APLICACIÓN DE MÉTODOS DE APRENDIZAJE PROFUNDO PARA RECONOCIMIENTO DE IMÁGENES DE PLATOS DE COMIDA

APPLICATION OF DEEP LEARNING METHODS FOR IMAGE RECOGNITION OF FOOD PLATES

TRABAJO FIN DE GRADO
CURSO 2022-2023

AUTOR
CHRISTIAN ESTEBAN GÓMEZ

DIRECTOR
GONZALO PAJARES MARTINSANZ
COLABORADORA EXTERNA: CLARA ISABEL LÓPEZ GONZÁLEZ

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

APLICACIÓN DE MÉTODOS DE APRENDIZAJE PROFUNDO PARA RECONOCIMIENTO DE IMÁGENES DE PLATOS DE COMIDA

APPLICATION OF DEEP LEARNING METHODS FOR IMAGE RECOGNITION OF FOOD PLATES

TRABAJO DE FIN DE GRADO EN INGENIERÍA DEL SOFTWARE

AUTOR
CHRISTIAN ESTEBAN GÓMEZ

DIRECTOR
GONZALO PAJARES MARTINSANZ
COLABORADORA EXTERNA: CLARA ISABEL LÓPEZ GONZÁLEZ

CONVOCATORIA: JUNIO 2023

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

5 DE JUNIO DE 2023

DEDICATORIA

A toda mi familia y a Raquel, por haber
estado a mi lado apoyándome y
animándome todo este tiempo.

AGRADECIMIENTOS

A mi profesor y tutor Gonzalo Pajares Martinsanz, por toda la ayuda y orientación recibida, así como por su esfuerzo y empeño tanto en su asignatura como en la elaboración de este trabajo. También a Clara, por su contribución en el presente trabajo.

RESUMEN

Aplicación de métodos de Aprendizaje Profundo para reconocimiento de imágenes de platos de comida

El presente trabajo tiene como finalidad el desarrollo de una aplicación que permite reconocer, a partir de una imagen y haciendo uso de técnicas de Aprendizaje Profundo diferentes platos de comida.

Dentro del contexto del Aprendizaje Profundo, la aplicación utiliza dos modelos de Redes Neuronales Convolucionales, AlexNet y GoogLeNet, para realizar la clasificación de alimentos. Dichos modelos poseen la capacidad de ser adaptados y reentrenados para la identificación de los platos de comida seleccionados, siendo necesario un banco de imágenes por cada plato que queramos que identifique.

La aplicación desarrollada para este trabajo permite configurar paramétricamente ambos modelos de redes y realizar el entrenamiento con imágenes. Una vez el modelo esté entrenado, el usuario que desee identificar un plato de comida puede seleccionar una imagen almacenada en su ordenador, o bien realizar una fotografía con la cámara de su smartphone a través de la aplicación de MatLab.

De forma complementaria y dentro del paradigma del Internet de las Cosas, la aplicación se conecta a la plataforma ThingSpeak, para que cada vez que se realice una clasificación, se envíe el nombre del plato a la plataforma y haga saltar una acción, que se encarga de la publicación de un tweet con la información del nombre y calorías del plato identificado.

Palabras clave

Aprendizaje Profundo, Redes Neuronales Convolucionales, AlexNet, GoogLeNet, Reconocimiento de imágenes, Clasificación de platos de comida, Internet de las Cosas, ThingSpeak.

ABSTRACT

Application of Deep Learning methods for image recognition of food plates

The purpose of this project is the development of an application that allows to recognize, from an image and using Deep Learning techniques, different food dishes.

Within the context of Deep Learning, the application makes use of two Convolutional Neural Networks models, AlexNet and GoogLeNet, to classify food. These models can be adapted and retrained to identify selected food dishes, requiring an image bank for each food dish that we want to identify.

The application developed for this project allows some parametric configuration for both network models and training them with images. Once the model is trained the user who wants to identify a food dish can select an image stored on the computer or take a picture with the camera on a smartphone through MatLab application.

In a complementary way and within Internet of Things paradigm, the application connects to ThingSpeak platform, so each time a classification is made, the name of the dish is sent to the platform and triggers an action, consisting of publishing a tweet with information of the name and calories of the identified dish.

Keywords

Deep Learning, Convolutional Neural Networks, AlexNet, GoogLeNet, Image recognition, Food dishes classification, Internet of Things, ThingSpeak.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

1.1 Motivación

La nutrición ha sido y es uno de los aspectos más importantes en nuestras vidas, ya que tiene un impacto directo en nuestra salud. Desde hace años se han desarrollado y usado múltiples herramientas en este ámbito para ayudar a las personas a llevar un control y así poder lograr una alimentación más saludable. El principal problema que tenían o tienen es el requerimiento de introducir de forma manual la información de los alimentos, lo cual resulta ser un proceso a veces tedioso y propenso a errores.

Gracias a la llegada de la tecnología, y más concretamente, de los dispositivos móviles y sus aplicaciones, el uso de estas herramientas se ha popularizado, en gran medida, gracias a su posibilidad de registrar la ingesta desde cualquier lugar y a la facilidad para, haciendo uso de la cámara del dispositivo móvil, tomar una fotografía del alimento y, mediante el uso de metodologías inteligentes, determinar de forma automática y directa el contenido nutricional de este, así como su información calórica. Además, con el avance constante de la tecnología, estas aplicaciones se han vuelto más sofisticadas y precisas, posibilitando, no solamente la identificación de alimentos individuales, sino también de platos de comida completos, teniendo en cuenta sus componentes.

Cabe destacar el trabajo realizado por Shifat y col. (2022) sobre la realización de un sistema para reconocer alimentos que, utilizando principios de aprendizaje automático y de visión por computador, han logrado obtener unos resultados de precisión de un 98.05%. Por otro lado, encontramos el sistema creado por Liu y col. (2019), destinado a mejorar la precisión de los *datasets* pequeños que se usan para realizar el entrenamiento, haciendo uso de técnicas tales como aumento de imágenes (*image augmentation*) para crear un banco de imágenes más equilibrado y amplio.

Aunque estos dos trabajos anteriores no se enfocan directamente en el cálculo de calorías, su enfoque resulta relevante para mejorar la precisión y velocidad de respuesta para este tipo de aplicaciones.

A pesar de que aún quedan tareas por delante para perfeccionar y ampliar estas herramientas, trabajos como el de Aktı y col. (2022) demuestran el interés por lograr la expansión de estas, en este caso, enfocándose en el desarrollo de una aplicación de reconocimiento de alimentos centrada en la cocina del Medio Oriente, la cual ha permanecido prácticamente sin explorar, haciendo uso del modelo Mobilenet-v2 y de métodos de aumentos de datos para las clases subrepresentadas, logrando una precisión del 94% en 23 clases de alimentos.

También deberían mencionarse los estudios realizados por Lo y col. (2020) para la implementación de técnicas de aprendizaje profundo para realizar estimaciones sobre el volumen o la variabilidad de las porciones de los alimentos de un plato de comida a través de una simple imagen.

El trabajo de Barylak Alcaraz (2021), dentro del contexto de Internet de las Cosas (IoT, *Internet of Things*) se sitúa en la misma línea que la presente aplicación, de forma que se identifican también platos de comida mediante un dispositivo móvil, realizando las gestiones correspondientes al cálculo, almacenamiento y envío de mensajes a través de la plataforma ThingSpeak (2023). Dicho trabajo sirve de inspiración para el desarrollo de la aplicación que se presenta a continuación.

1.2 Objetivos

Este proyecto tiene como principal objetivo el desarrollo de una aplicación de escritorio para identificar y clasificar imágenes de platos de comida haciendo uso de métodos de aprendizaje profundo. Tiene como finalidad servir de utilidad para todas aquellas personas que deseen conocer las calorías de un plato de comida o que estén interesadas en llevar un control de su ingesta de calorías, para así poder mejorar su salud.

Además, se desarrollará una versión móvil que tendrá acceso constante a internet y que se conectará con una plataforma IoT, la cual almacenará y procesará los datos generados por la aplicación en la clasificación para, finalmente, publicarlos en la red social Twitter.

De tal forma, se generan los siguientes objetivos específicos:

- Aplicación y adaptación de métodos de aprendizaje profundo, en este caso, Redes Neuronales Convolucionales (RNC), más concretamente AlexNet y GoogLeNet.
- Rentrenamiento de los distintos modelos de redes haciendo uso de un banco de imágenes previamente seleccionado.
- Desarrollo de una interfaz gráfica para la aplicación que sea amigable, fácil de usar y que permita realizar todo el proceso de entrenamiento y clasificación de imágenes.
- Conexión con plataformas externas IoT, en este caso ThingSpeak, para almacenar datos en la nube y realizar distintas operaciones con ellos.
- Publicación de los datos resultantes de la clasificación en redes sociales, en este caso Twitter, para informar y llevar un control de los resultados.

1.3 Plan de trabajo

Para la realización de este trabajo se han seguido una serie de tareas, mostradas a continuación según el orden de su ejecución:

- **Identificación y selección del lenguaje de desarrollo**
  Para el desarrollo de la aplicación se propusieron dos opciones principalmente, MatLab o Python. Ambas opciones permiten implementar y trabajar con redes neuronales convolucionales gracias a los módulos propios de cada una, *Deep Learning Toolbo*x en el caso de MatLab y *TensorFlow* en el caso de Python.
  Finalmente se optó por seleccionar MatLab debido a su integración con *MatLab App Designer* para desarrollos de interfaz gráfica y por su facilidad de conexión con la futura plataforma ThingSpeak.
- **Identificación y selección de las redes neuronales convolucionales**
  Para poder conseguir el propósito de esta aplicación, clasificar imágenes de platos de comida, se pensó en el uso de redes neuronales convolucionales, de las cuales, existen una gran variedad de modelos, por

ello debía seleccionar las más apropiadas. Se decidió utilizar los modelos AlexNet y GoogLeNet, los cuales son modelos ya pre-entrenados, que permiten reentrenarse para realizar clasificaciones de las imágenes que se desee en cada momento, lo que se conoce técnicamente como trasferencia de aprendizaje (*transfer learning*).

- **Búsqueda del banco de imágenes**

  Para poder volver a entrenar los modelos de redes seleccionados, era necesario disponer de un banco de imágenes lo suficientemente grande. Después de varias búsquedas, se eligió el *dataset iFood – 2019 at FGVC6*, que incluye imágenes de platos de comida clasificados por su nombre.

- **Desarrollo de los distintos módulos de la aplicación**
  - **Redimensionamiento de imágenes**: Los modelos de red seleccionados necesitan recibir las imágenes con unas dimensiones específicas, 227 x 227 x 3 en el caso de AlexNet y 224 x 224 x 3 en el caso de GoogLeNet.
  - **Selección de los posibles distintos modelos de redes**: Al disponer de varios modelos de redes neuronales convolucionales, hay que posibilitar la opción de seleccionar uno u otro, tanto a la hora de realizar el entrenamiento, como a la de realizar la clasificación.
  - **Entrenamiento y parametrización de las redes neuronales convolucionales**: Para poder realizar la clasificación de las imágenes, es necesario realizar previamente el entrenamiento de ambos modelos de red haciendo uso del banco de imágenes. Estos permiten ser configurados a través de una serie de parámetros, por lo que sería necesario ofrecer una forma de poder modificarlos para encauzar el proceso con mayor precisión, mayor velocidad o un punto intermedio.
  - **Clasificación de imágenes de platos de comida**: Haciendo uso de los archivos de entrenamiento generados en el proceso de entrenamiento, la aplicación tiene que permitir seleccionar una imagen almacenada en nuestro ordenador o tomada

directamente con la cámara de un dispositivo móvil, clasificarla y mostrar el resultado obtenido por pantalla.

- **Diseño y elaboración de la interfaz gráfica de la aplicación**
  Con el objetivo de lograr un uso intuitivo y sencillo de la aplicación, se procedió a elaborar una interfaz gráfica amigable. Se realizaron bocetos (o *mockups*) para cada una de las vistas que tendría la aplicación para, posteriormente, haciendo uso de la herramienta *MatLab App Designer*, implementarla de modo que permitiera realizar todas las funciones descritas en el punto anterior.
- **Desarrollo de la versión móvil de la aplicación**
  Para facilitar el uso de la aplicación y poder usarla desde cualquier lugar sin necesidad de disponer de un ordenador, se decidió desarrollar una versión móvil sencilla haciendo uso de MatLab Drive y la aplicación móvil de MatLab. De esta forma, podríamos tomar una fotografía con un dispositivo móvil y obtener el resultado de la clasificación directamente.
- **Conexión a la plataforma ThingSpeak**
  Para ampliar la funcionalidad de la aplicación, se decidió hacer uso de la plataforma ThingSpeak, la cual permite conectarse con la aplicación en MatLab y ofrecer varias utilidades en consecuencia. En este caso, se utilizaría para almacenar la información calórica de cada plato de comida que se identifique, de tal forma, que cada vez que se escanee un plato, se conecte con una cuenta de Twitter para publicar un *tweet* con la información resultante de la clasificación.

1.4 Organización de la memoria

La memoria se organiza de forma que en el capítulo dos, se describen los métodos y técnicas aplicadas para implementar la funcionalidad requerida. En el capítulo tres se describe el diseño de la aplicación. En el capítulo cuatro se muestran y comentan los resultados obtenidos. Finalmente, en el capítulo cinco se resumen las conclusiones obtenidas y se realizan las pertinentes consideraciones sobre los avances de futuro.

Capítulo 2 - Métodos y técnicas aplicadas

Para la realización de este trabajo se han empleado las Redes Neuronales Convolucionales (RNC) AlexNet y GoogLeNet. Las RNC son un tipo especializado de redes neuronales, con una topología basada en rejilla para el procesamiento de datos (Pajares y col., 2021), en este caso, para procesar e identificar imágenes de platos de comida.

Es propio que este tipo de modelos de red dispongan de dos fases: aprendizaje, en la cual se ajustan los pesos del modelo, contenidos en los núcleos de convolución, y clasificación. Además, poseen una serie de operaciones en común, las cuales son necesarias para completar el proceso.

2.1 Operaciones aplicadas en las RNC

2.1.1 Convolución

La convolución es una operación que se define como sigue para el procesamiento de imágenes, que como se sabe son estructuras bidimensionales (2-D),

$$C(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{2-1}$$

El resultado de la convolución es $C(i,j)$ para un determinado píxel (i,j) cuando la entrada es una imagen I o bien un elemento de un tensor cuando se trata de capas más internas. En la ecuación anterior K hace referencia a lo que se denomina núcleo de convolución. La aplicación de esta expresión al contexto de las imágenes resulta en el esquema gráfico mostrado en la Figura 2-1. El núcleo de convolución K con sus correspondientes valores, se posiciona sobre la cuadrícula superior izquierda para obtener el resultado P que se posiciona en la cuadrícula superior izquierda. A continuación, se desplaza el núcleo una posición hacia la derecha para obtener de forma similar el resultado Q y así sucesivamente se desplaza el núcleo de izquierda a derecha y de arriba hacia abajo hasta completar los valores de las celdas en la matriz resultante.

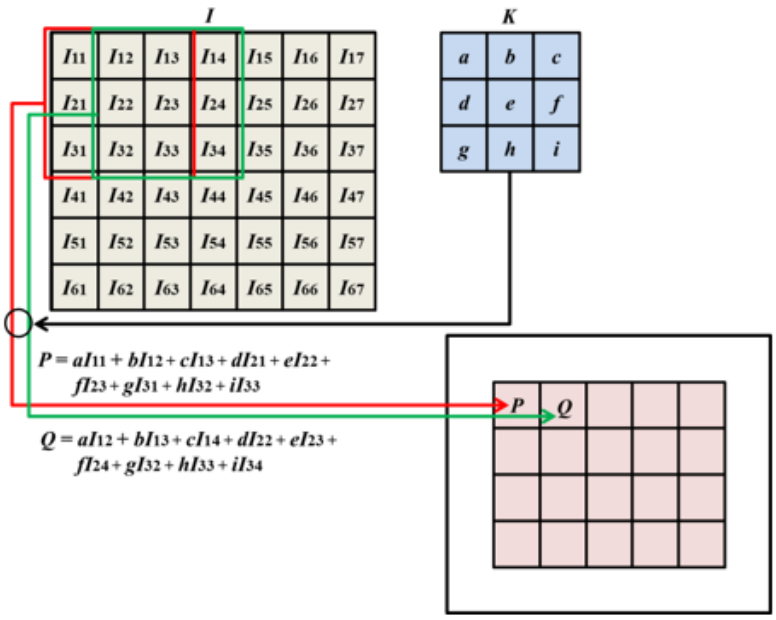


Figura 2-1. Ejemplo de convolución 2-D

No obstante, si este núcleo en lugar de desplazarse una única celda se desplaza más de una, es necesario establecer lo que se conoce como *stride (s)* que puede tomar valores mayores que la unidad, Además, como el núcleo puede desbordar la imagen o tensor de entrada cuando la celda correspondiente al valor e del núcleo K se sitúa en las filas o columnas más exteriores, existen valores del núcleo que no se utilizan, en cuyo caso estas filas o columnas quedan sin procesar. Por esta razón, si se desea que el resultado tenga las mismas dimensiones de la imagen, es necesario añadir filas y columnas rellenas de ceros, operación que se conoce como *padding* (p). En función de los valores s y p para una imagen de entrada con dimensión i se obtiene la correspondiente dimensión de salida o según las siguientes relaciones. En estas expresiones k es la dimensión del núcleo de convolución que, por ejemplo, en el caso del ejemplo anterior, $k = 3$.

$$Relación\ 5: o = \left[\frac{i-k}{s}\right] + 1 \quad Relación\ 6: o = \left[\frac{i+2p-k}{s}\right] + 1 \qquad (2\text{-}2)$$

2.1.2 ReLU

Esta función conocida como Unidad Lineal Rectificada (ReLU, Rectified Linear Unit) se define como $f(x) = max(0, x)$, cuya representación es la que se muestra en la Figura 2-2, y cuyas características y su utilidad se centran en lo siguiente:

a) Evitar la saturación del gradiente durante el proceso de optimización por el método del gradiente descendente.

b) Disminuir la complejidad computacional por el hecho de reducir valores negativos a cero.

