entrenamiento realiza un total de 8 épocas, con 80 iteraciones.

En otras pruebas llevadas a cabo aumentando ligeramente el tamaño de la tasa de aprendizaje, se han obtenido resultados aceptables pero peores que los indicados, bajando en ocasiones del 90%, aunque no distanciándose mucho. También se han realizado pruebas disminuyendo el número de épocas a 6 o probando a disminuir el tamaño de minilote a 10, pero estos resultados no superaban los alcanzados con los parámetros óptimos explicados anteriormente, a pesar de que en general se superaba siempre el 90% de precisión de validación.

Al igual que ocurrió con los modelos de AlexNet y GoogLeNet, las pruebas con una tasa de aprendizaje más alta como 0,01 no nos daba resultados correctos. Esta vez la función de pérdida no se descontrolaba por completo, y el entrenamiento era capaz de terminar, pero solo alcanzaba una precisión de validación del 16,67%.

Como en los modelos anteriores, la evolución de las gráficas correspondientes a la precisión y la pérdida en entrenamiento y validación evolucionan de forma paralela, llegando a la conclusión de que la generalización es apropiada.

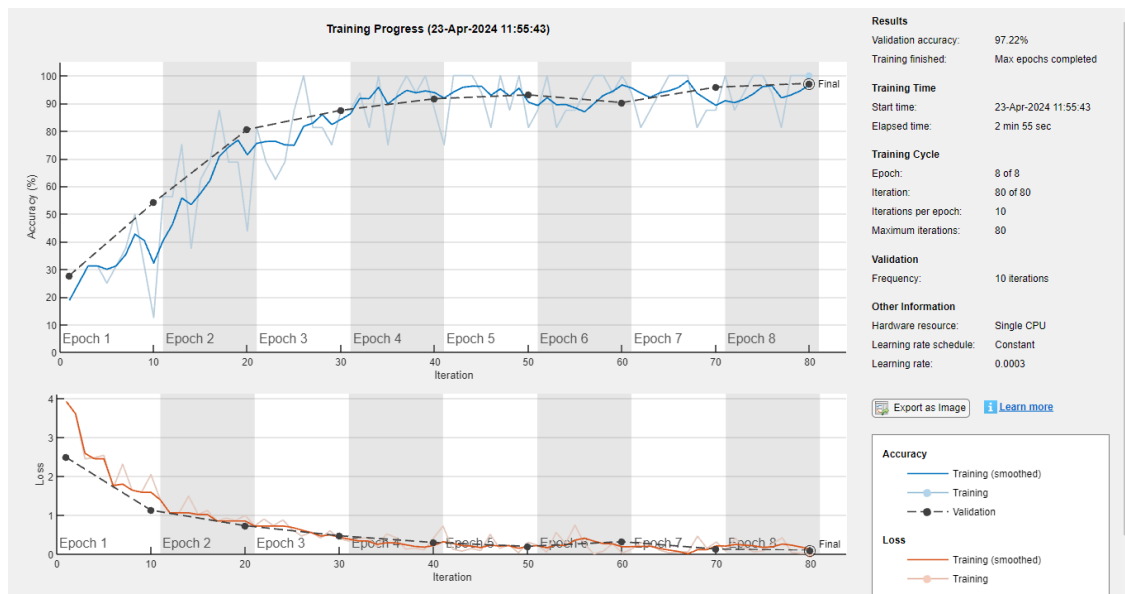


Figura 4-12. Resultado de entrenamiento óptimo SqueezeNet

Desde el punto de vista de la clasificación, con el entrenamiento realizado anteriormente, podemos ver en la Figura 4-13 cómo las ocho imágenes de prueba clasificadas han sido todas correctas. Cuatro de ellas con una seguridad de un 100% por parte del modelo, y el resto de ellas sin bajar del 95%, lo que nos indica que en estas pruebas el modelo está bastante seguro de que está haciendo la clasificación de manera precisa y correcta. Y efectivamente, como se puede ver, las ocho imágenes de prueba han sido clasificadas correctamente.

Por último, analizando la matriz de confusión de este modelo, ilustrada en la Figura 4-14, podemos observar que los resultados son muy satisfactorios. La diagonal principal queda claramente reflejada, existiendo únicamente dos instancias que reflejan error. Cada una de estas dos instancias son relativas a un monumento diferente. Esto nos da unos resultados muy gratificantes, con una seguridad de un 100% en cuatro de los monumentos, y un 91,7% en los dos restantes. En general, todos estos resultados mostrados respaldan que el entrenamiento ha sido muy correcto y que el modelo es capaz de clasificar correctamente los diferentes monumentos.

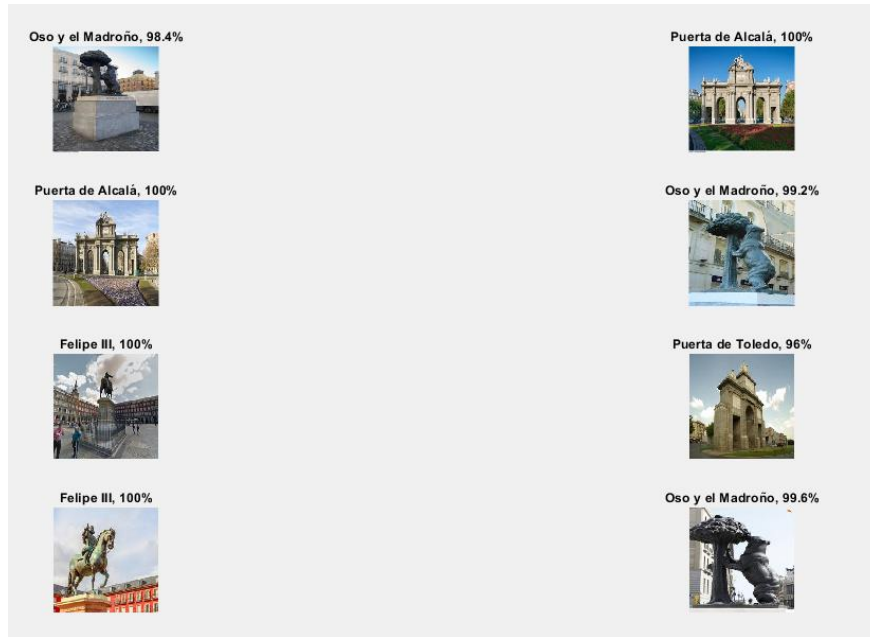


Figura 4-13. Imágenes clasificadas con SqueezeNet

Matriz Confusion Validacion: squeezeNet						
True Class	Felipe III	11				
	Fuente de Cibeles		12			
	Neptuno			12		
	Oso y el Madroño	1			12	
	Puerta de Alcalá					11
	Puerta de Toledo				1	12
		91.7%	100.0%	100.0%	100.0%	91.7%
		8.3%				8.3%
		Felipe III	Fuente de Cibeles	Neptuno	Oso y el Madroño	Puerta de Alcalá
		Predicted Class				

Figura 4-14. Matriz de confusión para el modelo SqueezeNet

## 4.5 Entrenamiento y clasificación con InceptionV3

A pesar de que el modelo InceptionV3 es una evolución de la arquitectura de GoogLeNet podemos observar que existen diferencias en el modelo de red. En la Figura 4-15 se muestra cómo este modelo tiene un total de 315 capas, y posee un total de 21.8

millones de pesos para aprender. También se vuelve a reflejar la inexistencia de avisos y errores, indicando que el modelo está listo para realizar el entrenamiento.