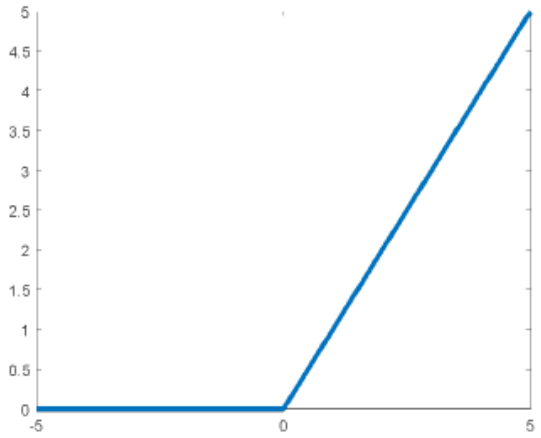


Figura 2-2. Representación de la función ReLu

2.1.3 Normalización

La operación de normalización se aplica con el fin de acelerar la convergencia durante el proceso de aprendizaje. En la expresión siguiente se define una función de normalización por lotes,

$$b_{x,y}^{i} = a_{x,y}^{i} \Big/ \left(k + a \sum_{j=\max(0,i-\frac{n}{2})}^{\min(N-1,i+\frac{n}{2})} \left(a_{x,y}^{i}\right)^{2}\right)^{\beta} \qquad (2\text{-}3)$$

donde N es el número de filtros de la capa dada, $a_{x,y}^{i}$ es la actividad de la neurona y k, α, n, β son hiperparámetros. En Krizhevsky (2012) se proponen como más apropiados los siguientes valores para los parámetros $k = 2, n = 5, \alpha = 10^{-4}, \beta = 0{,}75$.

2.1.4 Pooling

Se trata de una función cuyo objetivo es agrupar valores en las capas de características, de tal forma que dada una pequeña traslación en la entrada no afecte a los valores de las salidas. En la Figura 2-3 se muestra un ejemplo gráfico de agrupamiento por máximos, es decir seleccionando el valor máximo en cada una de las ventanas de agrupamiento, en este caso de dimensión 2x2.

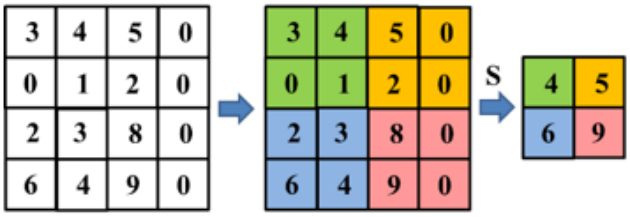


Figura 2-3. Max pooling

2.1.5 Dropout

Es un mecanismo para evitar sobresaturaciones en la red neuronal. Consiste en anular determinado tipo de neuronas de forma aleatoria mediante un hiperparámetro que indique la probabilidad de supervivencia de cada neurona.

2.1.6 Softmax

Consiste en proyectar los valores de la salida en la capa final al rango [0, 1], proporcionando así una especie de probabilidad de pertenencia a cada una de las clases para una imagen de entrada dada. Su función se define según la siguiente expresión.

$$softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} \quad (2\text{-}4)$$

2.1.7 Gradiente descendente

Es una técnica de minimización utilizada para el ajuste de los pesos involucrados en los modelos de las redes durante el proceso de retro-propagación en el entrenamiento. La función para optimizar se conoce como función *objetivo* y cuando

se está minimizando, se le denomina también *loss function* o *error function*. Se trata de minimizar una función objetivo que tiene la forma,

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} J_i(w) \tag{2-5}$$

donde el parámetro w que minimiza $J(w)$ debe estimarse, constituyendo el objetivo principal del aprendizaje. J_i se asocia con la *i-ésima* observación en el conjunto de datos utilizados para el entrenamiento (ajuste). El gradiente descendente se usa para minimizar esta función de forma iterativa, con iteraciones t,

$$w(t+1) = w(t) - \varepsilon\frac{1}{n}\sum_{i=1}^{n} \nabla J_i(w) \tag{2-6}$$

De esta forma iterativa el método recorre el conjunto de entrenamiento y realiza la actualización anterior para cada muestra. Se pueden aplicar varios pasos sobre el conjunto de entrenamiento hasta que el algoritmo converja. Estas muestras así seleccionadas constituyen lo que se denomina *batch*, como se describe más adelante a la hora de seleccionar los parámetros de entrenamiento. En este sentido, es habitual utilizar exactamente la técnica conocida como Gradiente Descendente Estocástico (SGD, *Stochastic Gradient Descent*) (Bishop, 2006; Ruder, 2017).

Las implementaciones típicas pueden usar una razón de aprendizaje adaptativa para que el algoritmo converja. En pseudocódigo, el método de gradiente descendente estocástico es como sigue,

1. Elegir un vector inicial de parámetros w (puede ser aleatoriamente) y una razón de aprendizaje a ε.
2. Repetir hasta que se consigue un mínimo aproximado.

    2.1 Seleccionar aleatoriamente ejemplos en el conjunto de entrenamiento.

    2.2 Para $i = 1,2,..n$, hacer $w(t+1) = w(t) - \varepsilon\nabla J_i(w)$

2.2 Modelos de arquitecturas en las RNC

Como se ha indicado previamente, son dos los modelos de redes seleccionados, AlexNet y GoogLeNet, que se describen a continuación y cuyos fundamentos teóricos pueden encontrarse descritos con mayor precisión en Pajares y col. (2021).

2.2.1 AlexNet

La red AlexNet (ImageNet, 2020; Russakovsky y col., 2015; Krizhevsky y col., 2012; BVLC AlexNet Model, 2021) es una de las utilizadas en el ámbito de las RNC, ganadora de la competición ImageNet LSVRC (2012) para el reto del año 2012.

En la Figura 2-4 se muestra la estructura de la red AlexNet con ilustración gráfica de las operaciones involucradas, identificando las mismas con indicación de las dimensiones de filtros en las capas convolucionales y totalmente conectadas, para las operaciones de Convolución (*conv*), ReLU (*relu*), Normalización (*norm*), *Pooling* (*pool*), *dropout* (*drop*) o *softmax*. Se incluyen las dimensiones de los filtros, el *stride* (*s*), el *padding* (*p*) y el número de filtros (*K*) en cada capa.

En la tercera columna de la Tabla 2-1 se indican los parámetros (pesos) que se aprenden en este modelo, que son los pesos en las capas (primera columna) de convolución (*conv1* a *conv5*) y las totalmente conectadas (*Fully Connected*, *fc6*, *fc7* y *fc8*), independientemente de que se aplique *dropout* o no en ellas. Obsérvese que, como pesos a aprender se incluyen en cada capa de convolución un valor de *bias* por filtro que es un parámetro de ajuste adicional. Por otra parte, en la Tabla 2-1 también se muestran en la segunda columna las dimensiones de salida correspondientes a la capa que se indica.

Tabla 2-1. Parámetros aprendidos por el modelo AlexNet

Nombre de capa	Dimensión de salida	Pesos
conv1	55×55×96	W = 11×11×3×96 b = 1×1×96
conv2	27×27×256	W = 5×5×48×256 b = 1×1×256
conv3	13×13×384	W = 3×3×256×384

		b = 1×1×384
conv4	13×13×384	*W* = 3×3×192×384 *b* = 1×1×384
conv5	13×13×256	*W* = 3×3×192×256 *b* = 1×1×256
fc6	1×1×4096	*W* = 4096×9216 *b* = 4096×1
fc7	1×1×4096	*W* = 4096×4096 *b* = 4096×1
fc8	1×1×4096	*W* = 1000×4096 *b* = 1000×1

Además, en la estructura de la Figura 2-4 se indican los números de *Relación*, que establecen precisamente la relación entre las dimensiones de salida de cada capa (*o*) considerando los parámetros de entrada (*i, k, p, s*), según las ecuaciones en (2-2), se definen las distintas relaciones posibles, junto con su número correspondiente asociado.

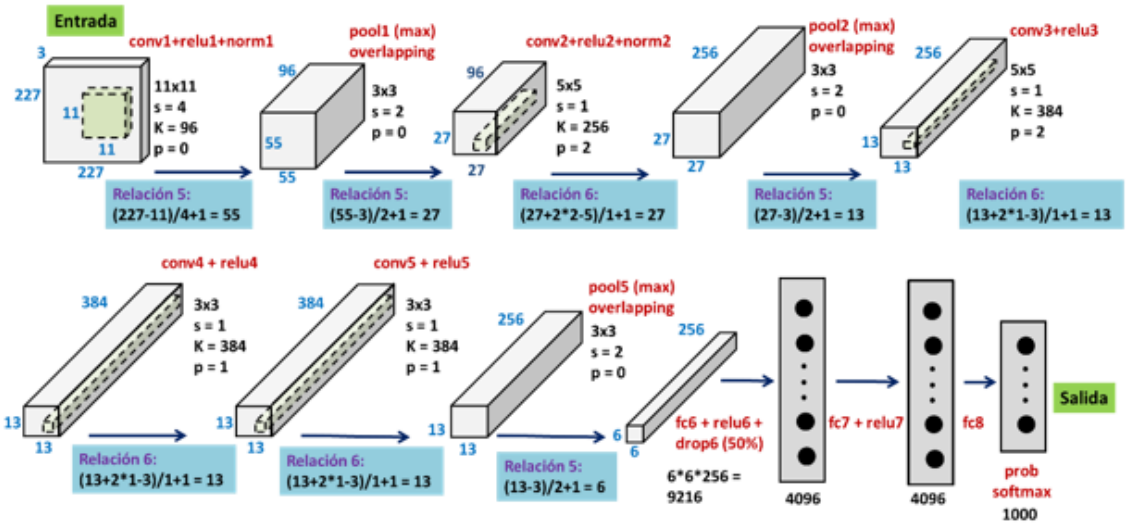


Figura 2-4. Modelo de red AlexNet

En algunas implementaciones, Matlab (2022b), tras la función *relu7* se realiza una operación de *dropout* del 50% modificando las dimensiones de la salida de esta capa.

2.2.2 GoogLeNet

La red ganadora del concurso ILSVRC 2014 fue GoogLeNet (también conocida como *Inception V1*) de Google (BVLC GoogLeNet Model, 2021), inspirada en el trabajo de Szegedy y col. (2014), habiendo logrado una tasa de error de 6.67%. Se trata de un resultado muy cercano al nivel humano, de modo que los organizadores del desafío necesitaron de la intervención experta. Resulta que, en realidad, la evaluación por parte del experto era bastante difícil de realizar, requiriendo algún entrenamiento adicional para determinar la precisión de GoogLeNet. La red usó una red neuronal convolución inspirada en LeNet, pero añadiendo un elemento novedoso denominado módulo *inception*. Un módulo *inception* (Inc) se basa en varias convoluciones relativamente pequeñas para reducir drásticamente el número de parámetros. Se utilizó la normalización por lotes (*batch*), distorsiones de imagen y RMSProp como método de optimización, basado en un aprendizaje adaptivo para cada dimensión. La arquitectura del modelo GoogleLeNet consiste en una RNC de 22 capas de profundidad, que consigue una reducción en el número de parámetros de 60 millones (AlexNet) a 4 millones. De esas 22 capas, 9 son de tipo *inception* de distintas categorías. En total, el número de capas es de 144, donde cada una de las 9 capas *inception* contiene la estructura mostrada en la Figura 2-5, en la que se han incorporado las unidades ReLU presentes en el módulo. El modelo completo propuesto por (Szegedy, y col. 2014) es el mostrado en la Figura 2-6, donde 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 del tamaño de la ventana; Normalización (LRN, *Local Response Normalization*); capas totalmente conectadas (FC) y por supuesto los módulos *Inception* (*Inc*), en este caso V1 que constituyen la parte nuclear de este tipo de redes. En las capas de convolución y *pooling* se indica también el desplazamiento (stride), que en este caso se expresa entre paréntesis con el símbolo s seguido del correspondiente valor de desplazamiento en el caso de tipo '*same*', y con el símbolo V también con el correspondiente valor de desplazamiento, en el caso de tipo '*valid*'. Como puede comprobarse, este esquema presenta dos redes extra, que son exactamente sendos clasificadores auxiliares, y constan de un *pooling* promediado de dimensión 5×5 y s = 3 de tipo V que proporciona salidas de tamaños 4×4×512 y

4×4×528 respectivamente. Le sigue una capa de convolución 1×1 con 128 filtros para reducción de la dimensionalidad y una unidad ReLU. Continúa una unidad *dropout* con capas totalmente conectadas finalizando con unidades *softmax* (Softmax0 y Softmax1). La salida final de la red también es una unidad *softmax* (Softmax2).

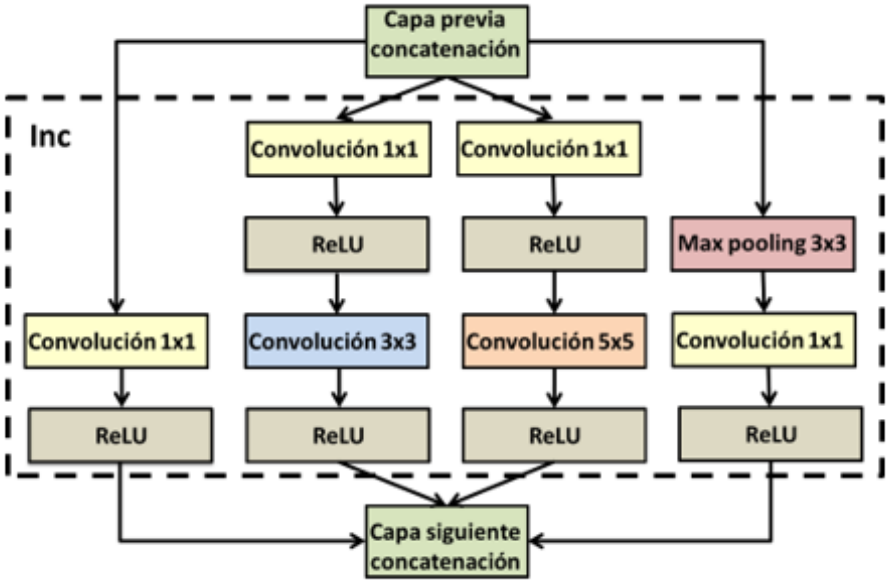


Figura 2-5. Modelo inception con incorporación de unidades ReLu

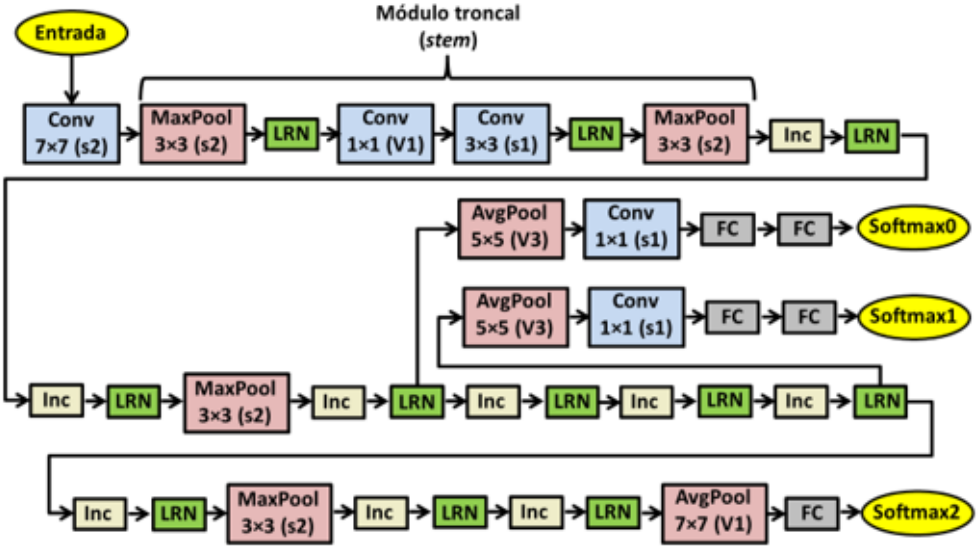


Figura 2-6. Modelo completo GoogLeNet (inception V1)

Capítulo 3 - Diseño de la aplicación

Una vez introducida la parte teórica relativa a las RNC, se procede a describir en orden, el diseño detallado de la arquitectura utilizada para el desarrollo de la aplicación, la metodología y gestión del proyecto, así como las herramientas y recursos usados.

3.1 Arquitectura

El desarrollo de la solución puede dividirse en tres módulos principales conectados entre sí, para los cuales es necesario disponer de un ordenador y de un dispositivo móvil con conexión a internet para su correcto funcionamiento. En la Figura 3-1 se puede observar el esquema seguido para cada módulo, el cual se describe en detalle a continuación.

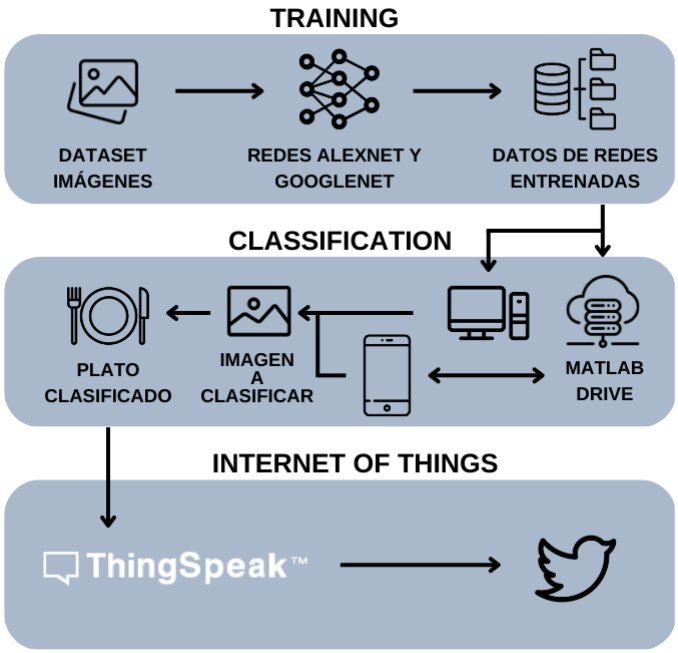


Figura 3-1. Esquema módulos aplicación

- **Módulo Training:** a partir de un conjunto de imágenes previamente seleccionadas y clasificadas según el nombre del plato de comida al que pertenecen, se reentrenan las redes elegidas, en este caso, AlexNet y GoogLeNet, para que sean capaces de identificar los platos deseados.

Este proceso genera un archivo que se almacena localmente en el ordenador y que contiene el nuevo modelo de entrenamiento de cada red para que, llegado el momento de la clasificación, no sea necesario volver a entrenar la red. Este archivo, también será almacenado en la plataforma de MatLab Drive.

- **Módulo Classification:** dependiendo del dispositivo desde el que se desee realizar la clasificación, se utilizará el archivo de entrenamiento almacenado en memoria, en el caso de hacerlo desde un ordenador, o el almacenado en la plataforma MatLab Drive, si se realiza desde un dispositivo móvil. Tomamos y seleccionamos la imagen a clasificar. Una vez se obtenga el resultado, este se mostrará por pantalla y, si nos encontramos desde un dispositivo móvil, la información se enviará a ThingSpeak.
- **Módulo Internet of Things**: compuesto por la plataforma ThingSpeak, se encarga de recibir y almacenar la información del plato de comida clasificado y de activar un disparador (programado en esta plataforma) cada vez que se produce este proceso, el cual se encarga de publicar un tweet en la plataforma Twitter con la información calórica del plato.

Con respecto a la arquitectura del proyecto, indicar que ésta se basa en un modelo multicapa, permitiendo así desacoplar las distintas partes con el objetivo principal de facilitar el mantenimiento y la escalabilidad del mismo de cara a futuras mejoras e implementaciones. Las distintas capas de este desarrollo aparecen explicadas a continuación.

- **Capa de lógica de negocio:** se encarga de ejecutar todas las operaciones necesarias para que la aplicación funcione correctamente, incluyendo el procesado de imágenes para su redimensionamiento, el entrenamiento de las redes siguiendo los parámetros establecidos y la clasificación de los platos de comida.
- **Capa de presentación:** se encarga de ofrecer una interfaz gráfica amigable con la que el usuario puede interactuar.

- **Capa de integración o de datos:** se encarga del almacenamiento y del acceso a los datos necesarios por la aplicación en cada momento. En el caso del diseño propuesto para este proyecto en la Figura 3-1, no es necesario el uso de una base de datos para almacenar la información, sino que se utilizarán archivos almacenados tanto en memoria como en la plataforma online MatLab Drive.

A continuación, se explica el diagrama de casos de uso que aparece en la Figura 3-2, y que reúne todas las operaciones que lleva a cabo la aplicación.

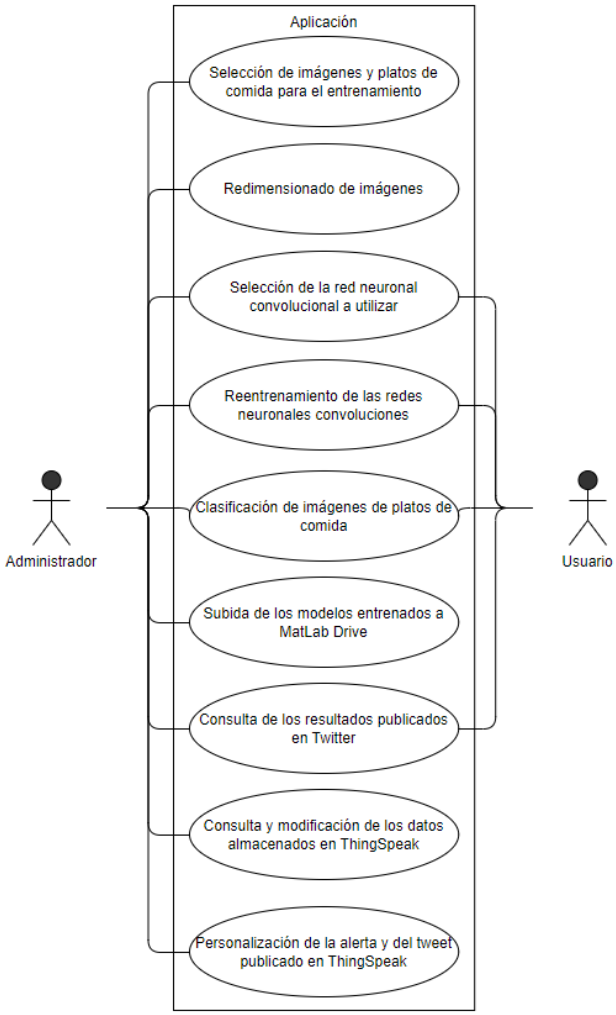


Figura 3-2. Diagrama de casos de uso

En dicho diagrama se observan dos roles distintos, el de administrador y el de usuario. El administrador, al ser el encargado de gestionar la aplicación completa, tiene

acceso a todas las funciones de la misma, que se listan a continuación: selección de las imágenes y los platos de comida para el entrenamiento, redimensionado de imágenes, selección de la red neuronal convolucional a utilizar para el entrenamiento o para la clasificación, reentrenamiento de las redes neuronales convolucionales con el banco de imágenes existente, subida de los modelos reentrenados a MatLab Drive, consulta de los resultados publicados en Twitter, consulta y modificación de los datos almacenados en ThingSpeak y personalización de la alerta y del tweet publicado en ThingSpeak. Por otro lado, el usuario tiene acceso a las funciones necesarias para el funcionamiento de la aplicación, lo que implica tener acceso a todas las funciones salvo a la selección de imágenes y de platos de comida para el entrenamiento, al redimensionado de imágenes, a la subida de modelos reentrenados a MatLab Drive, a la consulta y modificación de los datos almacenados en ThingSpeak y a la personalización de la alerta y del tweet publicado en ThingSpeak.

3.1.1 Versión de escritorio

En la versión de escritorio se pueden ejecutar todas las funciones mostradas en la Figura 3-2, salvo las relacionadas con la parte de Twitter y ThingSpeak. A continuación, se detalla más en profundidad cada una de ellas.

3.1.1.1 Selección de imágenes y platos de comida para el entrenamiento

Para poder realizar el entrenamiento de las redes neuronales convolucionales, es necesario disponer de un banco de imágenes lo suficientemente amplio para cada plato de comida que queramos poder clasificar. En esta ocasión, se han seleccionado 300 imágenes para cada uno de los diez platos distintos elegidos, que pueden encontrarse en la Figura 3-3 mostrada abajo.

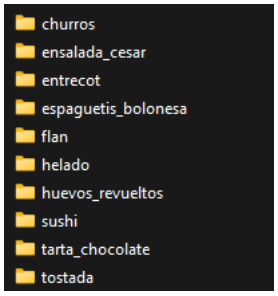


Figura 3-3. Platos de comida seleccionados

A continuación, en la Figura 3-4 se expone una imagen representativa por cada plato para mostrar un ejemplo de las imágenes con las que vamos a trabajar.

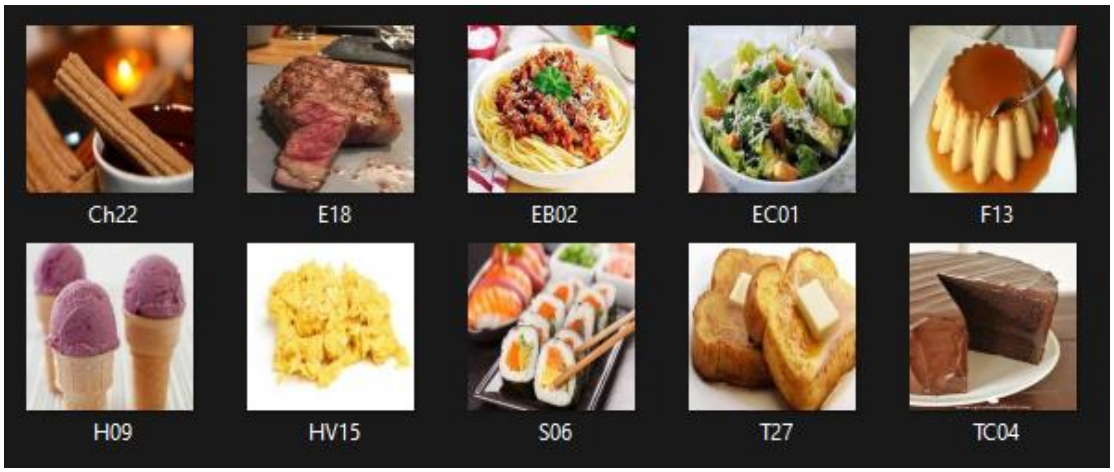


Figura 3-4. Ejemplos de imágenes aleatorias de los platos de comida

3.1.1.2 Redimensionado de imágenes

Dependiendo del modelo de red a utilizar para el entrenamiento, las imágenes usadas tienen que cumplir un requerimiento de tamaño, en el caso de AlexNet, 227 x 227 x 3 pixeles y en el caso de GoogLeNet, 224 x 224 x 3 pixeles. Para la implementación de esta funcionalidad, se ha decidido incluirla directamente en la operación de reentrenamiento para que, automáticamente redimensione todas las imágenes antes de comenzar de forma que, en el caso de existir alguna imagen no redimensionada, no desemboque en un error de ejecución.

3.1.1.3 Selección de la red neuronal convolucional a utilizar

Tanto para realizar el proceso de entrenamiento de la red, Figura 3-5, como a la hora de requerir la clasificación de una imagen, Figura 3-6, el usuario dispone de la libertad para seleccionar si desea usar la red de AlexNet o la de GoogLeNet.

Figura 3-5. Selección de la red en la clasificación

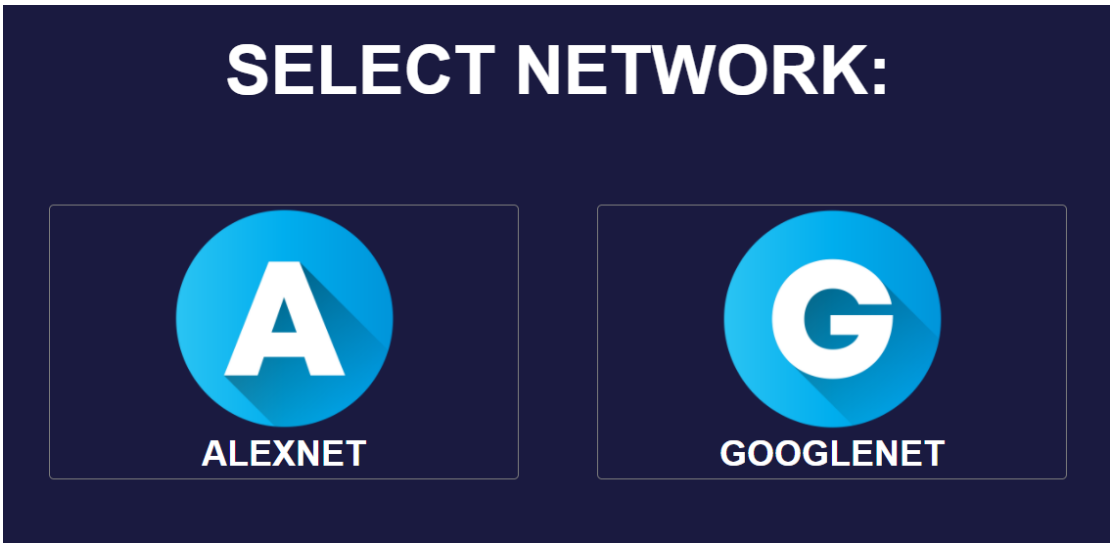


Figura 3-6. Selección de la red en el entrenamiento

3.1.1.4 Reentrenamiento de las redes neuronales convolucionales

Una vez seleccionado el tipo de red que se desea entrenar según el interfaz mostrado en la Figura 3-6, se abre una nueva ventana, Figura 3-7, que permite al usuario modificar múltiples parámetros de configuración y seleccionar las ventanas de proceso que quiere que aparezcan a medida que se realiza el entrenamiento.

Figura 3-7. Ventana para realizar y configurar el entrenamiento de la red

El objetivo de cada uno de los parámetros que pueden configurarse se describe a continuación:

- **MiniBatchSize:** número de imágenes del dataset que se utilizan en cada iteración del entrenamiento.
- **MaxEpochs:** número de épocas máximas, en cada época las imágenes actualizan sus pesos.
- **InitialLearnRate:** tasa de aprendizaje inicial para realizar el entrenamiento. Si el valor es más alto de lo debido, puede resultar en errores en los resultados de clasificación, si, por el contrario, es más baja de lo debido, puede provocar mayores tiempos para completar el entrenamiento.
- **ValidationFrequency:** número de iteraciones entre cada validación que se realiza de la red.
- **OptimizationMethod:** posibles métodos de optimización para entrenar las redes neuronales convolucionales: Adam o Sgdm.
- **NumTrainImages:** número de imágenes del dataset de cada clase de plato de comida que se seleccionan de forma aleatoria para realizar el entrenamiento.
- **NumValidationImages:** número de imágenes del dataset de cada clase de plato de comida que se seleccionan de forma aleatoria para realizar una validación de los resultados en el entrenamiento.

El contenido y objetivos de las diferentes ventanas que pueden mostrarse de forma opcional se explicarán más adelante en esta memoria.

3.1.1.5 Clasificación de imágenes de platos de comida

Una vez realizado el entrenamiento de las redes, el usuario puede empezar a clasificar imágenes desde el interfaz mostrado en la ventana de la Figura 3-8 pulsando sobre el botón verde en la mitad derecha, no sin antes, haber seleccionado desde el desplegable localizado en la parte izquierda, el tipo de red que desee utilizar y la imagen que quiera utilizar, localizada en su ordenador.

Una vez comienza el proceso, es posible conocer el estado actual del mismo gracias a una barra de progreso que muestra la operación actual y el porcentaje. Al finalizar, se puede consultar el nombre del plato de comida obtenido como resultado de la clasificación.

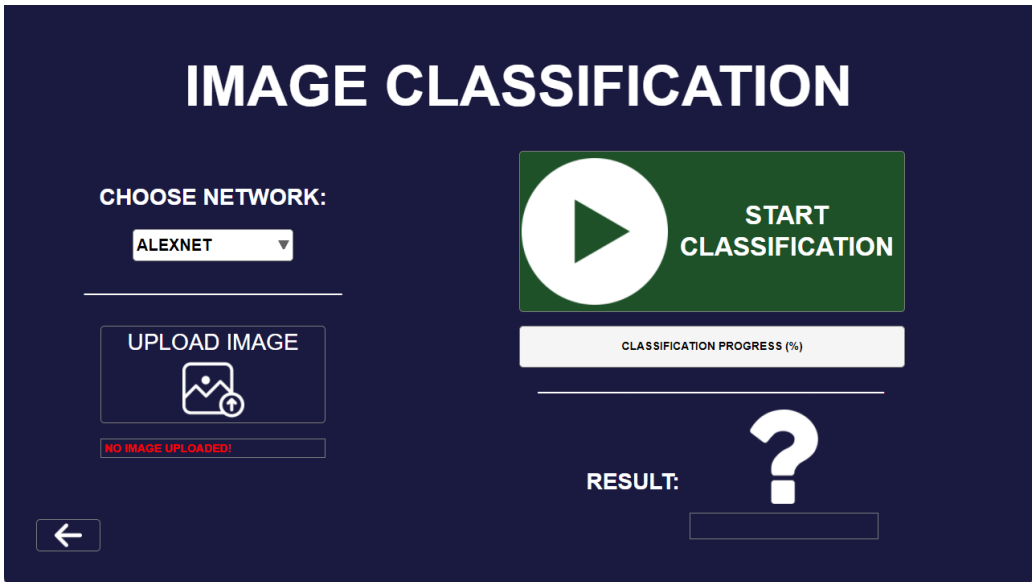


Figura 3-8. Ventana para realizar la clasificación de platos de comida

3.1.1.6 Subida de los modelos entrenados a MatLab Drive

Para poder llevar a cabo una clasificación de alimentos desde un dispositivo móvil debido a sus limitaciones, es necesario subir a la plataforma MatLab Drive el archivo de modelo de red ya entrenado de cada una de las redes usadas. Para poder ejecutar este proceso disponemos de la extensión para MatLab, Drive Connector, que permite sincronizar con la nube ciertos archivos seleccionados que se encuentran almacenados en nuestro ordenador.