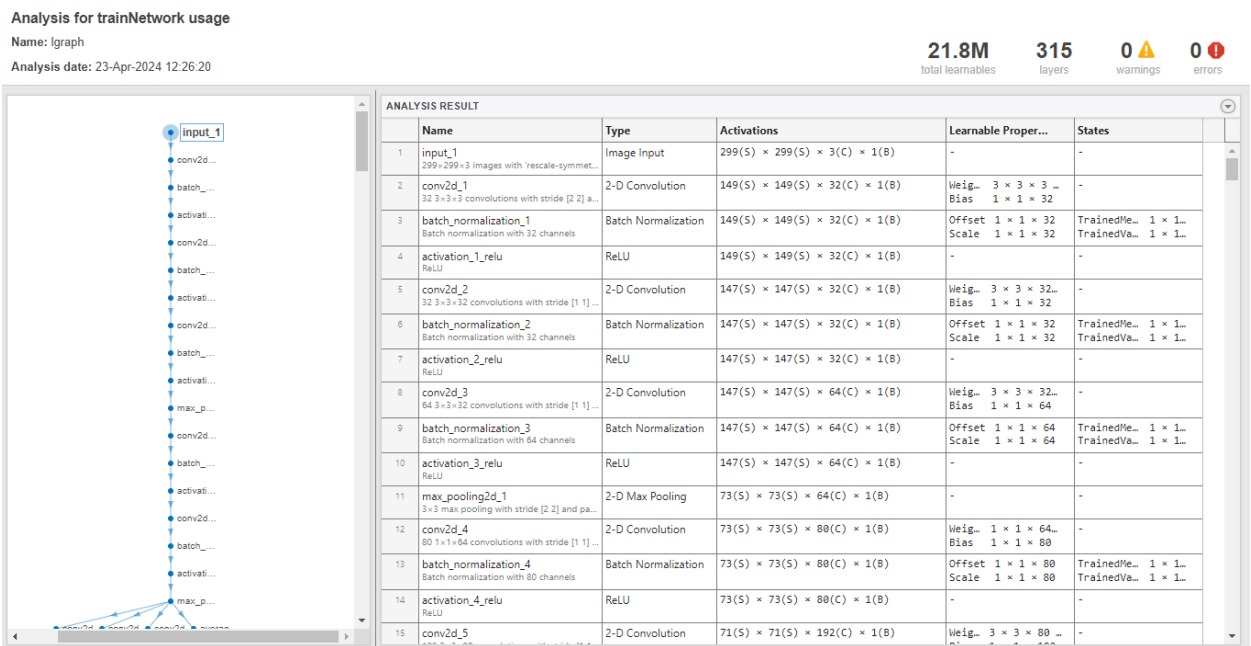


Figura 4-15. Analizador modelo red InceptionV3

Tras la realización de diferentes pruebas de entrenamiento para este modelo, hemos podido observar que a priori es el menos eficiente en cuanto a tiempo de entrenamiento. Ya que en general, los entrenamientos con diferentes combinaciones de parámetros han necesitado un mayor tiempo en comparación a los otros modelos. A la hora de la precisión, este modelo se ha mostrado muy sólido en la mayoría de las pruebas realizadas. La mejor combinación de parámetros que hemos podido establecer ha sido la misma que es utilizada en AlexNet resultando:

- Tamaño del minilote: 10
- Número de épocas: 6
- Tasa de aprendizaje: 0,0003 (para SGDM, para adam y RMSProp se puede usar igualmente 0,0003 o también 0,0001)

Cómo se ilustra en la Figura 4-16 hemos decidido elegir un entrenamiento completamente ideal. Aunque como se mencionó previamente, debido a la aleatoriedad inherente en el proceso, no siempre se obtiene un entrenamiento tan

ideal. En el ejemplo de la figura podemos ver cómo se ha alcanzado una precisión de validación del 100%, siendo esto considerado como el mejor resultado posible. En el gráfico superior de la curva de aprendizaje se puede ver que en general hay una mejora constante durante las seis épocas, con una subida muy pronunciada en las tres primeras. En la quinta época se produce una ligera bajada, pero es corregida en la última alcanzando la precisión ideal. Del mismo modo, se puede observar en el gráfico inferior que representa la función de pérdida, que durante todo el entrenamiento la gráfica desciende hasta quedar aproximada al valor ideal de 0. Este proceso de entrenamiento ha necesitado un total de 8 minutos y 39 segundos para completarse, un tiempo menos eficaz que otros modelos, pero a cambio obtiene unos resultados de entrenamiento inmejorables. Se han realizado un total de 102 iteraciones durante las 6 épocas.

En otras pruebas realizadas con este modelo, desde decrementar las épocas a 4 o aumentarlas a 8, como también aumentar la tasa de aprendizaje a un valor de 0,001 hemos obtenido resultados igualmente satisfactorios, llegando a alcanzar precisiones por encima de 95%, de forma que esta combinación reflejada en la figura es la que ha brindado los mejores resultados. Al realizar otras pruebas aumentando aún más la tasa de aprendizaje hasta el 0,01 se ha obtenido un resultado similar a SqueezeNet, donde el proceso de entrenamiento finalizaba, pero tan solo se alcanzaba una precisión de validación del 15,28%, consiguiendo que la función de pérdida no se volviese incalculable derivando en un resultado desastroso.

Como en los modelos previos se vuelve a reproducir el mismo patrón en relación a la evolución de las gráficas correspondientes a la precisión y la pérdida en entrenamiento y validación, que aunque con algunas variaciones, evolucionan de forma paralela, de suerte que el modelo consigue un nivel de generalización adecuado.