3.1.2 Versión móvil y web

La versión móvil, al tratarse de una versión reducida de la aplicación, únicamente permite realizar la función de clasificación de platos de comida y, es la encargada de ejecutar todas las operaciones de ThingSpeak, programadas específicamente en esta plataforma, así como la gestión de la plataforma Twitter al necesitar de una conexión a internet para su ejecución.

3.1.2.1 Clasificación de imágenes de platos de comida

Para realizar la clasificación desde un dispositivo móvil, es necesario que este disponga de cámara, conexión a internet y tenga instalada la aplicación de MatLab Mobile. Además, en esta última es necesario que iniciar sesión con la cuenta de usuario donde se han alojado previamente los modelos de red entrenados y el código necesario parar aplicar las funcionalidades necesarias.

Para realizar una clasificación, se requiere disponer de los dos comandos de la Figura 3-9, que en definitiva son programas MatLab alojados en MatLab Drive. En primer lugar, hay que ejecutar el comando InicioCamaraChristian, que permite comprobar e inicializar el estado de la cámara asociada con el dispositivo móvil a utilizar para, a continuación, usando el comando CamaraClasificacionChristian, abrir la cámara para poder tomar una foto, redimensionar la imagen a las dimensiones requeridas por cada modelo de red y finalmente realizar la clasificación.

```
>> InicioCamaraChristian
>> CamaraClasificacionChristian
```

Figura 3-9. Comandos para realizar la clasificación desde un dispositivo móvil

En la Figura 3-10 se puede ver el resultado que aparece por la pantalla de la aplicación después de haber realizado la clasificación sobre una imagen de un helado. Tal y como se puede observar, no solo realiza la clasificación sobre una de las redes, sino que, al no disponer de una interfaz gráfica más amigable, se ha decidido que la haga sobre ambas redes y muestre ambos resultados. Se identifica, por tanto, cada uno de los modelos de red utilizados, así como la categoría a la que pertenece la imagen concreta, según el tipo de comida y según el modelo de red. En ambos casos, la clasificación ha sido correcta, coincidiendo con la categoría esperada.

Figura 3-10. Resultado de una clasificación realizada desde el móvil

3.1.2.2 Consulta y modificación de los datos almacenados en ThingSpeak

La plataforma ThingSpeak permite almacenar y analizar datos, así como realizar ciertas operaciones con ellos, tal como la publicación de tweets. Para poder incorporarlo a la aplicación, dentro de la plataforma se necesita crear un canal, en nuestro caso llamado Calories, con la configuración mostrada en la Figura 3-11. Estos canales permiten almacenar información en distintos campos modulares, como si de una tabla se tratase. Para nuestro caso, crearemos tres campos, correspondientes al nombre del plato (name) que vamos a almacenar, así como sus calorías (kcal) y el número de veces que se ha clasificado ese plato desde el inicio (numScans).

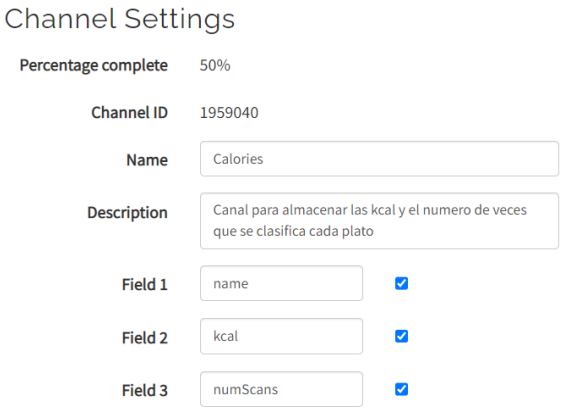


Figura 3-11. Configuración del canal de ThingSpeak

Para rellenar la información relativa a estos campos, se utiliza la información que aparece en la Tabla 3-1. Datos subidos a ThingSpeak, aunque ThingSpeak permite introducir los datos manualmente desde la misma, es posible hacerlo desde la propia aplicación de MatLab de una forma mucho más cómoda y automática haciendo uso de la API del canal para enviar la información y que esta se quede almacenada.

Tabla 3-1. Datos subidos a ThingSpeak

Plato de comida	Calorías (kcal)	NumScans
Churros	360	0
Ensalada cesar	44	0
Entrecot	222	0
Espaguetis	123	0
Flan	133	0
Helado	213	0
Huevos Revueltos	229	0
Sushi	103	0
Tarta chocolate	440	0
Tostada	249	0

Una vez se envía la información, se puede acceder al canal, observándose según se indica en la Figura 3-12. Canal creado de ThingSpeak tras insertar datoscómo, no solo ha almacenado la información, sino que ha creado dos gráficas de líneas representativas, una correspondiente al número de calorías y otra al número de veces que se ha escaneado un plato, permitiendo así obtener una visión general mucho más rápida y cómoda.

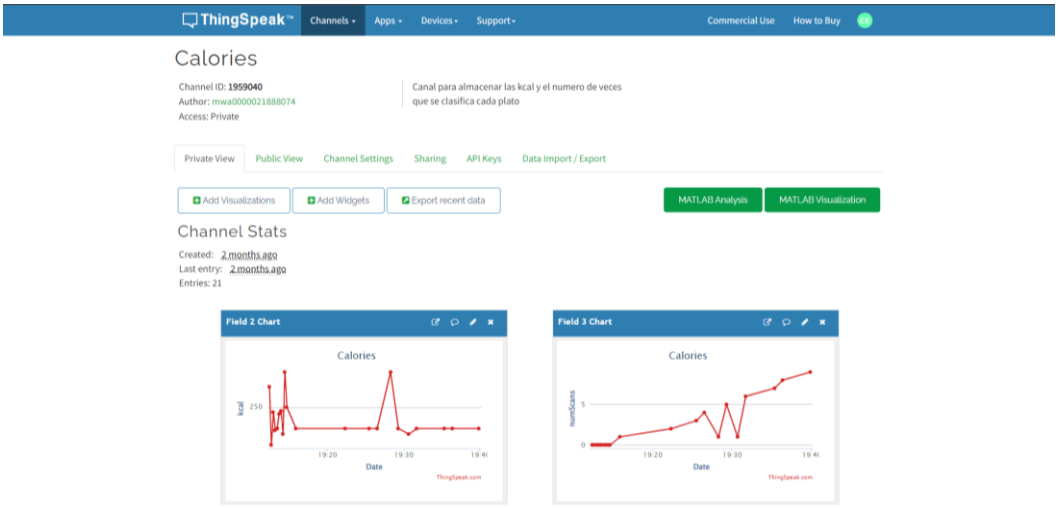


Figura 3-12. Canal creado de ThingSpeak tras insertar datos

3.1.2.3 Personalización de la alerta y del tweet publicado en ThingSpeak

Además de lo comentado en el punto anterior, ThingSpeak permite conectarse a una cuenta de Twitter de una forma muy sencilla, gracias a una aplicación dentro de la plataforma llamada ThingTweet, que se encarga de generar una API Key, como la mostrada en la Figura 3-13, y que será necesaria para poder realizar la publicación de tweets más adelante.

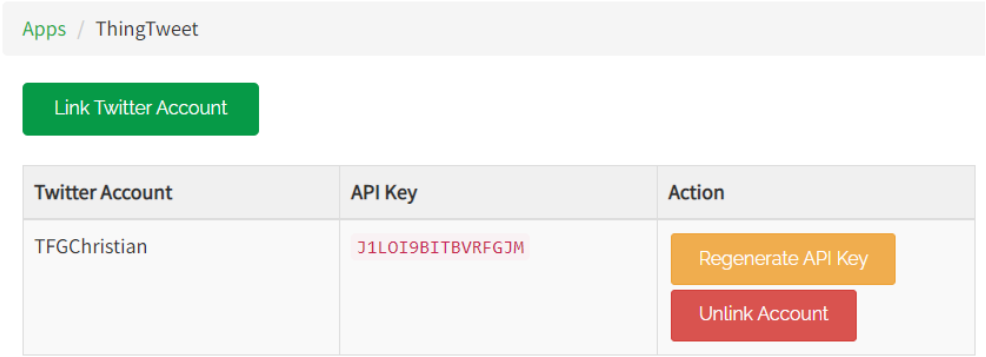


Figura 3-13. Conexión a Twitter desde ThinsSpeak

Otra aplicación disponible dentro de la plataforma llamada ThingHTTP permite crear peticiones HTTP. La configuramos tal y como se muestra en la Figura 3-14 para realizar una petición de tipo POST para que se publique un tweet con la información almacenada en distintos campos de nuestro canal, en este caso, el nombre del plato de comida, sus calorías y la fecha de escaneo.

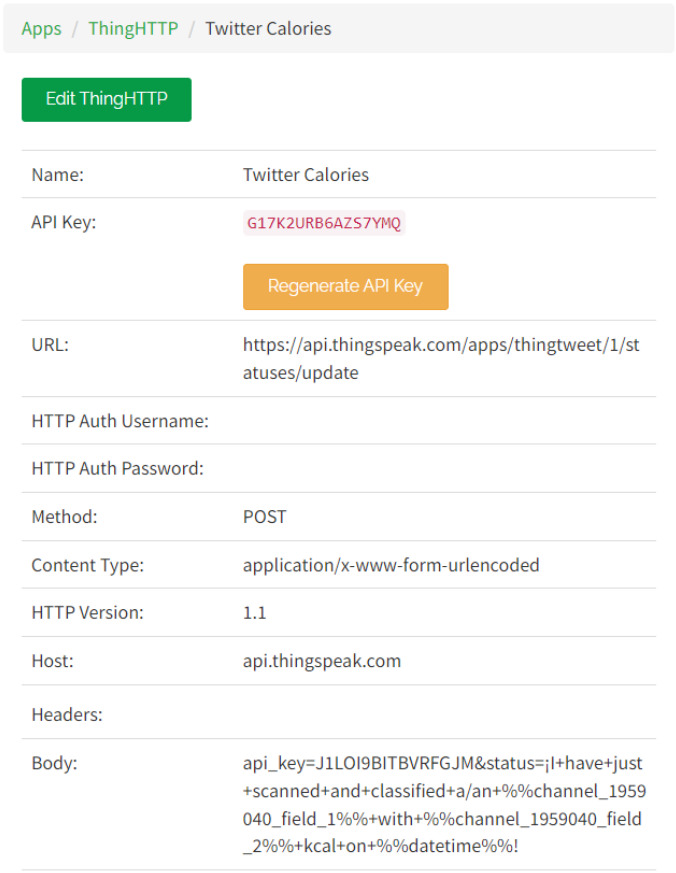


Figura 3-14. Configuración de ThingHTTP para la publicación de tweets

Para que la petición creada anteriormente se ejecute cada vez que se requiera, ThingSpeak dispone de una aplicación llamada React, que es necesario configurar y que permite establecer un disparador para que, cuando se cumpla una condición, en nuestro caso que el número de veces que se ha escaneado el plato que acaba de recibir de la clasificación sea mayor que cero, ejecute el ThingHTTP configurado con anterioridad. En la Figura 3-15 se puede observar el React programado con los parámetros y valores que se indican. Uno de los campos a considerar es el relativo a la frecuencia a la que se activa esta funcionalidad, en este caso cada vez que se inserta el dato (*on data insertion*) en el campo del canal especificado. El tipo de dato es otro de los componentes a especificar. Finalmente, se identifican tanto la fecha de creación como de última ejecución.

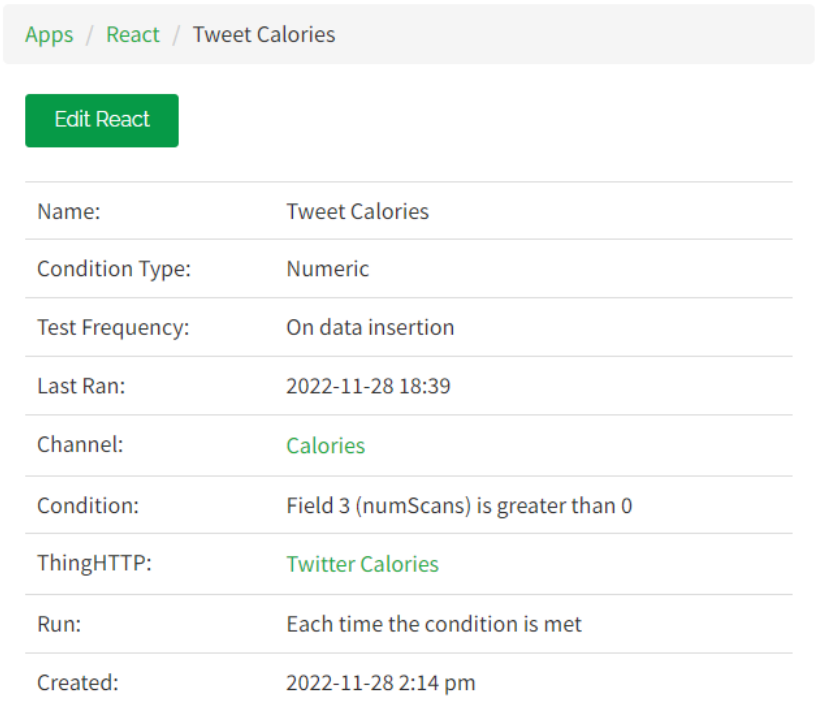


Figura 3-15. Configuración del React

3.1.2.4 Consulta de los resultados publicados en Twitter

Una vez configurados todos los recursos anteriores, el usuario puede realizar una clasificación desde su dispositivo móvil e inmediatamente después, acceder a su cuenta de Twitter para observar el resultado.

En la Figura 3-16 se muestra una publicación de ejemplo de la cuenta de Twitter usada para la demostración, @TFGChristian, tras haber realizado una fotografía a un flan.

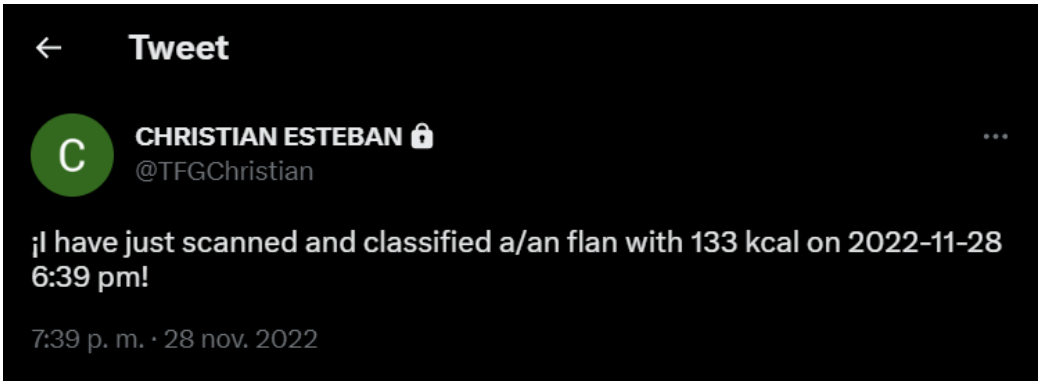


Figura 3-16. Tweet publicado tras realizar la clasificación de un plato de comida

3.2 Metodología seguida para la gestión del proyecto

Al tratarse de un trabajo unipersonal, las dificultades en el ámbito de la organización con respecto a un trabajo en equipo se reducen de forma significativa. Aun así, debido a las características de la aplicación, el desarrollo de este proyecto se ha caracterizado por considerar los siguientes principios propios de la metodología agile.

1. El trabajo total se divide y reparte para poder ser realizado por partes y en fases temporales de similar duración entre ellas, dos semanas aproximadamente en este caso.
2. Capacidad de medir el progreso actual del desarrollo, mediante un tablero Kanban con la aplicación de escritorio Trello.
3. Cualquier problema, cambio o añadido al trabajo puede realizarse sin mucha dificultad.
4. El desarrollo del trabajo se realiza de forma incremental, agregando funcionalidad al proyecto en cada fase, de tal forma que pueda ir ejecutándose.
5. Las tareas en que se divide el trabajo deben ser lo más simples y sencillas posibles.
6. Preocupación por la calidad de la aplicación finalizada.

Antes de comenzar con el desarrollo del trabajo, se llevaron a cabo varias reuniones con el equipo docente para determinar y clarificar el ámbito y objetivos del proyecto en su conjunto.

Una vez determinados estos y aproximadamente una vez cada quince días, se establecía una reunión de seguimiento para poder presentar los objetivos y el progreso alcanzado hasta ese momento. Estas reuniones también sirvieron para poder esclarecer posibles dudas o problemas que hubiesen surgido, así como para determinar posibles ampliaciones al trabajo.

3.3 Herramientas y recursos

A continuación, se enumeran y describen las herramientas y recursos utilizados para el desarrollo de la aplicación.

- **Matlab:** plataforma de programación de alto nivel para el análisis de datos y la creación de modelos algorítmicos principalmente, permite descargar módulos para ampliar distintas implementaciones y características. Se ha instalado en su versión R2022b de escritorio en un ordenador con sistema operativo Windows 11, con los módulos Image Processing Toolbox, Deep Learning Toolbox, Deep Learning Toolbox Model for AlexNet Network y Deep Learning Toolbox Model for GoogLeNet Network. También permite la creación de interfaces gráficas con MatLab Design App.
- **MatLab Drive:** almacenamiento compartido en la nube de archivos para usar en la versión de escritorio o móvil de MatLab.
- **MatLab Mobile:** aplicación móvil de MatLab, permite acceder y ejecutar archivos de código almacenados en MatLab Drive.
- **ThingSpeak:** plataforma relacionada con el internet de las cosas que permite recolectar, almacenar y procesar datos de múltiples dispositivos. Puede conectarse de forma directa con MatLab para realizar análisis de los datos recibidos o para crear alertas y disparadores.
- **Git y GitHub Desktop:** herramienta de control de versiones para el código fuente del proyecto, acompañado con el cliente de escritorio de GitHub para gestionarlo de forma sencilla.
- **Flaticon:** base de datos gratuita con millones de iconos personalizables con posibilidad de descarga en múltiples formatos, usados en esquemas y en el diseño de la interfaz.
- **Microsoft Word:** procesador de textos para la realización de esta memoria usando la plantilla proporcionada.

- **Google Mail y Meet:** plataformas usadas para la comunicación y envío de información al tutor, así como para la realización de videollamadas para la discusión de temas relacionados con el trabajo y la resolución de dudas.
- **Trello:** aplicación usada en la gestión del proyecto, permite la creación de un tablero Kanban para realizar un seguimiento de todas las tareas a realizar para cumplir con los objetivos propuestos.
- **Canva:** web con múltiples herramientas de diseño y creación de composiciones, utilizada para realizar diseños básicos de la interfaz de usuario, sirviendo como guía de cara a su futura implementación.
- **Visual Paradigm:** plataforma online que ofrece múltiples herramientas para la gestión de proyectos, entre ellas, creación de diagramas, generación de informes, modelado de software y generación de código.

Capítulo 4 - Resultados obtenidos

En este capítulo se describen los resultados obtenidos en los distintos módulos de la aplicación según las funcionalidades descritas previamente.

4.1 Interfaz UI de la aplicación

Al iniciar la aplicación para su ejecución, aparece la pantalla de inicio mostrada en la Figura 4-1que nos da la bienvenida a la misma.

Figura 4-1. Pantalla de inicio

4.1.1 Menú principal

El botón START que aparecen en la Figura 4-2, conduce a la ventana del menú principal mostrada en la Figura 4-2, que permite elegir entre dos opciones:

- **Network training:** módulo encargado de configurar los parámetros de la red, así como de realizar el entrenamiento.
- **Image Classification:** módulo encargado de realizar una clasificación de una imagen.

Figura 4-2. Pantalla de menú principal

4.1.2 Entrenamiento de redes

Una vez seleccionado el botón Network Training de la Figura 4-2, se muestra una ventana como la mostrada en la Figura 4-3 para seleccionar qué modelo de red elegir para el entrenamiento, disponemos de dos opciones, AlexNet o GoogLeNet.

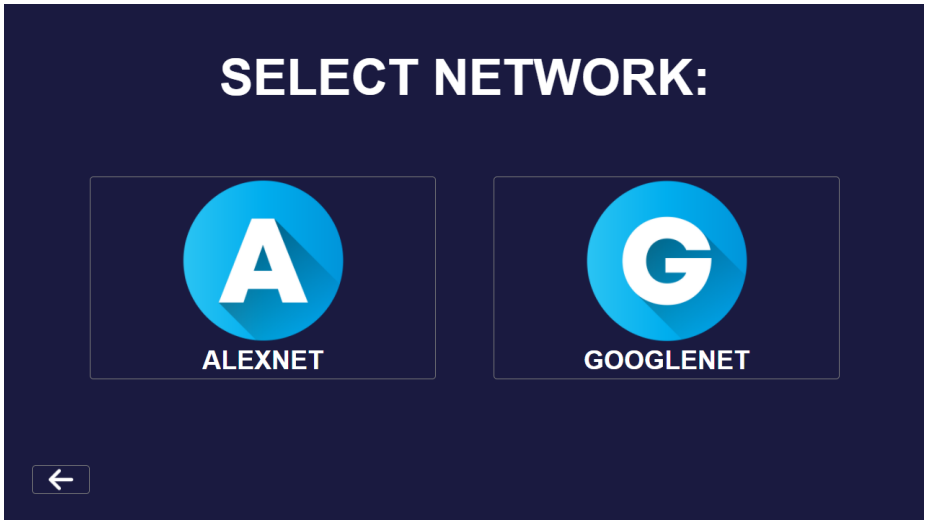


Figura 4-3. Pantalla de selección de la red a entrenar

4.1.2.1 AlexNet

Si se elige la opción AlexNet de la Figura 4-3, se muestra una ventana como la de la Figura 4-4.

Figura 4-4. Pantalla de entrenamiento para AlexNet

4.1.2.2 GoogLeNet

Si se selecciona la opción GoogleNet de la Figura 4-3, se muestra una ventana como la de la Figura 4-5.

Figura 4-5. Pantalla de entrenamiento para GoogLeNet

Las pantallas del entrenamiento de ambas redes comparten la mayoría de las opciones de configuración, que se explican a continuación, usando como referencia las secciones enmarcadas con colores de la Figura 4-7:

- **Sección roja:** panel con los distintos parámetros de configuración que permite la red, con posibilidad de introducir el valor deseado para cada uno de ellos.
- **Sección verde:** panel de selección para seleccionar las ventanas quieren mostrarse a la hora de realizar el entrenamiento de la red.
- **Sección azul:** botón para comenzar con el entrenamiento de la red elegida.
- **Sección amarilla:** botón para eliminar el entrenamiento almacenado para la red elegida.
- **Sección morada:** barra de progreso con distintas fases que muestra el progreso y porcentaje actual del entrenamiento de la red hasta su finalización.

Figura 4-6. Secciones dentro de la pantalla de entrenamiento

4.1.3 Clasificación de imágenes

Una vez seleccionado el botón Image Classification de la Figura 4-2, se muestra la ventana mostrada a continuación en la Figura 4-7.

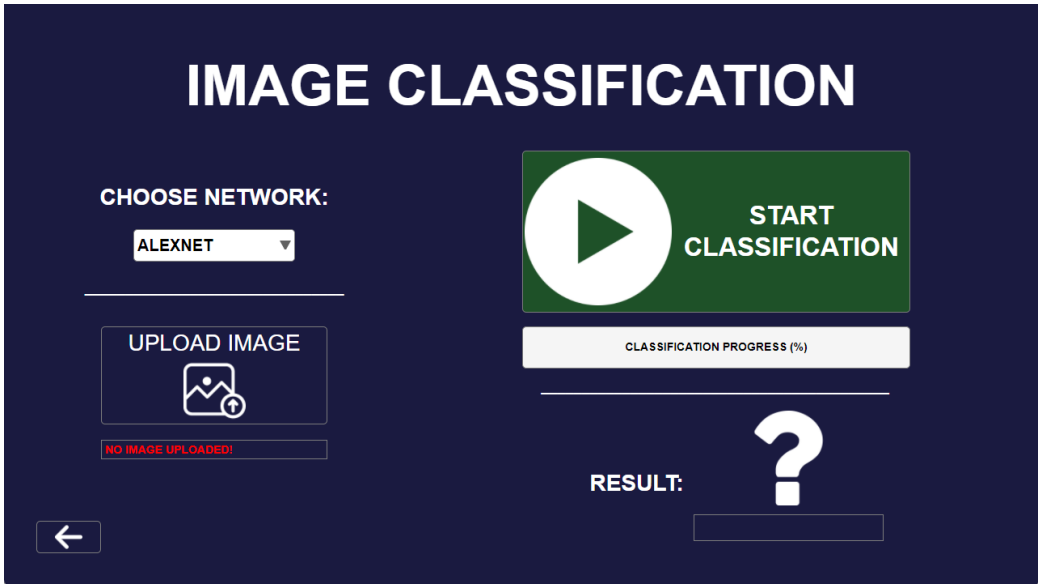


Figura 4-7. Pantalla de clasificación de platos de comida

Para la explicación de las distintas secciones dentro de esta pantalla, se utilizan como referencia las secciones de colores de la Figura 4-8, mostrada a continuación:

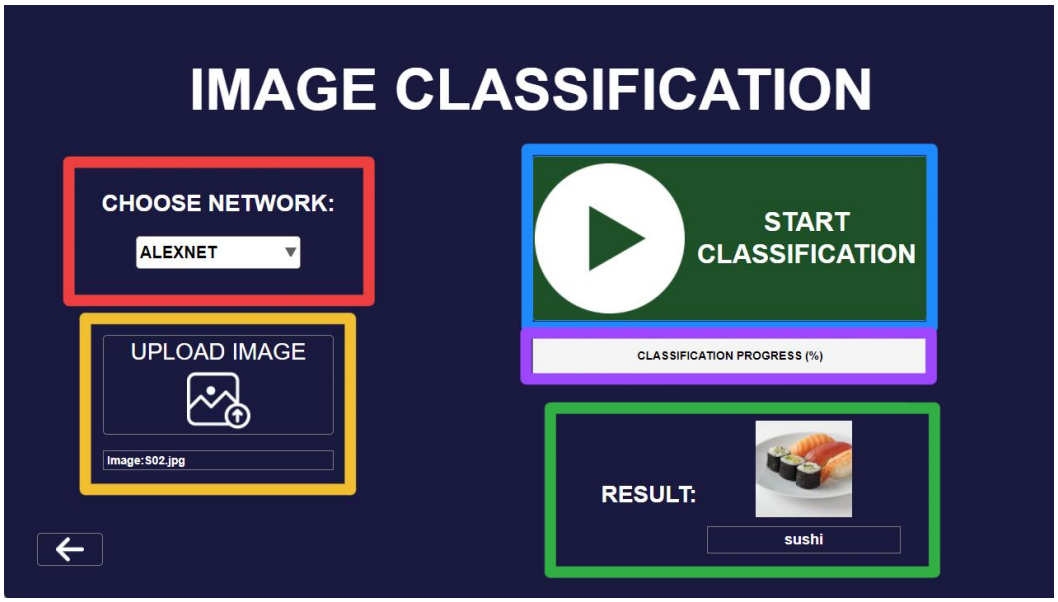


Figura 4-8. Secciones dentro de la pantalla de clasificación

- **Sección roja:** desplegable que permite seleccionar entre ambas redes disponibles, para usarla en la clasificación.

- **Sección amarilla:** la parte superior consta de un botón que permite seleccionar una imagen almacenada en el ordenador y la parte inferior muestra el nombre del archivo seleccionado.
- **Sección azul:** botón que permite empezar con la clasificación de la imagen seleccionada usando la red elegida.
- **Sección morada:** barra de progreso que permite ver el estado y porcentaje de evaluación actual de clasificación.
- **Sección verde:** muestra en la parte superior la imagen clasificada y en la parte inferior el nombre del plato que comida que ha detectado.

4.2 Dataset de imágenes

Para el entrenamiento de ambas redes se han utilizado imágenes procedentes del Dataset de la competición iFood – 2019 at FGVC6 (2019) almacenado en la plataforma Kaggle.

El Dataset contiene 251 carpetas, cada una lleva por nombre y pertenece a un plato de comida concreto. Dentro de cada una de las carpetas, se encuentran tres subcarpetas: Entrenamiento, validación y test. Entre todas ellas contienen más de 600 imágenes entre las tres, repartidas en una proporción del 65%, 25% y 10%, correspondientemente.

4.3 Entrenamiento

Para llevar a cabo el proceso de entrenamiento, este debe realizarse de forma individual para cada uno de los modelos de red. Gracias a la posibilidad de configurar cada modelo, como se puede ver en a la izquierda de la Figura 4-4 y Figura 4-5, se puede precisar la forma en que se realizará el entrenamiento. Predeterminadamente, se proporcionan unos valores por defecto, pero pueden modificarse según las preferencias del usuario. A continuación, se describe más en profundidad el proceso de entrenamiento para los dos modelos de red seleccionados, haciendo uso del módulo *Deep Learning Network Analyzer* de MatLab.

4.3.1 Modelo AlexNet

Antes de iniciar con el proceso de entrenamiento, el analizador del modelo mostrado en la Figura 4-9 (*Deep Learning Network Analyzer*) proporciona información sobre las distintas capas junto con sus conexiones, el número de pesos a aprender por el modelo (*total learnables*), en este caso 60.9M millones, el número de capas totales (25), así como la inexistencia de avisos (*warnings*) o errores (*errors*), necesario para poder comenzar con el proceso.

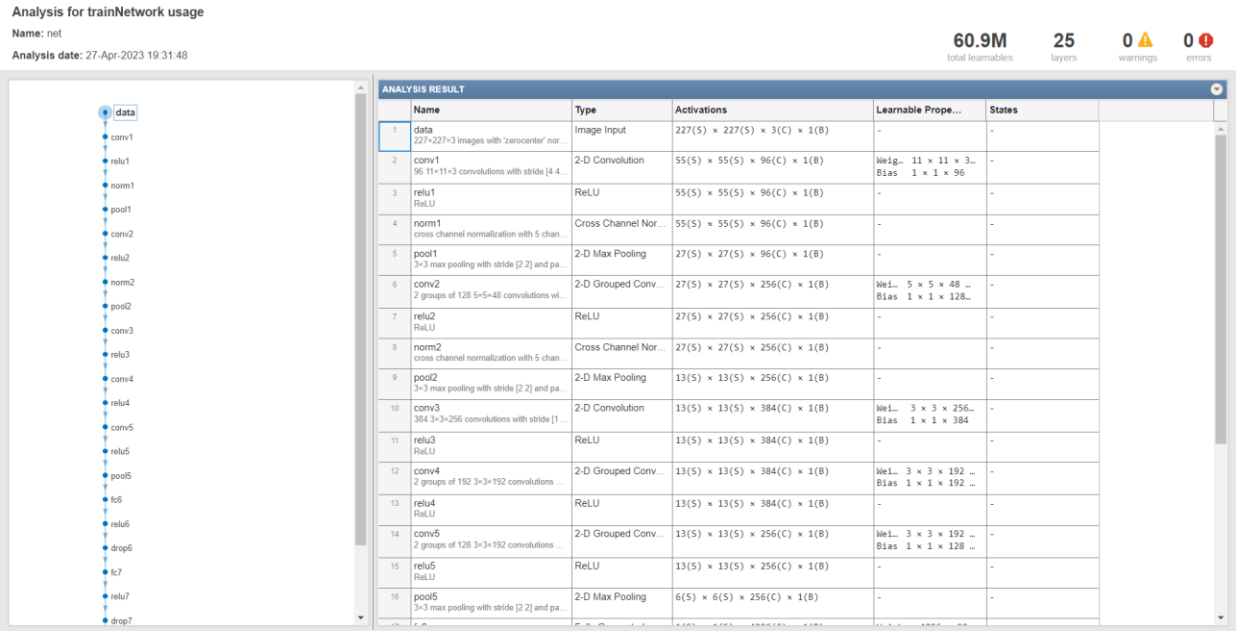


Figura 4-9. Analizador del modelo de red AlexNet

En primer lugar, se ha llevado a cabo un entrenamiento con los parámetros por defecto configuradas en la aplicación, tal y como puede observarse en la Figura 4-10. Se puede percibir cómo la función de precisión (*Accuracy*), gráfico superior azul, comienza con valores bajos, pero asciende rápidamente hasta valores cercanos al 70% poco después de terminal el primer *epoch*. A partir de ahí, se consigue mantener el valor, rozando el 80% hasta el final, en donde se logra un 78.89% de precisión. A pesar de no llegar al 100%, los resultados son aceptables. Conviene señalar que el valor 100% representa el ideal. Se puede experimentar, para tratar de conseguir mejores resultados de precisión, modificando los valores de ciertos parámetros del modelo tales como el número de iteraciones o el número de imágenes del *dataset*.

En cuanto a la función de pérdida (*loss*), gráfico inferior naranja, se puede observar cómo cae rápidamente durante todo el *epoch 1* hasta conseguir cierta estabilidad a partir del *epoch 4*, logrando valores por debajo al uno. La función de pérdida tiene como objetivo medir el error de las muestras, por lo que un valor bajo cercano a cero, siendo este el ideal, significa un buen ajuste.