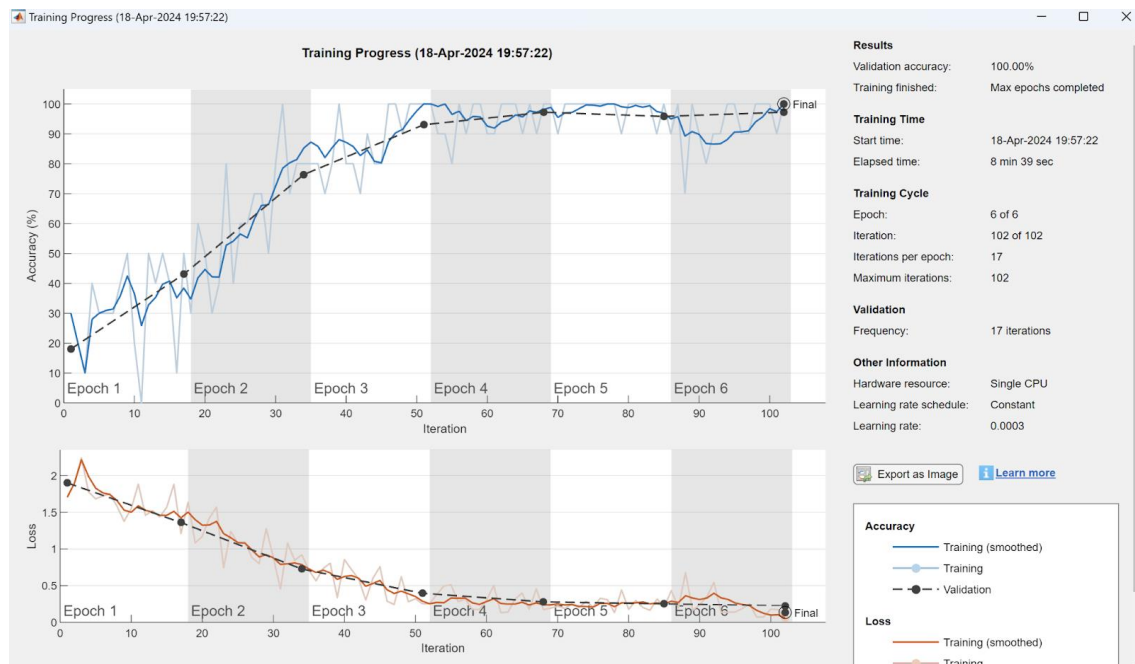


Figura 4-16. Resultado de entrenamiento óptimo InceptionV3

A la hora de clasificar con este modelo entrenado, se observa tal y como queda reflejado en la Figura 4-17 que el modelo clasifica correctamente los ocho monumentos de prueba, aunque curiosamente sus porcentajes de predicción son un poco más bajos en algunos casos que en los mejores casos anteriores. Se observa cómo en una imagen de la Fuente de Cibeles se obtiene un 51,4% o en otra de la Fuente de Neptuno un 66,8%, si bien, igualmente el modelo las ha clasificado correctamente. Las seis imágenes restantes logran un porcentaje de predicción por encima del 85%, siendo todas clasificadas correctamente.

La Figura 4-18 muestra la matriz de confusión del modelo, siendo este un ejemplo de cómo debería ser el resultado más ideal posible. Se representa una clara diagonal en la matriz, dónde no existen instancias que reflejen fallos. El modelo obtiene un 100% de seguridad en su predicción. En general, esto demuestra que el entrenamiento ha sido extremadamente satisfactorio y que los resultados de clasificación también lo son.

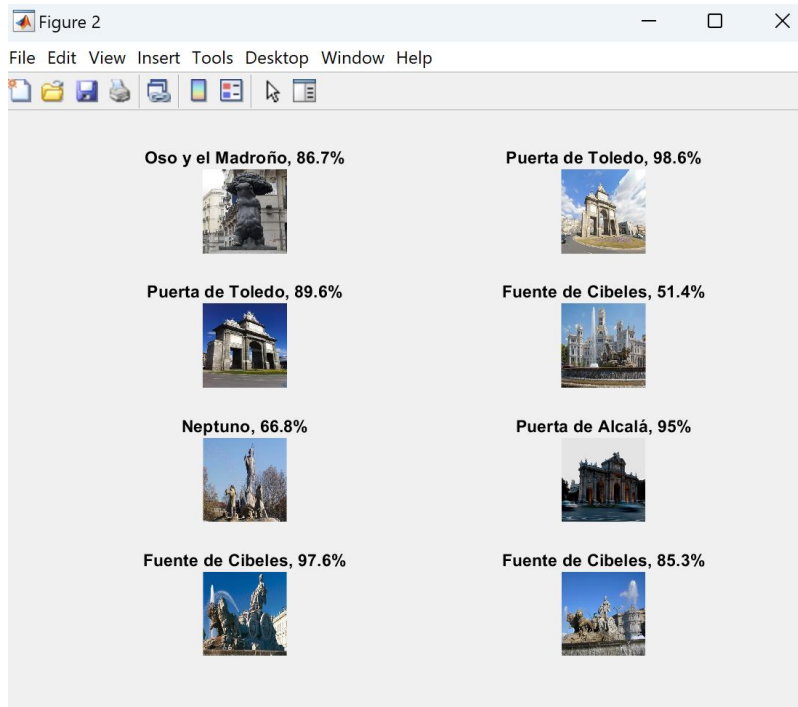


Figura 4-17. Imágenes clasificadas con InceptionV3

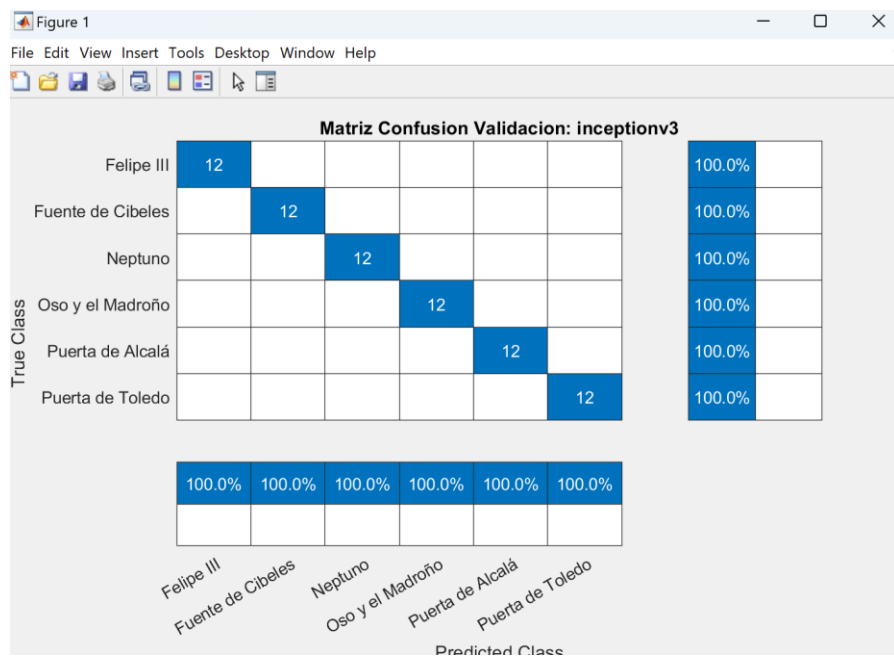


Figura 4-18. Matriz de confusión para el modelo SqueezeNet

## 4.6 Errores comunes

Los errores más comunes detectados durante el desarrollo del proyecto han sido los siguientes:

- Utilizar una tasa de aprendizaje demasiado elevada en el entrenamiento

Como se ha observado durante las pruebas, una tasa de 0,01, con el tamaño disponible de *dataset* sobre las seis clases utilizadas, no ofrece buenos resultados. En varios casos el propio proceso de entrenamiento se interrumpe por el valor de la función de pérdida que se vuelve incalculable (NaN) y en los casos en los que se completa, la precisión no supera el 25%.

- Excesivo número de épocas en el entrenamiento

Si se asignan valores muy altos en las épocas, por ejemplo por encima de 10, el modelo se adapta demasiado bien a los datos del *dataset* de entrenamiento, impidiéndole generalizar correctamente nuevos datos, es decir comete errores significativos con las imágenes de validación. Esto provoca que el rendimiento en la precisión disminuya y se obtengan peores valores.

- Identificar imágenes con varios monumentos

Probar a identificar un monumento ofreciendo una imagen que mezcle dos monumentos o más genera una clara confusión al modelo, que como es lógico no podrá predecir ambos monumentos, así es que predecirá uno de ellos y puede equivocarse respecto al deseado por el usuario.

- Identificar imágenes con obstáculos o modificadas

Utilizar imágenes donde el monumento no aparezca claramente o al completo, teniendo ciertos obstáculos que impidan una visión clara, como árboles o vehículos. También tratar de identificar imágenes alteradas y modificadas con cambios de color excesivos, eliminación del fondo o inclusión de adhesivos y marcas de agua. Estas imágenes, a veces pueden ser identificadas correctamente, pero en otras ocasiones el modelo no es capaz de predecir el monumento correcto, dando resultados desfavorables.







# Capítulo 5 - Conclusiones y trabajo futuro

## 5.1 Conclusiones

Cuando propusimos este TFG a nuestro director y comenzamos a estudiar y llevar a cabo el diseño del proyecto teníamos un gran afán de sumergirnos en el apasionante mundo de la inteligencia artificial. Entre las tecnologías que nos llamaban la atención se encontraba el aprendizaje profundo con sus redes neuronales convolucionales que han revolucionado y permitido llevar a cabo el procesamiento e identificación de imágenes. Esto nos ha dado la oportunidad de estudiar y comprender los modelos CNN pre-entrenados, su funcionamiento y explorar el proceso de entrenamiento mediante el ajuste de diferentes parámetros. Todo esto nos permitió conocer y explorar un nuevo campo de la informática en el ámbito de la Inteligencia Artificial, considerando que tiene una importancia crucial para el futuro del desarrollo tecnológico.

Además de este aprendizaje teórico, hemos tenido la oportunidad de poner en práctica lo aprendido mediante el desarrollo de una aplicación minimalista e intuitiva utilizando un lenguaje como Matlab. Consiguiendo de este modo conocer y familiarizarnos con una herramienta nueva y un nuevo lenguaje de programación. El proceso de desarrollo de esta aplicación nos resultó gratificante gracias a las facilidades que ofrece Matlab y su herramienta de diseño gráfico Matlab App Designer. En general, encontramos este entorno como un ambiente acogedor y amigable que nos ha proporcionado facilidades para poder llevar a cabo el proyecto.

Desde el punto de vista tecnológico, hemos llegado a la conclusión de que utilizando las herramientas mencionadas se puede llevar a cabo el desarrollo de una aplicación que permita el reconocimiento de imágenes mediante el uso de aprendizaje profundo mediante CNNs, proporcionando resultados satisfactorios en el reconocimiento de imágenes, en este caso de monumentos culturales correspondientes al patrimonio cultural de la ciudad de Madrid. En este sentido se ha desarrollado una aplicación íntegramente desde cero, que es apta para entrenar modelos CNN y reconocer diferentes monumentos a partir de imágenes proporcionadas.

Para concluir, este trabajo nos ha demostrado la gran importancia de realizar un trabajo de investigación previo sobre las tecnologías y conceptos desconocidos, antes de embarcarnos en un proyecto de este estilo. También, tenemos que recalcar la importancia del trabajo en equipo y la cohesión. La importancia de compartir nuestras ideas y dudas para superar todos los desafíos que se nos presentaron en el proyecto, pudiendo avanzar y llevándolo a cabo. Podemos decir orgullosamente que nos hemos demostrado a nosotros mismos que somos capaces de aceptar y superar nuevos desafíos, así como adquirir conocimientos sobre nuevas tecnologías en este campo que tanto nos apasiona.

## 5.2 Trabajo Futuro

Respecto al trabajo futuro, se han identificado los siguientes puntos que podrían ser llevados a cabo para mejora o ampliación:

- Aumentar el alcance de la aplicación, cubriendo un mayor número de monumentos que se encuentren en Madrid. Esto implicaría tener que aumentar el tamaño del *dataset* para añadir más monumentos, teniendo así un mayor número de clases y un mayor número de imágenes.
- Realizar una mejora de la interfaz que se muestra a la hora de identificar un monumento. Aumentando la profundidad de información que se ofrece del monumento, de esta forma, en lugar de ofrecer un enlace alternativo para consultar esta información, se podría añadir a la interfaz toda la información posible con un índice, y por ejemplo, incluir también un mapa para ubicar la localización exacta del monumento, monumentos cercanos y sus rutas de cómo llegar.
- Incluir un historial de identificaciones. Un apartado dónde el usuario pueda acceder a aquellos monumentos que haya accedido anteriormente, incitando de esta manera al usuario a completar el escaneo de todos los monumentos disponibles.

- Implementación de una aplicación móvil, con extensión funcional. De este modo, el usuario podría capturar una imagen desde la cámara de su dispositivo y escanear el monumento en cuestión.
- Integración de alguna red social dentro de la aplicación. Permitiendo al usuario compartir su escaneo una vez ha identificado el monumento. De esta manera podría compartir que mediante el uso de la aplicación ha identificado correctamente un monumento.
- Adición de nuevos modelos para ser entrenados y poder ser usados para clasificar e identificar, como por ejemplo ResNet y VGG-16.
- Adición de nuevos optimizadores (solvers) como BFGS o Adamax.





# Introduction

## Background

### **Tourism and Monuments**

After the crisis caused by COVID-19 and its pandemic, statistical data projections regarding global tourism levels in recent years indicate that tourism levels reached and surpassed pre-pandemic figures. In 2023, Spain surpassed 85 million visitors of international origin, exceeding by 1.9% the figures established in 2019 as a pre-pandemic reference according to La Moncloa (2024). Spain captivates travelers worldwide, whether for its climate, gastronomy, landscapes, or numerous points of interest throughout the Iberian Peninsula.

Tourism plays a crucial role in Madrid, being the city with the most visitors in the last year. Madrid is positioned as the third city in the world of the most visited urban destinations in 2023 (Viajes National Geographic 2023). However, many visitors, especially those exploring the city for the first time, may be unaware of a significant number of iconic monuments.

In this context we believe there is a need for an effective application that helps tourists identify these monuments, allowing them to learn their official names and, if desired, obtain more information about the monument.

To validate the conceptual solution, among all the different emblematic monuments of the city of Madrid, which are part of its cultural heritage, we have chosen six, which we find representative and striking. The application remains open, so in the future a large list of existing monuments throughout the city could be added. Below is a breakdown of the six monuments, including a small context of each one (Turismo de la Ciudad de Madrid, 2024. Plataforma oficial de la comunidad de Madrid, 2024) as an indication of content of tourist information that could be provided after the identification by the application.