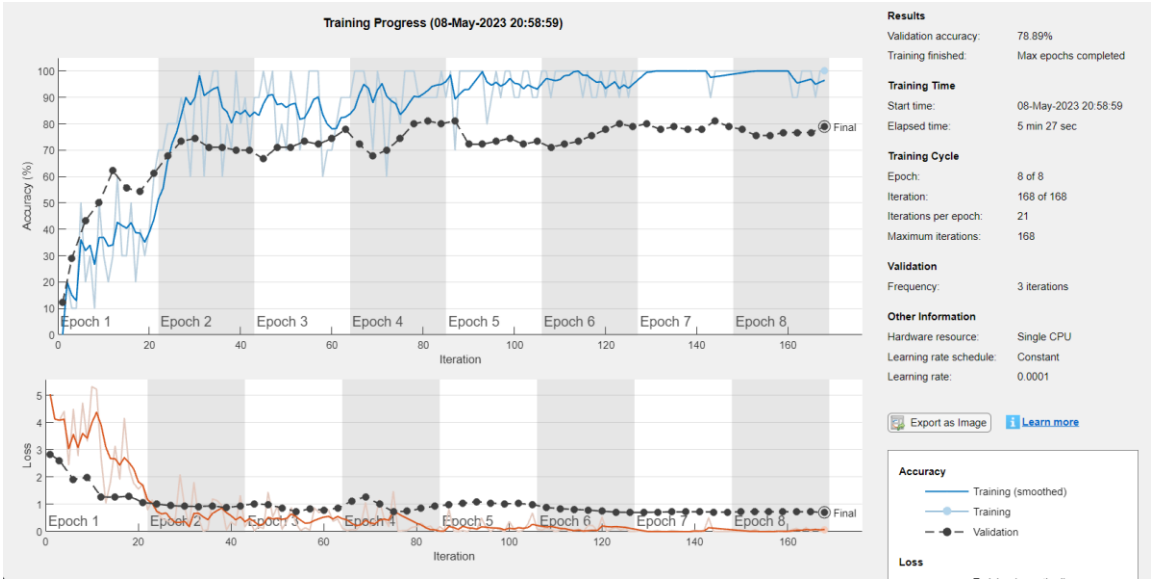


Figura 4-10. Entrenamiento AlexNet, valores por defecto

En la Figura 4-11 puede observarse una representación gráfica de los pesos (*weights*) aprendido en la primera capa de convolución AlexNet tras realizar el entrenamiento. Si un peso tiene un valor nulo, su color es negro, por el contrario, un color más claro significa un valor de peso mayor.

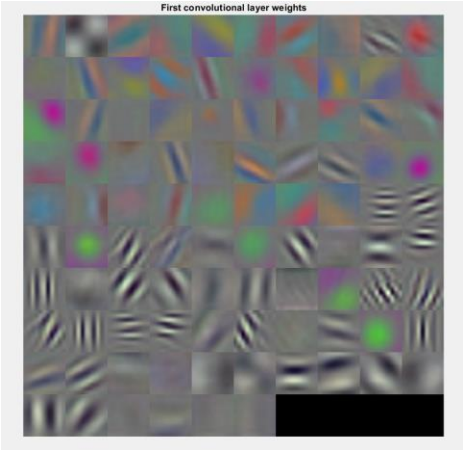


Figura 4-11. Pesos de la primera capa convolucional, valores por defecto AlexNet

Para comprobar si es posible mejorar el valor anterior de precisión modificando los parámetros del modelo, se realizar el entrenamiento de nuevo con los ajustes mostrados en la Figura 4-12. Se ha aumentado el número de *epochs* (*MaxEpochs*), y por tanto de iteraciones, hasta 15, siendo el valor por defecto 8, el número de imágenes de validación (*NumValidationImages*) hasta 10, siendo previamente 8, y el número de imágenes de entrenamiento, siendo ahora 66 y antes 49.

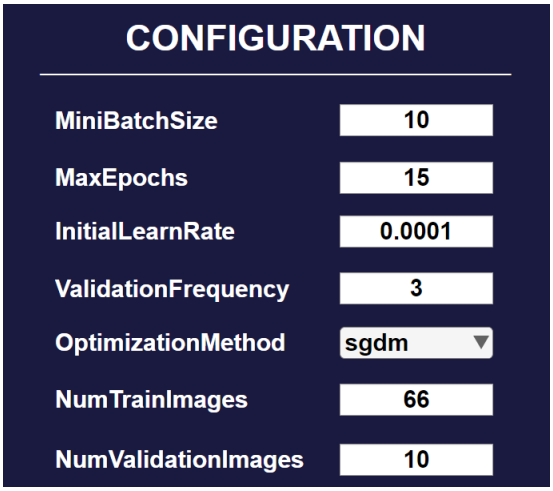


Figura 4-12. Valores priorizando precisión en AlexNet

Volviendo al analizador del modelo, Figura 4-13, se observa un comportamiento muy similar al de la Figura 4-10 con los valores por defecto. Rápidamente, la función de precisión sube hasta valores cercanos al 80% al terminar el *epoch* 2, para estabilizarse hasta el final del entrenamiento, en donde alcanza una precisión final de 81.11%. Con estos resultados solo queda decidir, dependiendo de la situación, si merece la pena dedicar un 40% aproximadamente más de tiempo, en lograr un 2.8% más de precisión.

La función de pérdida ha logrado de la misma forma estabilizarse en valores cercanos al uno nada más terminar el *epoch* 2.

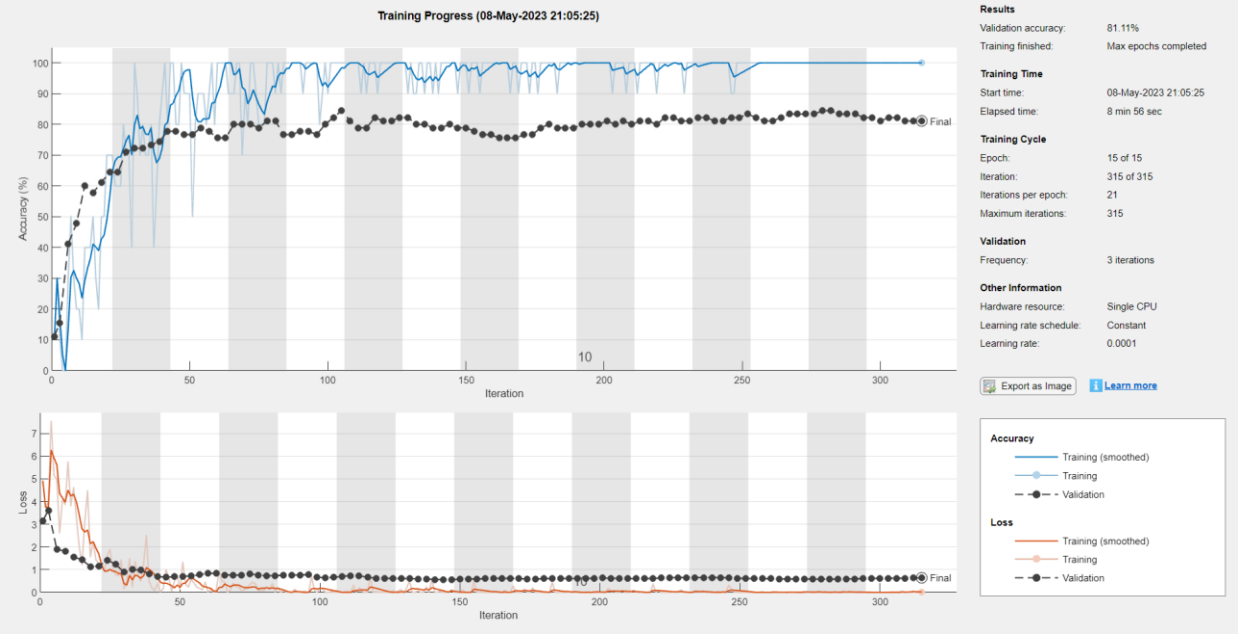


Figura 4-13. Entrenamiento AlexNet, valores priorizando precisión

Por otro lado, se ha realizado el entrenamiento del modelo usando una configuración basada en priorizar la velocidad, Figura 4-14. Para ello, se modifican los parámetros correspondientes a *MaxEpochs*, pasando de 8 a 4, *NumValidationImages*, de 8 a 6, y *NumTrainImages*, que pasa de 49 a 32.

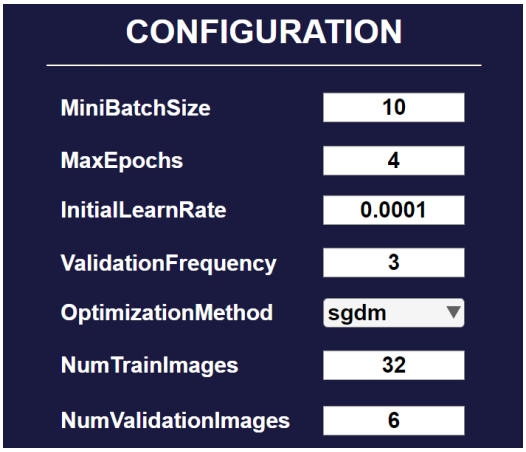


Figura 4-14. Valores priorizando velocidad en AlexNet

Como se puede ver en el analizador del modelo de la Figura 4-15, la función de precisión empieza creciendo también de forma rápida, aunque no consigue alcanzar los mismos valores que con las parametrizaciones anteriores. Además, no alcanza la misma estabilidad a lo largo del proceso, teniendo altibajos en los *epochs 2, 3* y *4*.

Aun así, ha alcanzado una precisión de un 76.78%, un 2.7% menos, lo cual, no es un mal resultado teniendo en cuenta que ha tardado un 50% menos de tiempo en el entrenamiento. Para la función de pérdida, ocurre algo similar, no logra unos valores tan estables y ahora se encuentra por encima del uno, algo peor que en casos anteriores.

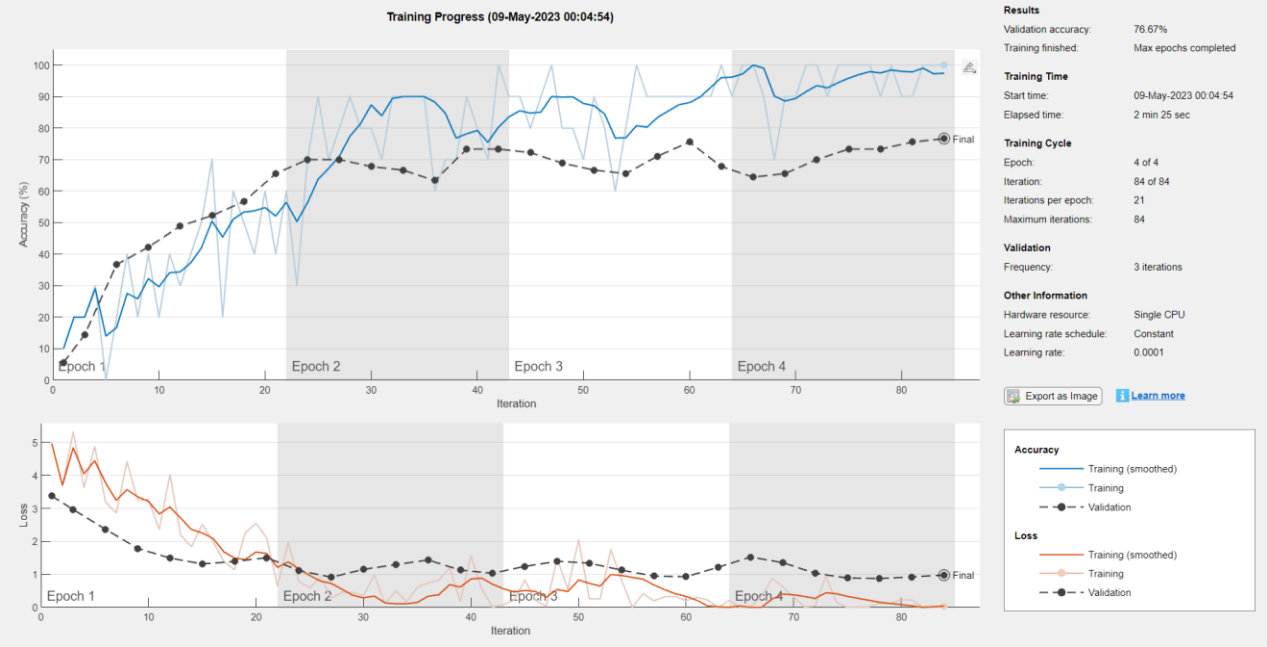


Figura 4-15. Entrenamiento AlexNet, valores priorizando velocidad

4.3.2 Modelo GoogLeNet

Para el modelo GoogLeNet, también se dispone del analizador del modelo, Figura 4-16, en este caso se observa un número menor de pesos a aprender por el modelo (*total learnables*), 6.9M millones y un número de capas totales (144) mucho mayor y compleja, y al igual que para el modelo AlexNet, es necesaria la inexistencia de avisos (*warnings*) o errores (*errors*) para poder comenzar con el proceso de una forma segura.

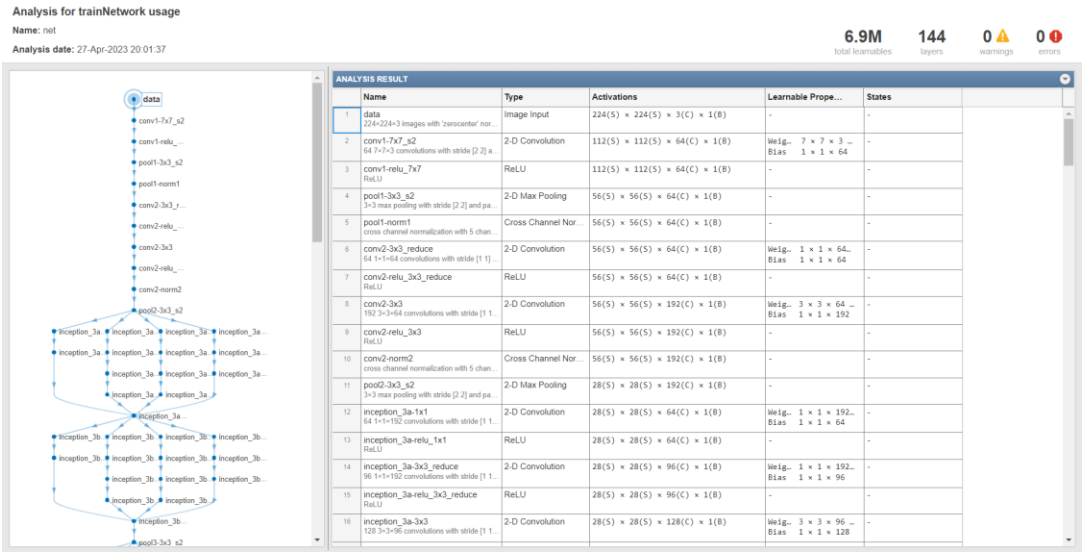


Figura 4-16. Analizador del modelo de red GoogLeNet

En un primer entrenamiento usando los parámetros por defecto definidos en la aplicación, Figura 4-17, la función de precisión crece de forma rápida durante el *epoch 1* hasta alcanzar un 65% aproximadamente, para seguir haciéndolo de forma muy uniforme, formando una curva suave y con muy pocas variaciones. Logrando un 83.33% de precisión al final de la ejecución.

Para la función de pérdida, ocurre algo similar, se observa que en el *epoch 1* decrece hasta el valor uno, suavizándose y estabilizándose a partir de ahí y hasta el final en valores próximos al 0.5, el cual es un muy buen resultado.

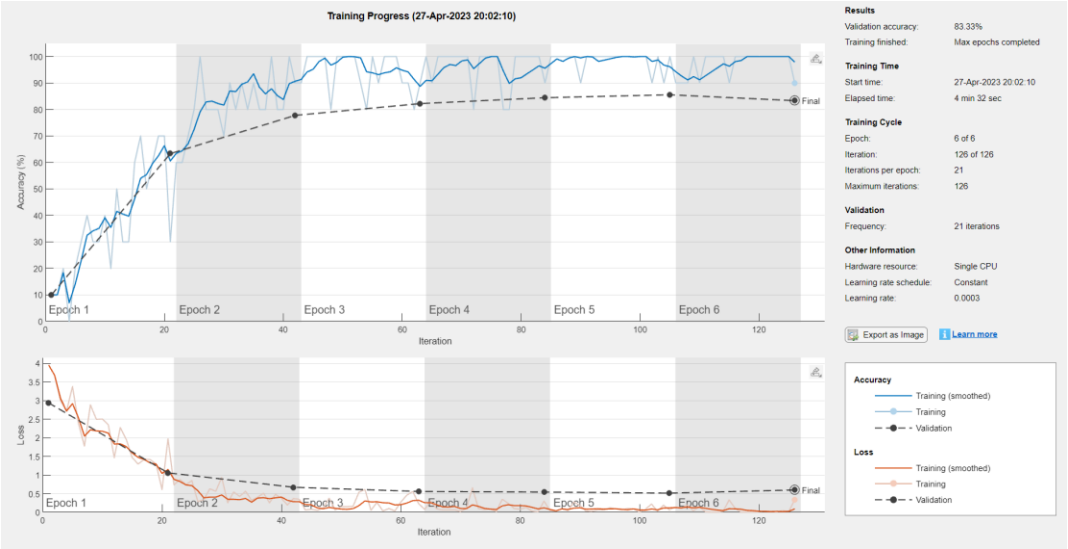


Figura 4-17. Entrenamiento GoogLeNet, valores por defecto

En la Figura 4-18 se muestra la matriz de confusión relativa a la de validación, en la que se muestran la precisión del resultado de la clasificación de distintos los platos de comida.