1. Fuente de Cibeles (Cibeles Fountain): designed by the architect Ventura Rodríguez, it was built between 1777 and 1782. The fountain represents the goddess Cibeles, a symbol of Earth, agriculture, and fertility. The goddess is drawn in a chariot by two lions, representing the mythological characters Hipómenes and Atalanta. Additionally, it is an iconic place for fans of Real Madrid football team, where their titles are celebrated, like the Spanish national football team.
2. Fuente de Neptuno (Neptune Fountain): also designed by architect Ventura Rodríguez, it was built between 1777 and 1786, Representing the sea god, Neptune, holding his trident, a chariot shaped like a shell emerges from a rocky base, pulled by two seahorses, symbols of turbulent seas and storms. The fountain is also used as an iconic location by another football team in the capital, Atlético de Madrid, celebrating their titles in this emblematic place.
3. Estatua del Oso y el Madroño (Bear and Strawberry Tree Statue): this is the work of a Spanish sculptor Antonio Navarro Santafé, built in 1967. Located in the emblematic Puerta del Sol, where the zero-kilometer point of the country is located. The statue represents the heraldic arms of Madrid, with the strawberry tree surpassing the bear height, the bear rests its hand on the trunk and directs its jaws toward one of the fruits.
4. Puerta de Alcalá: this monument is one of the five former royal gates that provided access to the city, designed, and built by Italian architect Francesco Sabatini between 1769 and 1778. It consists of a neoclassical granite triumphal arch, the first of its kind built in Europe after the fall of the Roman Empire.
5. Puerta de Toledo: this gate was designed by Spanish neoclassical architect Antonio López Aguado built between 1813 and 1827. It is another of the five former royal gates providing access to the city. The triumphal arch is neo-Roman style, composed of three arches. It was erected as a commemoration of the Spanish War of Independence after the French occupation.
6. Estatua de Felipe III: the equestrian statue was initiated by Italian sculptor Juan de Bolonia and finally completed by his disciple Pietro Tacca in 1616. A gift from the Grand Duke of Florence to the King of Spain, the statue depicts King

Felipe III with a visible head, dressed in half-armor, with the Order of the Toisón de Oro hanging from his chest. In his right hand, he holds the baton of command, while in his left hand, he holds the reins of the horse, symbolizing his control over the reins of the state.

### **Technological Advancement, Deep Learning, and Convolutional Neural Networks**

From the origins of computing to nowadays the experienced technological advancement has been unbelievable, completely changing the way we live, study, work and communicate.

Starting in the 1940s with the beginning of computing, which marks the prelude to this unprecedented technological advance, continuing until the 1960s with the revolutionary appearance of the internet that years later radically changed the way in which we access and share information, to finally lead to the current era in which we live.

We find ourselves in an age where Artificial intelligence and its various applications have gained significant importance. From virtual assistants like Google Assistant or Siri, recommendation systems like the algorithms used by Netflix for suggesting series and movies or by amazon for recommending products, to natural language processing such as Google Translate or customer service chatbots used by companies to address problems. Image processing technologies like Google Lens or augmented reality and facial filters present in social media also contribute to this technological landscape.

One of the most important branches of Artificial Intelligence is Deep Learning, which is a fundamental tool for the image processing conducted in our project. Its evolution has been marked by changes in how machines interpret and understand visual information, opening new possibilities for recognizing patterns and classifying images.

Deep Learning allows the models to automatically discover relevant features from the provided input data. This capability has revolutionized how we approach image recognition, offering an unprecedented level of accuracy and generalization to date.

Finally, Convolutional Neural Networks (CNNs) are the core of Deep Learning. Instead of relying on manual engineering, the convolutional layers of these networks apply filters, enabling the detection of precise patterns to recognize and identify given images. Initially, and considering the architecture of this network models, in the first layers simple characteristics are detected, while in the following sublayers they are combined to recognize more complex characteristics. They are effective in extracting features from provided images, with improved training algorithms enabling exceptional image classification. It is precisely regarding image recognition, where the present work is located for the recognition of monuments of the cultural heritage of the city of Madrid.

## **Motivation**

The primary impetus for investigating and developing this project is born from the need for understanding the concept of image recognition and its functioning. As well as facing the challenges present in identifying and precisely classifying the monuments of Madrid as part of its cultural heritage.

Firstly, the tourism context explained earlier and its significant importance in the city we live, with the different monuments that it has. This context gives us the opportunity of applying the technologies mentioned to ease and improve the touristic experience and help to recognize and know more about these emblematic places.

Our main interest is focused on the technology of Artificial Intelligence, its branch of Deep Learning and particularly the Convolutional Neural Networks. They bring us the opportunity to be able to learn, understand and work with these technologies, enabling us to achieve our goal.

Furthermore, the growing evolution and interest for Artificial Intelligence which is used in work and daily life fields, mark the importance and social impact that these technologies are having. Our work not only focuses on the purpose of addressing a local need but also to understand and contribute to the general advance of technology and the application of Artificial Intelligence for image recognition.

Therefore, the motivation of this project is a fusion rooted in the cultural richness of Madrid and the contribution to visitors unfamiliar with the city. Through this work we aspire to be part of these technologies like Deep Learning and its growing importance in Artificial Intelligence. This project will not only provide an application but will open new innovative possibilities to help the tourists to know the cultural richness of Madrid through the use of image recognition.

## Goals

The main goal of this project is to develop a desktop application capable of identifying and classifying different images from monuments of Madrid. The objective is to highlight the cultural richness of the city and its emblematic monuments and help those unfamiliar with these iconic locations.

The objective is to develop a conceptual application that can be used easily for an average user and by experienced users that have knowledge about the technologies that the application uses.

Therefore, part of the objective is to include a user manual which explains the different functions of the application and help understand the introduction of the different concepts and fields that can be modified for making the training of a model and the classification and identification of images given.

Additionally, once the user has introduced an image and our application has classified the monument giving the name of it, we want to offer a historical and cultural context for satisfying the curiosity of users interested in that monument.

To understand the main goal of this project, we give a schematic breakdown in the next section which summarize all the planned goals for this project:

- Acquire knowledge by learning Artificial Intelligence technologies and its application by Deep Learning.
- Know and learn the functioning of Deep Learning and convolutional neural networks exploring its application in image recognition.

- Develop a minimalist and intuitive application suitable for users with varying levels of experience.
- Integrate all functional technologies under a friendly interface, performing both individual and integration tests.
- Include a user manual capable of explaining the performance of the application and explaining the concepts used.
- To experience with different models of convolutional neural networks observing the results and comparing them to understand the functioning of each model.
- Help users with the identification of monuments in Madrid, accurately and efficiently.
- Provide users with the opportunity to know the historical and cultural context, promoting the cultural wealth of Madrid.
- Generate detailed documentation of the project that collects information about the application and its development, the models used, and the results obtained. Facilitating comprehension of the project.

## **Work plan**

### **Study and choice of programming languages**

We studied two different languages: Python and Matlab. Both allow the implementation of image recognition using convolutional neural networks.

Python by using TensorFlow was an attractive option for its wide community that offers different resources. Also, due to its versatility in integrating different technologies. Meanwhile, Matlab, by using different modules such as Deep Learning Toolbox or Toolbox Computer Vision was also an interesting option. Furthermore, it has been used for a longer time and provides various facilities.

Finally, we opted to use Matlab for its wide range of specialized tools, its efficiency, and its performance. Also, Matlab offers us the ease for consulting the help

documentation and designing the application using Matlab App Designer. It seemed to us a robust, reliable, and accessible option.

### **Study and choice of convolutional neural networks**

To perform image processing we needed to have models of Convolutional Neural Networks. There are many and different types of these networks models, among all of them we decided to choose these four options: AlexNet, GoogLeNet, SqueezeNet, InceptionV3.

The choice of using these four networks is based in that they are considered as some of the most efficient and popular networks in the deep learning field. Therefore, they seemed like solid options to try and experiment with them. We chose four networks for studying the different results and comparing them.

### **Search and selection of images**

Once we decided the monuments that were going to be part of our application, we needed to search for enough images to train the models. We made an internet search looking for different images of each monument, trying to find images that had different angles and positions that allow us to recognize the monument from any perspective. The size of the images was not a factor to consider since in the model training, we needed a resizing of the images into specific sizes.

For this purpose, free use data banks have been used, with Pixabay (2023) and Openverse (2023) being the ones finally chosen.

### **Conceptual design of the application**

Before starting the development of the application, we designed the way we wanted it to look by using a whiteboard. We carried out a series of mockups that we later digitized to take as an example. Before this, we decided the name of our application with a short but descriptive name, resulting in MonuMad, combining the words 'monument' and 'Madrid' associated with its use. The logo was created using the creative tool Adobe Illustrator (2023). We opted for a horizontal design where the word MonuMad is followed by the figure of a bear and a strawberry tree (arbutus). Both are

iconic symbols of Madrid. We chose the colours blue and white, which seemed like relaxing colours that would later help us design the application with a similar style.

We designed different mock-ups in person using a whiteboard, deciding the positioning of the logo and different elements such as buttons, fields, and selectors. We created a total of three mock-ups. One for the homepage, which includes options for training, classification, and identification of a monument. We also designed a mock-up for the help page. This page is used as a user manual with various questions that users may ask and their corresponding answers. Finally, we created a mock-up for the model training page, which included various options and fields that users can modify.

While designing these mockups we also make notes on the specific functionalities that each button would have. After this, we used Balsamiq Wireframes to digitalize our designs, to maintain flexibility for future improvements and changes.

### **Development of modules and functionalities of the application**

Below are the different modules with greater relevance developed with a brief explanation of its functionalities:

- Image resizing: due to the CNN models requiring specific dimensions we developed a module (ConvertirSizesRedes.m) which carries out the resizing. Resulting in images with dimensions:  $224 \times 224 \times 3$  (GoogLeNet),  $227 \times 227 \times 3$  (AlexNet and SqueezeNet),  $229 \times 229 \times 3$  (InceptionV3). All of them arranged after resizing in their respective individual folders.
- Training the models with configurable parameters: for training, we developed three modules (Training224.m, Training227.m, Training229.m) that use the previously resized images. In these modules there are different parameters such as type of solver, minibatch size and initial learning rate. All of them can be modified by users via the graphic interface, allowing them to customize the model training.
- Classification of images: once the model is trained it's possible to classify images giving the result of the process. For this purpose, we developed a module (Classify.m) to carry out the classification displaying various windows to illustrate the different results.

- Monument recognition: we developed a module (Search.m) that uses the user provided photo and classifies it using the pre-trained CNN model showing the monument contained in the image.

### **Development of the graphic interface**

We used Matlab's graphic design tool called Matlab App Designer. This tool is intuitive and easy to use. Using a panel located on the left side of the screen you can select different interface elements such as buttons, images, numeric or text fields, and dropdowns. It only requires the elements to be dragged over the central panel of the screen which represents the application. Once the selected element has been dragged in the desired position it's possible to modify certain settings and assign associated functionalities.

### **Study of the obtained results**

Once we developed and made the application operational, we carried out different tests. By trying different combinations in adjusting the model training of its network. This allows us to analyse the behaviour of the models and compare different networks. Helping us understand the functioning of each model.

### **Search for Historical and cultural context**

To obtain data about the historical and cultural context we used official platforms specialized in local tourism, such as the official tourism page of the community of Madrid and the official community page itself. Both provide additional information about monuments, their locations, and other relevant data.

Additionally, for providing more information and details we used the online encyclopaedia of Wikipedia. Its free access and collaborative model offered a wealth of valuable information for users interested in learning more about the monuments and accessing other pages related to the monuments.

The combination between these three sources allows us to provide the user an extensive data resource ensuring information about historical and cultural context. Achieving a better experience for our users.



## **Development of the user manual**

Finally, once we had fully finished the application, we developed the information and help section, which serves as a user manual. In this section, we explained the detailed functioning of the application, providing details about customizable parameters and other miscellaneous information.

This section includes the following questions:

- What is MonuMad?
- What monuments are available?
- How to use MonuMad?
- Explanation of the Train button
- Explanation of the Classify button
- Explanation of the Identify button
- Allowed image formats
- App history
- Applications authors



# Conclusions and future work

## Conclusions

When we proposed our bachelor's thesis to our supervisor, and we started to study and work on this project we had a strong desire to dive into the passionate world of artificial intelligence. Among the technologies that caught our attention were deep learning and convolutional neural networks, which have revolutionized the process of image recognition. This gave us the opportunity to study and understand pre-trained CNN models, their functioning, and explore the training process by adjusting different parameters. All of this allowed us to explore a new field in computing that we consider important for the future development of technology.

In addition to theoretical learning, we had the opportunity to practice by developing a minimalist and intuitive application using Matlab, a tool which we were unfamiliar with. Helping us to know it and get used to working with this new tool, learning a new programming language along the way. We found the process of development of the application gratifying thanks to the facilities provided by Matlab and its graphical design tool, Matlab App Designer. Overall, we found this environment friendly and conducive to develop our project.

From a technological point of view, we have reached the conclusion that it's possible to develop an application using the tools mentioned that allows the process of image recognition, in this case of cultural monuments corresponding to Madrid's cultural heritage. Regarding this, an application has been developed entirely from scratch, which can also train CNN models and recognize different monuments with the given images.

In conclusion, this project has highlighted the importance of making a previous research process about unknown technologies and concepts before undertaking a project of this magnitude. We must also emphasize the importance of teamwork and cohesion. The value of sharing our ideas and doubts to overcome all the challenges we encountered. We can proudly say that we have proven ourselves we are capable of

accepting and overcoming new challenges as well as acquiring knowledge about these new technologies in fields that we are so passionate about.