En la diagonal principal, las casillas de color azul indican las clasificaciones correctas realizadas para cada categoría. Por ejemplo, para la primera categoría "churros", el valor es 4.

Si se analiza el resultado por filas, en el caso anterior "churros", de imágenes de este plato, 4 se han clasificado como tal, mientras que una, lo ha hecho como "helado". Las columnas de la tabla de la derecha con porcentajes, en este mismo caso, 80% y 20%, indican la precisión de la clasificación. Lo óptimo es obtener un 100% en todas, como en el caso de la categoría "espaguetis_bolonesa", en la que, de 8 imágenes, todas han dado como resultado su propia categoría.

En la parte inferior, aparece otra tabla con porcentajes, destinada a la comparación entre las categorías predichas y las verdaderas. Al igual que en la tabla anterior, lo mejor es alcanzar el 100% en cada categoría, como en el caso de los "huevos_revueltos" que, de 9 imágenes predichas como tal categoría, todas han sido clasificadas de forma satisfactoria. En el caso peor, está la categoría "churros" que, de un total de 9 predicciones, sólo se han clasificado correctamente 4.

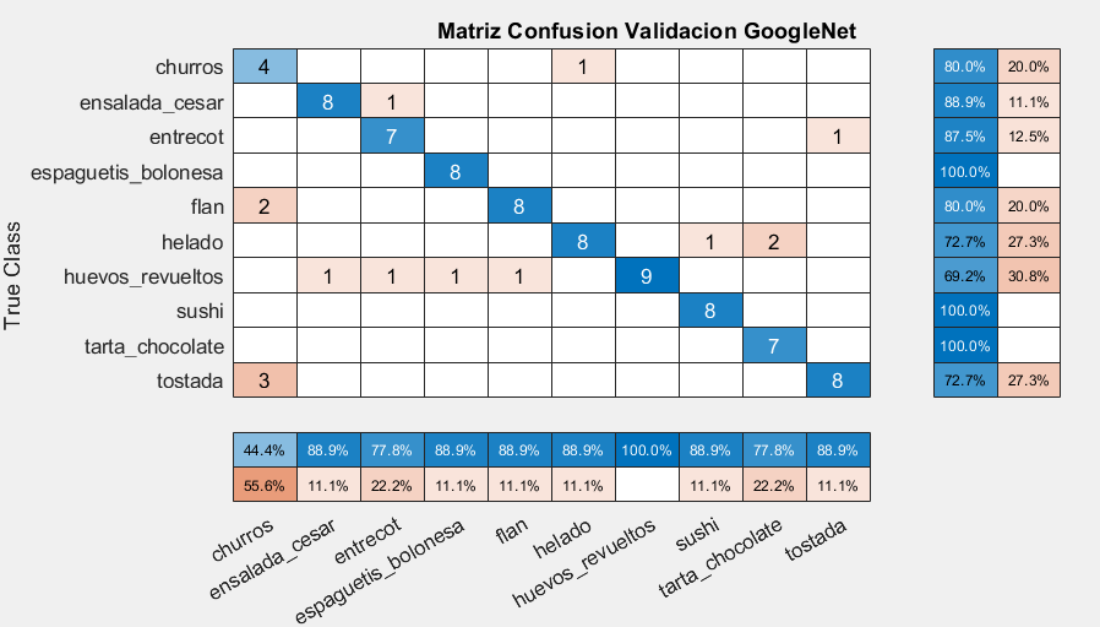


Figura 4-18. Matriz de confusión de GoogLeNet, valores por defecto

Para comparar resultados de precisión con otras configuraciones al igual que para el modelo AlexNet, se procede a modificar los parámetros para realizar un entrenamiento que priorice obtener una mayor precisión, Figura 4-19. Para ello, el valor de *MaxEpochs*, pasa del valor 6 al 15, *NumTrainImages*, de 16 a 32 y *NumValidationImages*, de 8 a 16.

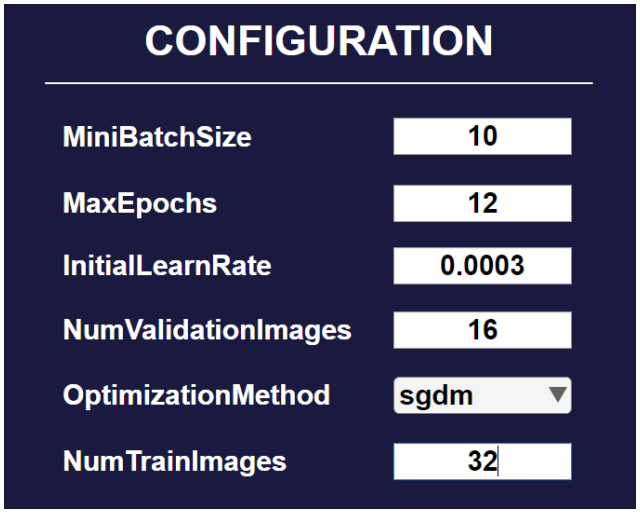


Figura 4-19. Valores priorizando precisión en GoogLeNet

En primer vistazo, se ve como en el *epoch 1*, Figura 4-20, la función de precisión del entrenamiento ha alcanzo un valor cercano al 80%, para luego estabilizarse y, con muy pocos altibajos, terminar con un 88.89%. Este un muy buen resultado en comparación a la configuración por defecto teniendo en cuenta que ha tardado 2 minutos y 15 segundos más.

En torno a la función de pérdida, se aprecia un comportamiento similar al de la configuración anterior, bajando rápidamente durante el *epoch 1*, logrando un valor, eso sí, inferior al 1, para luego estabilizarse hasta el final, acercándose a un valor intermedio entre el 0 y el 0.5.

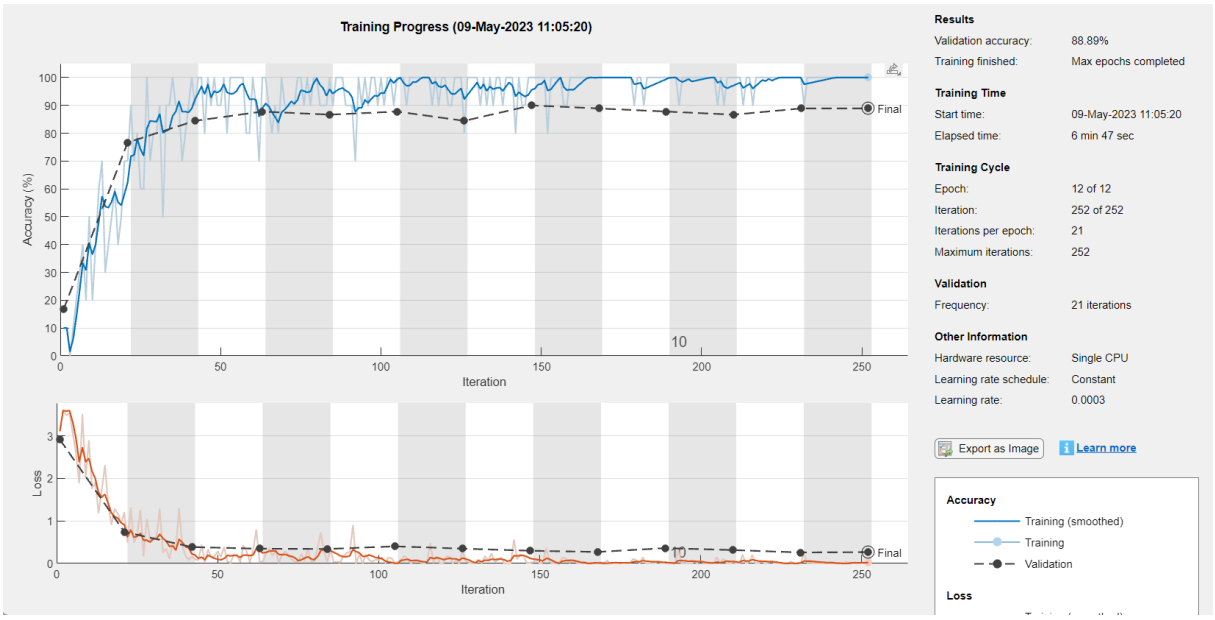


Figura 4-20. Entrenamiento GoogLeNet, valores priorizando precisión

Para obtener un entrenamiento basado en conseguir una velocidad inferior, se modifican los siguientes parámetros, *MaxEpochs*, pasando de 6 a 5, *NumValidationImages*, que cambia de 8 a 4 y *NumTrainImages*, de 16 a 10, como se observa en la Figura 4-21.

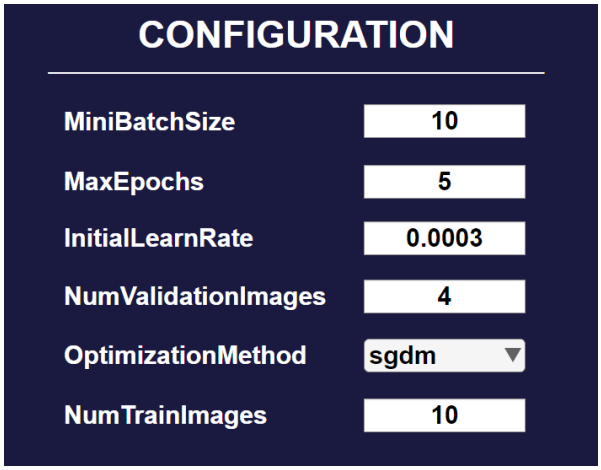


Figura 4-21. Valores priorizando velocidad en GoogLeNet

Tomando el resultado del entrenamiento, Figura 4-22, se observa cómo la función de precisión al terminar el *epoch 1* no consigue un valor tan alto como en la ocasión anterior, limitándose a alcanzar un 60%, y llegando al 80% tras el *epoch* 2. El proceso termina alcanzando un 87.73% en apenas 3 minutos y 50 segundos de ejecución, lo cual

sorprende debido a que ha conseguido mejores resultados que con la configuración por defecto en un tiempo menor.

La función de pérdida sufre un proceso similar al de precisión, hasta el final del *epoch 2* no se observa un valor inferior al 1, valor que alcanzaba el entrenamiento al finalizar el *epoch 1* con los parámetros destinados a priorizar una mayor precisión. Finalmente, termina con un valor próximo al 0.5.

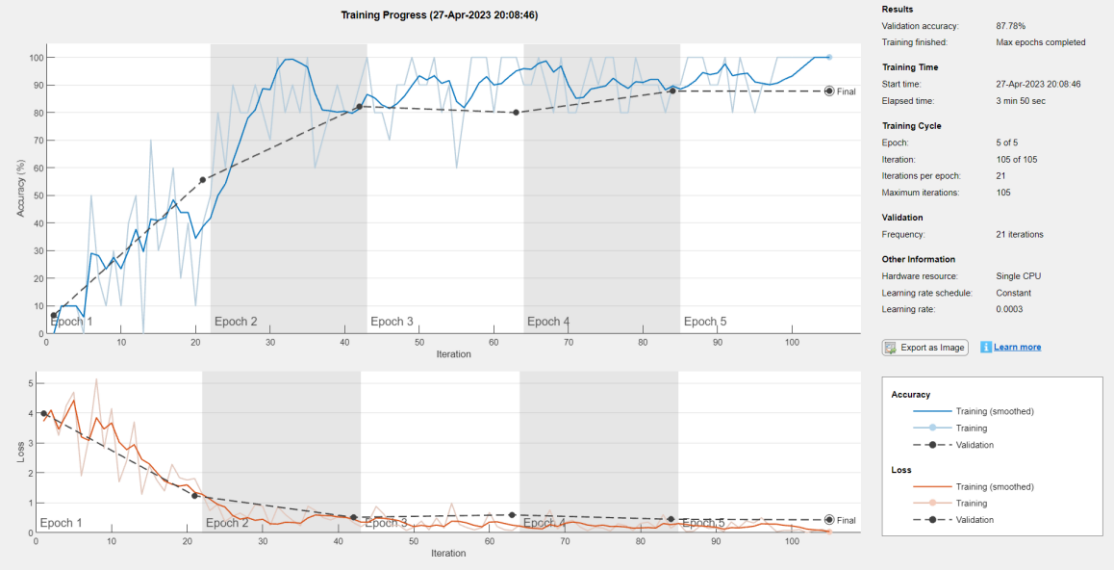


Figura 4-22. Entrenamiento GoogLeNet, valores priorizando velocidad

Entre los dos modelos de redes analizados, AlexNet y GoogLeNet, destaca este último como el más apropiado para el conjunto de imágenes disponible, logrando una mayor precisión en todas las distintas configuraciones aplicadas y empleando tiempos similares.

4.4 Clasificación

Una vez terminado el entrenamiento desde la versión de escritorio de la aplicación, los modelos de redes se almacenan en un archivo con sus pesos actualizados para las imágenes usadas en el proceso. Este archivo se utiliza para realizar la clasificación de las imágenes que reciba y así mostrar el resultado por pantalla, si se desea realizar desde la aplicación de escritorio no es necesario realizar ningún proceso extra, pero si se desea realizar desde la versión móvil, es necesario subir estos modelos a la carpeta del proyecto en MatLab Drive.

4.5 MatLab Drive y ThingSpeak

Una vez obtenido el resultado de la clasificación de un plato de comida desde la versión móvil de la aplicación gracias al uso de MatLab Drive, el proceso pasa a ThingSpeak, que rescata la información almacenada de las calorías correspondientes al plato identificado y actualiza el campo *numScans*, que guarda la cantidad de veces que ese plato en concreto ha sido identificado. Una vez superado este proceso, el disparador configurado dentro de la plataforma se activa, desencadenando en la publicación de un tweet en la cuenta previamente configurada, en el que se indica el nombre del plato detectado, sus calorías y la fecha en la que se ha realizado la clasificación, tal y como podemos observar en la Figura 4-23.

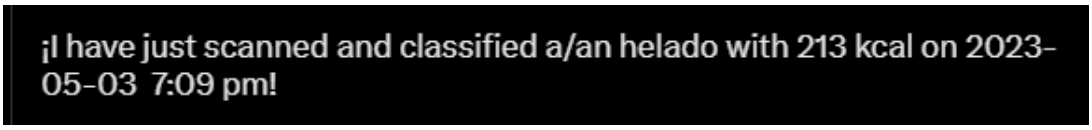


Figura 4-23. Ejemplo de tweet generado por ThingSpeak

Capítulo 5 - Conclusiones y trabajo futuro

5.1 Conclusiones

En el presente trabajo se ha desarrollado una aplicación para la identificación de platos de comida. Para lograrlo se ha usado el programa MatLab, que lo ha hecho posible gracias a su gran cantidad de módulos destinados al aprendizaje profundo y a las redes neuronales convolucionales.

Para el desarrollo del trabajo, ha sido necesario un banco de imágenes clasificadas por el nombre del plato y redimensionadas, para poder llevar a cabo el entrenamiento de las redes de AlexNet y GoogLeNet. A continuación, y gracias a las posibilidades de este tipo de redes, se llevó a cabo su reentrenamiento con los parámetros deseados y las imágenes seleccionadas.

El objetivo principal de este trabajo puede dividirse en dos pilares. El primero y más importante, consiste en el desarrollo de una aplicación de escritorio, constituida por una interfaz gráfica completa, pero simple, para que pueda ser usada sin problema por cualquier persona y la cual, se encarga del realizar todo el proceso de parametrización y entrenamiento de las redes, así como de la clasificación de alimentos. Y el segundo, en el desarrollo de una versión móvil reducida usando MatLab Mobile y MatLab Drive, pudiendo obtener la clasificación del plato usando una fotografía tomada con la cámara de un dispositivo móvil habilitado a tal efecto y sin tener que pasar a la versión de escritorio, usando eso sí comandos en lugar de una interfaz gráfica.

Como parte optativa de ampliación, la versión móvil se conecta y envía la información del plato reconocido a la plataforma ThingSpeak, perteneciente al ámbito del Internet de las Cosas, para publicar un tweet con la información calórica del plato, la cual se encuentra previamente almacenada en la propia plataforma.

Tal y como se mencionaba anteriormente, las imágenes con las que se reentrenan las redes han sido seleccionadas para el conseguir el propósito de esta aplicación, y cualquier usuario puede mejorar su eficacia o ampliar sus posibilidades, añadiendo más imágenes.

Por último, puede concluirse que se han conseguido todos los objetivos establecidos al principio para la realización de esta aplicación, la cual permite realizar su función de forma adecuada.