## **Future Work**

Regarding future work we propose the following points that could be implemented:

- Increasing the scope of our application, covering a larger number of monuments in Madrid. This would involve expanding the dataset adding more monuments, having a larger number of classes and images.
- Improving the interface that displays monument data when it's identified. Instead of providing an alternative for more information, we could integrate all the information in the interface, also adding a map to show where the monument is located and display nearby monuments along with routes on how to get there.
- Include a record of identifications. A section where users can review the previously identified monuments, encouraging them to complete scanning all available monuments.
- Implementation of a mobile application. This would allow users to take a photo using their device's camera and scan the monument.
- Integration of social media inside the application. Allowing users to share the identified monument, demonstrating they've used our application for accurate monument identification.
- Adding new models to be trained and used for classification and identification, such as ResNet and VGG-16.
- Adding new solvers, such as BFGS or Adamax.





## CONTRIBUCIONES PERSONALES

### Francisco José Fernández Ferreiro

#### Trabajo de investigación:

- Análisis del tipo de objeto o entidad final a reconocer mediante el uso de la aplicación. Se sopesaron varias opciones: monumentos, razas de gatos, estadios deportivos o videojuegos.
- Estudio y aprendizaje del funcionamiento de las redes neuronales convolucionales y el proceso de reconocimiento de imágenes.
- Estudio de los tipos de redes neuronales convolucionales para su entrenamiento. Seleccionando finalmente AlexNet, GoogLeNet, InceptionV3 y SqueezeNet.
- Análisis de Matlab como posible lenguaje de programación para la aplicación.
- Análisis y elección de tres de los monumentos: Felipe III, Fuente de Neptuno y Puerta de Toledo.
- Análisis, búsqueda y elección de imágenes para el *dataset* correspondiente a los monumentos de Felipe III, Fuente de Neptuno y Puerta de Toledo.
- Estudio de las librerías y *add-ons* necesarios para el reconocimiento de imágenes mediante el uso de aprendizaje profundo usando Matlab.
- Diseño de esquemas conceptuales de la interfaz gráfica de la aplicación.

#### Trabajo de desarrollo:

- Diseño de estructura de carpetas, codificación y flujo de ejecución de la aplicación.

- Codificación de todos los diferentes módulos para que la aplicación lleve a cabo el entrenamiento de un modelo, clasificación de imágenes e identificación de un monumento.
- Codificación del módulo para lograr la redimensión de las imágenes del *dataset* con los tamaños necesarios para cada modelo.

#### **Trabajo de gestión del proyecto:**

- Creación y mantenimiento de un repositorio en GitHub para llevar a cabo el control de versiones de la aplicación.
- Creación, modificación y organización de tareas mediante el uso de Jira Software.

#### **Trabajo de documentación:**

- Redacción del capítulo 2 de la memoria, consistente en los métodos conceptuales y técnicas aplicadas. Explicándose en este capítulo teóricamente que son las redes neuronales convolucionales, las operaciones que realizan y la explicación de los cuatro modelos diferentes de red utilizados en la aplicación.
- Redacción del capítulo 4 de la memoria, consistente en los resultados obtenidos en el entrenamiento y clasificación de los cuatro modelos utilizados por la aplicación.
- Redacción del capítulo 5 de la memoria, consistente en explicar las conclusiones obtenidas y el trabajo futuro.

#### **Control de calidad:**

- Realización de entrenamientos usando los modelos AlexNet y Squeezenet para lograr parámetros óptimos y resultados satisfactorios.
- Realización del proceso de clasificación de imágenes mediante el uso de los modelos de AlexNet y Squeezenet entrenados y comprobación del correcto funcionamiento y predicción.



- Pruebas del correcto funcionamiento de identificación de monumentos usando los modelos AlexNet y SqueezeNet.
- Puesta en común de los resultados obtenidos con los diferentes modelos, realizando un análisis de estos.
- Testeo y validación de las diferentes funcionalidades de la aplicación.

## Víctor Porras Martínez

### Trabajo de investigación:

- Elección de los monumentos: Estatua del Oso y el Madroño, Fuente de Cibeles y Puerta de Alcalá.
- Análisis, búsqueda y elección de imágenes para el *dataset* correspondiente a la Estatua del Oso y el Madroño, Fuente de Cibeles y Puerta de Alcalá.
- Análisis y estudio de los diferentes optimizadores (*solvers*) a utilizar en los procesos de entrenamiento. Seleccionando finalmente: Adam, RMSProp y sgdm.
- Análisis de Python como posible lenguaje alternativo de programación para la aplicación.
- Aprendizaje de uso de la herramienta de diseño gráfica de Matlab denominada *AppDesigner*. Profundizando en el funcionamiento para la creación de botones, desplegados, campos de texto, uso de variables y depuración.
- Diseño de esquemas conceptuales de la interfaz gráfica de la aplicación.

### Trabajo de desarrollo:

- Diseño y creación de las diferentes vistas de la aplicación. Incluyendo página principal, página de ayuda y ventanas de entrenamiento e identificación.
- Diseño gráfico del logo de la aplicación.

### Trabajo de gestión del proyecto:

- Creación y organización de un repositorio en Google Drive para las diferentes versiones de la memoria, recopilación de dudas y archivos necesarios para la memoria.

- Creación, modificación y organización de tareas mediante el uso de Jira Software.
- Planteamiento de dudas sobre el proyecto y comunicación con el director del TFG.

#### **Trabajo de documentación:**

- Estructuración general de la memoria.
- Redacción del capítulo 1 de la memoria, consistente en una introducción con los antecedentes, motivación, objetivos, plan de trabajo, alcance y limitaciones del proyecto.
- Redacción del capítulo 3 de la memoria, consistente en diseño y desarrollo de la aplicación. Explicando la arquitectura de esta, la interfaz de usuario, la gestión del proyecto llevada a cabo y las herramientas y recursos utilizados.
- Redacción del capítulo 5 de la memoria, consistente en explicar las conclusiones obtenidas y el trabajo futuro.
- Redacción del apartado de bibliografía de la memoria.
- Traducción al inglés y redacción de los dos puntos en este idioma dentro de la memoria, consistentes en la introducción y las conclusiones y trabajo futuro.
- Gestión de las ilustraciones, figuras y tablas de la memoria, incluyendo los títulos correspondientes a cada uno.
- Revisión de los índices de la memoria.

#### **Control de calidad:**

- Realización de entrenamientos usando los modelos GoogLeNet e InceptionV3 para lograr parámetros óptimos y resultados satisfactorios.

- Realización del proceso de clasificación de imágenes mediante el uso de los modelos de GoogLeNet y InceptionV3 entrenados. Y comprobación del correcto funcionamiento y predicción.
- Pruebas del correcto funcionamiento de identificación de monumentos usando los modelos GoogLeNet e InceptionV3.
- Puesta en común de los resultados obtenidos con los diferentes modelos, realizando un análisis de estos.
- Revisión y corrección de las diferentes versiones de la memoria.



## BIBLIOGRAFÍA

- [1]. Adobe Illustrator (2023). Herramienta creativa. Disponible on-line: <https://www.adobe.com/es/products/illustrator.html> (Accedido abril 2024)
- [2]. Balsamic Wireframes (2023). Herramienta de maquetado digital. Disponible on-line: <https://balsamiq.com/wireframes/> (Accedido abril 2024)
- [3]. Biblioteca UCM (2023). Imágenes Creative Commons y de Dominio Público. Disponible on-line: <https://biblioteca.ucm.es/edicionweb/bancos-de-imagenes> (Accedido octubre 2023).
- [4]. BVLC AlexNet Model (2020). Disponible on-line: [https://github.com/BVLC/caffe/tree/master/models/bvlc\\_alexnet](https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet) (Accedido abril 2024).
- [5]. BVLC GoogleNet Model (2020). Disponible on-line: [https://github.com/BVLC/caffe/tree/master/models/bvlc\\_googlenet](https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet) (Accedido abril 2024).
- [6]. Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep learning. MIT Press. Disponible on-line: <https://www.deeplearningbook.org/> (Accedido abril 2024).
- [7]. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv:1602.07360v4 [cs.CV].
- [8]. ImageNet LSVRC (2012). Large Scale Visual Recognition Challenge 2012 (ILSVRC2012). Disponible on-line: <http://www.image-net.org/challenges/LSVRC/2012/> (Accedido abril 2024).
- [9]. ImageNet LSVRC (2014). Large Scale Visual Recognition Challenge 2014 (ILSVRC2014). Disponible on-line: <https://image-net.org/challenges/LSVRC/2014> (Accedido abril 2024).
- [10]. ImageNet (2020). Disponible on-line: <http://www.image-net.org> (Accedido abril 2024).
- [11]. Jira Software (2024). Herramienta para la gestión de proyectos. Disponible on-line: <https://www.atlassian.com/es/software/jira> (Accedido abril 2024).
- [12]. Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Proc. 25th Int. Conf. on Neural Information Processing Systems (NIPS'12), vol. 1, pp. 1097-1105.
- [13]. La Moncloa (2024). Actualidad industria y turismo. Disponible on-line: <https://www.lamoncloa.gob.es/serviciosdeprensa/notasprensa/industria-turismo/Paginas/2024/020224-record-turistas-internacionales.aspx> (Accedido abril 2024).

- [14]. Matlab (2023). MathWorks. Disponible on-line: <https://es.mathworks.com/> (Accedido septiembre 2023).
- [15]. Openverse (2023). Banco de imágenes. Disponible on-line: <https://openverse.org/> (Accedido mayo 2024).
- [16]. Pajares, G., Herrera, P.J., Besada, E. (2021). Aprendizaje Profundo, RC-Libros, Madrid.
- [17]. Plataforma oficial de la comunidad de Madrid (2024). Disponible on-line: <https://www.comunidad.madrid/cultura/patrimonio-cultural> (Accedido abril 2024)
- [18]. Pixabay (2023). Banco de imágenes. Disponible on-line: <https://pixabay.com/es/> (Accedido Septiembre 2023).
- [19]. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S. Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV). 115(3), pp. 211–252.
- [20]. Search CC (2023). Search Creative Commons Disponible on-line: <https://search.creativecommons.org/> (Accedido octubre 2023).
- [21]. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2014). Going Deeper with Convolutions. Computing Research Repository. arXiv:1409.4842 [cs.CV].
- [22]. Turismo de la ciudad de Madrid (2024). Página oficial de Turismo de la ciudad de Madrid. Disponible on-line: <https://www.esmadrid.com/> (Accedido abril 2024).
- [23]. Viajes National Geographic (2023). Madrid ya es la tercera mejor ciudad del mundo para el turismo este 2023. Disponible on-line: [https://viajes.nationalgeographic.com.es/a/madrid-ya-es-tercera-mejor-ciudad-mundo-para-turismo-este-2023\\_19915](https://viajes.nationalgeographic.com.es/a/madrid-ya-es-tercera-mejor-ciudad-mundo-para-turismo-este-2023_19915) (Accedido abril 2024).





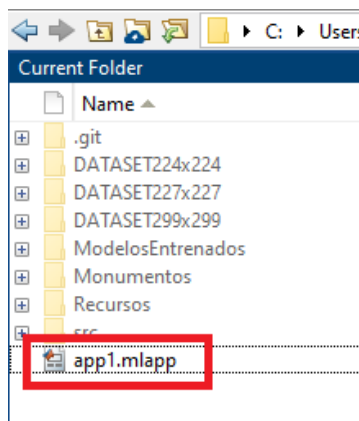


## APÉNDICES

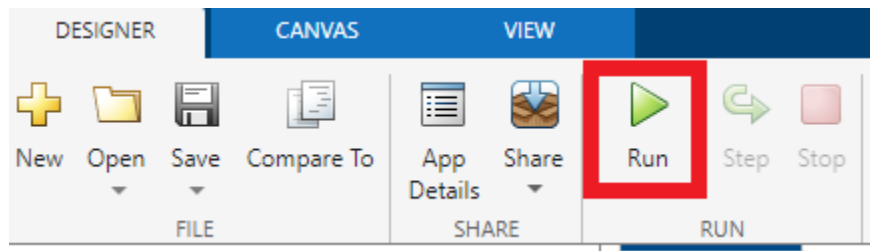
# Apéndice A - Manual de Usuario

A continuación se explican los pasos detallados que deben realizarse para ejecutar correctamente la aplicación:

1. Descargar el código de la aplicación ubicado en el siguiente repositorio alojado en GitHub: <https://github.com/Kutsoragi/TFG>
2. Descargar Matlab R2023b necesario para ejecutar la aplicación: [https://es.mathworks.com/products/new\\_products/release2023b.html](https://es.mathworks.com/products/new_products/release2023b.html)
3. Instalar Matlab junto a los siguientes *add-ons*, que configuran los toolboxes necesarios para el desarrollo y ejecución de la aplicación:
  - i. Computer Vision Toolbox
  - ii. Deep Learning Toolbox
  - iii. Image Processing Toolbox
  - iv. Deep Learning Toolbox Model for AlexNet Network
  - v. Deep Learning Toolbox Model for GoogLeNet Network
  - vi. Deep Learning Toolbox Model for Inception-v3 Network
4. Una vez instalado todos los recursos señalados, abrir la aplicación y situarse en la carpeta del código descargado en el primer paso. Hacer doble clic izquierdo o clic derecho → *Open* sobre el archivo `app1.mlapp` que aparece en el panel izquierdo:



5. Se abrirá la vista de Matlab App Designer, con una vista previa estática de la aplicación. Pulsar sobre el botón verde *Run* del panel superior:



6. La ventana con la aplicación se abrirá, mostrándose la vista principal, lista para usarse. Pudiéndose elegir la red en la parte inferior izquierda (1), acceder a la página de ayuda en la parte derecha (2) o utilizar los botones de Entrenamiento para entrenar el modelo o los de clasificación e identificación si el modelo ya está entrenado (3). Todos estos pasos son los que constituyen el cuerpo de la memoria, tal y como se ha descrito en los capítulos correspondientes.