5.2 Trabajo futuro

A continuación, se muestran algunas ideas y mejoras en relación con el trabajo realizado para poder llevarse a cabo en futuras versiones:

- Investigación y realización de una interfaz de usuario más sencilla y amigable para la versión móvil, que permita integrar toda la parte de MatLab Drive y de ThingSpeak.
- Incorporación o implementación del resultado del proceso de clasificación a aplicaciones destinadas a la accesibilidad de las personas.
- Ampliación de la variedad de modelos de redes a utilizar dentro de la aplicación como, por ejemplo, ResNet o VGG16.
- Utilización de los datos recogidos por ThingSpeak, como fecha, hora o número de veces, en cada clasificación para realizar estudios acerca del consumo de los platos.
- Posibilitar la opción de realizar publicaciones en otras redes sociales, como Instagram, compartiendo la imagen realizada junto con el resultado de cada clasificación.
- Aumento del número de platos de comida que las redes pueden clasificar.
- Conexión a una plataforma con información sobre restaurantes para localizar lugares donde poder degustar el plato clasificado.
- Conexión a la API de Google Maps o cualquier otra similar para, usando la localización de nuestro dispositivo móvil, detectar el lugar en el que hemos tomado la fotografía y poder dejar una reseña directamente.
- Utilizar la inteligencia artificial para detectar el tamaño del plato de comida de la fotografía y poder precisar más las calorías del mismo, para

poder detectar si la comida está en buen estado o incluso para poder mostrar la receta junto los ingredientes o los pasos a seguir.

- Usar los datos recogidos por la plataforma ThingSpeak para hacer recomendaciones al usuario sobre platos similares para hacer recomendaciones.

Introduction

Motivation

Nutrition has been and already is one of the most important aspects in our lives since it has a direct impact in our health. For years, multiple tools have been developed and used in this ambit to help people keep track and to be able to achieve a healthier diet. The main problem they had or has is the requirement to manually enter food information, which sometimes is a tedious and error-prone process.

Thanks to the advent of technology, and more specifically, mobile devices and their applications, the use of this tools has become popular, thanks to their ability to record intake from anywhere and the ease of making use of the mobiles device's camera, take a picture of the food and, through the use of intelligent methodologies, determine its nutritional content, as well as its caloric information. In addition, with the constant advancement of technology, these applications have become more sophisticated and precise, making it possible not only to identify individual foods, but also complete plates of food, taking into account their components.

The work carried out by Shifat et al. (2022) about the realization of a system to recognize foods that, using machine learning and computer vision principles, have managed to obtain precision results of 98.05%. On the other hand, we found the system created by Liu et al. (2019), centered at improving the accuracy of the small datasets used to perform the training, using image augmentation techniques to create a more balanced and comprehensive image bank.

Although these two previous works are not directly focused on the calculation of calories, their focus is relevant to improve the accuracy and speed of response for this type of applications.

Despite the fact that there are still tasks ahead to perfect and extend these tools, works such as Aktı et al. (2022) demonstrate the interest in achieving the expansion of these, in this case, focusing on the development of a food recognition application focused on Middle Eastern cuisine, which has remained practically unexplored, using the Mobilenet-model v2 and data augmentation methods for underrepresented classes,

achieving 94% accuracy in 23 food classes. Mention should also be made of the studies carried out by Lo et al. (2020) for the implementation of deep learning techniques to estimate the volume or variability of food portions on a plate of food through a simple image.

The work of Barylak Alcaraz (2021), within the context of the Internet of Things is in the same vein as this application. Food plates are also identified using a mobile device, making the procedures corresponding to the calculation, storage and sending of messages through the ThingSpeak (2023) platform. This work serves as inspiration for the development of the application that is presented below.

Goals

The main objective of this project is the development of a desktop application to identify and classify images of food dishes using deep learning methods. Its purpose is to be useful for all those people who want to know the calories in a plate of food or who are interested in keeping track of their calorie intake, in order to improve their health.

In addition, a mobile version will be developed that will have constant access to Internet connection and that will be connected to an IoT platform, which will store and process the data generated by the application in the classification to, finally, publish it on the Twitter social network.

In this way, the following specific objectives are generated:

- Application and adaptation of deep learning methods, in this case, Convolutional Neural Networks (RNC), more specifically AlexNet and GoogLeNet.
- Retraining of the different network models using a previously selected image dataset.
- Development of a graphical user interface for the application that is friendly, easy to use and that allows the entire training and image classification process to be carried out.
- Connection with external IoT platforms, in this case ThingSpeak, to store data in the cloud and perform different operations with it.

- Publication of the data resulting from the classification on social networks, in this case Twitter, to inform and keep track of the results.

Work plan

To carry out this work, a series of tasks have been followed, shown below in the order of their execution:

- **Identification and selection of the development language**
  For the development of the application, two main options were proposed, MatLab or Python. Both options allow you to implement and work with convolutional neural networks thanks to their own modules, Deep Learning Toolbox in the case of MatLab and TensorFlow in the case of Python.
  Finally, it was decided to select MatLab due to its integration with MatLab App Designer for graphical user interface developments and for its ease of connection with the future ThingSpeak platform.
- **Identification and selection of convolutional neural networks**
  In order to achieve the purpose of this application, classify images of food dishes, convolutional neural networks would be used. There are a wide variety of models, so the most appropriate ones had to be selected. It was decided to use the AlexNet and GoogLeNet models, which are already pre-trained models that allow to be retrained to classify the images that we want.
- **Image dataset search**
  In order to retrain the selected network models, it was necessary to have a large enough image bank.
  After several searches, the iFood – 2019 at FGVC6 dataset was chosen, which includes images of food dishes classified by name.
- **Development of the different modules of the application**
  - **Image resizing**: The selected network models need to receive images with specific dimensions, 227 x 227 x 3 in the case of AlexNet and 224 x 224 x 3 in the case of GoogLeNet.

- **Selection of the different possible network models**: When having several models of convolutional neural networks, it is necessary to allow the option of selecting one or the other when carrying out the training and when carrying out the classification.
- **Training and parameterization of convolutional neural networks**: In order to classify the images, it is necessary to previously train both network models using the image dataset. These allow to be configured through a series of parameters, so it would be necessary to offer a way to be able to modify them to lead the process with greater precision, greater speed or an intermediate point.
- **Classification of images of food dishes**: Making use of the training files generated in the training process, the application has to allow selecting an image stored on our computer, classifying it and displaying the result obtained on the screen.

- **Design and development of the graphical user interface of the application**
  In order to achieve an intuitive and simple use of the application, a friendly graphical user interface was developed. Mockups were made for each of the views that the application would have to, later, using the MatLab App Designer tool, implement it in such a way that it would allow all the functions described in the previous point to be carried out.
- **Development of the mobile version of the application**
  To facilitate the use of the application and to be able to use it from anywhere without the need of a computer, it was decided to develop a simple mobile version using MatLab Drive and MatLab Mobile application. In this way, we could take a picture with our mobile device's camera and get the classification result directly.
- **Connection to the ThingSpeak platform**
  To extend the functionality of the application, it was decided to make use of the ThingSpeak platform, which allows connecting with our application in MatLab and offering various utilities accordingly. In this case, it would be used to store the caloric information of each dish of food that we identify, in such a way that, every time a dish is scanned, it connects with a Twitter

account to publish a tweet with the information resulting from the classification.

Memory organization

The memory is organized so that in chapter two, the methods and techniques applied to implement the required functionality are described. In chapter three the design of the application is described. In chapter four the results obtained are shown and commented. Finally, in chapter five the conclusions obtained are summarized and the pertinent considerations on future advances are made.

Conclusions and future work

Conclusions

In the present project, an application has been developed for the identification of food dishes. To archive this goal, I used MatLab software, which has made it possible thanks to its modules for deep learning and convolutional neural networks.

For the development of the work, a bank of resized images classified by the name of the food dish was necessary to train AlexNet and GoogLeNet networks. Then, thanks to the possibilities of this type of networks, they were retrained with the desired parameters and the selected images.

The main objective of this project can be divided into two pillars. The first one and most important, in the development of a desktop application, formed by a complete graphical user interface, but keeping it simple, so that it can be used without any problem by anyone, and which handles with the entire process of parametrization and training of the networks, as well as with the food classification. And the second one, in the development of a basic command mobile version using MatLab Mobile and MatLab Drive, being able to obtain the classification of the food dish using a photograph taken directly with the camera of our smartphone and without having to use the desktop version.

As an optional part of the development, the mobile version connects and sends the information of the recognized food dish to ThingSpeak platform, belonging to Internet of Things, to publish a tweet with the caloric information of the food dish, which is previously stored in the platform.

As previously mentioned, the images with which both networks have been retrained have been selected to achieve the purpose of this application and any user can improve its efficiency or expand its possibilities by adding more images.

Finally, it can be concluded that all the objectives established at the beginning for the realization of this project, which allows it to perform its functions correctly, have been archived.

Future Work

Down below are some ideas and improvements related to the work done that can be done in future versions:

- Research and creation of a simpler and more friendly user interface for the mobile version, which allows the integration of MatLab Drive and ThingSpeak.
- Incorporation or implementation of the result of the classification process in applications destined for people accessibility.
- Expansion of the variety of networks models to be used within the application, such as ResNet or VGG16.
- Use of the data collected by ThingSpeak, such as date, time or number of times in each classification, to carry out studies on the consumption of food dishes.
- Enable the option of making posts on other social networks, like Instagram, sharing the photography made along with the result of the classification.
- Increase number of food dishes that networks can classify.
- Connection to a platform with information about restaurants to locate near places where you can eat the classified food dish.
- Connection to the Google Maps API, or any other similar, to detect the place where the photo has been taken using the location of our smartphone and be able to leave a review directly.
- Use artificial intelligence to detect the size of the food dish in the photograph and be able to specify its calories more precisely, be able to detect if the food is in good condition or be able to show the recipe together with the ingredients and the steps to follow.
- Use the data collected by ThingSpeak to make recommendations to the user about similar dishes.

BIBLIOGRAFÍA

1. Barylak Alcaraz, M.V. (2021). Implementación de un Ecosistema IoT para el control de la ingesta de calorías mediante mecanismos de Aprendizaje Profundo. Trabajo Fin de Máster. Universidad Complutense de Madrid. Disponible online: https://eprints.ucm.es/id/eprint/67235/ [Último acceso: enero 2023].
2. Bishop, C. M. & Nasrabadi, N. M. (2006): Pattern Recognition and Machine Learning, Springer, NY, USA.
3. Aktı, Ş., Qaraqe M., Ekenel, H. K. (2022): A Mobile Food Recognition System for Dietary Assessment. Computing Research Repository. arXiv:2204.09432 [cs.CV].
4. BVLC AlexNet Model (2021): Disponible online: https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet [Último acceso: enero 2023].
5. BVLC GoogleNet Model (2021): Disponible online: https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet [Último acceso: enero 2023).
6. Cognodata (n.d.): 12 principios de la metodología agile en el desarrollo de proyectos. Disponible online: https://www.cognodata.com/principios-metodologia-agile-desarrollo-proyectos/ [Último acceso: febrero 2023].
7. Elias Linderman (2020): *Appdesigner MATLAB EN ESPAÑOL Aplicacion de 2 Ventanas*. [Vídeo online]. Disponible online: https://www.youtube.com/watch?v=8JDwzEcCnT0&t=698s [Último acceso: marzo 2023].
8. iFood – 2019 at FGVC6 (2019): Disponible online: https://www.kaggle.com/c/ifood-2019-fgvc6 [Último acceso: diciembre 2022].
9. ImageNet LSVRC (2012): Large Scale Visual Recognition Challenge 2012 (ILSVRC2012). Disponible online: http://www.image-net.org/challenges/LSVRC/2012/ [Último acceso: enero 2023).

10. Imagenet (2020): Disponible online: http://www.image-net.org [Último acceso: enero 2023].
11. Kaggle. (2021): https://www.kaggle.com/ [Último acceso: diciembre 2022].
12. Kanbanize (n.d.): Gestión Ágil de Proyectos: una Guía Completa. Disponible online: https://kanbanize.com/es/agiles/metodologia-agile [Último acceso: febrero 2023].
13. 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.
14. Lo, F. P. W., Sun, Y., Qiu, J., & Lo, B. (2020). Image-Based Food Classification and Volume Estimation for Dietary Assessment: A Review. IEEE Journal of Biomedical and Health Informatics, 24(7), 1902-1917. doi: 10.1109/JBHI.2020.2987943.
15. Mathworks (2023): Desarrollar apps mediante App Designer. Disponible online: https://es.mathworks.com/help/matlab/app-designer.html [Último acceso: febrero 2023].
16. Mathworks (2023): Deep learning con imágenes. Disponible online: https://es.mathworks.com/help/deeplearning/deep-learning-with-images.html [Último acceso: enero 2023].
17. Mathworks (2023): ¿Qué son las redes neuronales convolucionales? Disponible online: https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html [Último acceso: noviembre 2022].
18. Mathworks (2023): Progress bar in app design (GUI). Disponible online: https://es.mathworks.com/matlabcentral/answers/750314-progress-bar-in-app-design-gui?s_tid=srchtitle [Último acceso: marzo 2023].
19. Matlab Drive (2023): Disponible online: https://es.mathworks.com/products/matlab-drive.html [Último acceso: marzo 2023].

20. Matlab Mobile (2023): Disponible online: https://play.google.com/store/apps/details?id=com.mathworks.matlabmobile&hl=en_US&pli=1 [Último acceso: febrero 2023].

21. Pajares Martinsanz, G., Herrera Caro, P. J. & Besada Portas, E. (2021): Aprendizaje Profundo. RC-Libros.

22. Pan, L., Qin, J., Chen, H., Xiang, X., Li, C., & Chen, R. (2019): Image augmentation-based food recognition with convolutional neural networks. Comput. Mater. Continua, 59(1), 297-313.

23. Programmer World (2019): *How to design Radio button, check boxes and drop down menu in MATLAB App Designer?* [Vídeo online]. Disponible online: https://www.youtube.com/watch?v=w9U0JKK6Pik [Último acceso: marzo 2023].

24. Ruder, S. (2017): An overview of gradient descent optimization algorithms. arXiv:1609.04747v2 [cs.LG].

25. Russakovsky, O., Deng, J., Su, H., Krause, J. (2015): Imagenet large scale visual recognition challenge. International Journal of Computer Vision 115.3: 211-252.

26. Shifat, S. M., Parthib, T., Pyaasa, S. T., Chaity, N. M., Kumar, N., Morol, M. K. (2022): A Real-time Junk Food Recognition System based on Machine Learning. Computing Research Repository. arXiv:2203.11836 [cs.CV].

27. 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].

28. ThingSpeak (2023): ThingSpeak for IoT Projects. Disponible online: https://thingspeak.com/ [Último acceso: abril 2023].

APÉNDICES

Apéndice A - Código y manual de usuario

Para ejecutar la aplicación de escritorio y/o ver el código fuente es necesario seguir una serie de pasos:

1. Descargar e instalar MatLab R2022b para escritorio junto a los siguientes *Add-Ons*: Deep Learning Toolbox, Deep Learning Toolbox Model for AlexNet Network, Deep Learning Toolbox Model for GoogLeNet Network e Image Processing Toolbox.
2. Ejecutar MatLab cómo administrador y seleccionar el directorio "Trabajo-Fin-de-Grado-main/Material Aplicación Escritorio/".
3. Abrir TGF.mlapp desde el panel lateral de archivos de MatLab y pulsar al botón *Run* desde la nueva ventana abierta de MatLab App Designer.
4. Si se desea visualizar el código fuente, hay que seleccionar la opción *View Code* dentro de la ventana mencionada en el punto anterior para cada archivo de vista *mlapp*.

Si la aplicación ha sido descargada por primera vez, es imprescindible realizar un primer entrenamiento de las dos redes neuronales convoluciones para que puedan generarse los archivos de entrenamiento y poder realizar las clasificaciones posteriormente.

Para ejecutar la aplicación desde un dispositivo móvil, es necesario seguir los siguientes pasos:

1. Descargar la aplicación *MatLab Mobile*.
2. Subir los archivos existentes dentro de la carpeta "Trabajo-Fin-de-Grado-main/Material Aplicación Móvil/" a MatLab Drive.
3. Subir los archivos de entrenamiento generados netTransferAlimentosAlexNet.mat y netTransferAlimentosGoogleNet.mat a MatLab Drive.

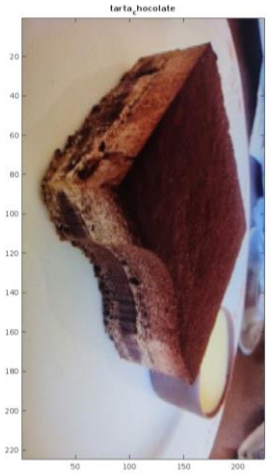
4. Dentro de la aplicación móvil de MatLab, ejecutar los siguientes comandos en orden: "*IniciarCamaraChristian*" y "CamaraClasificacionChristian".

La publicación de tweets a la hora de realizar una clasificación con la versión móvil puede consultarse desde el siguiente perfil de Twitter: https://twitter.com/TFGChristian.

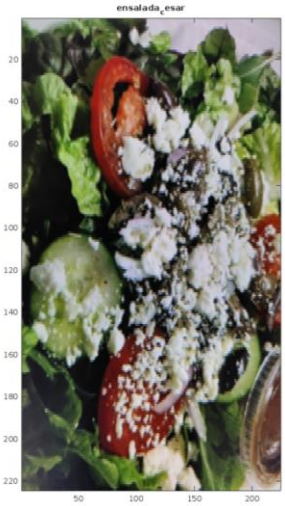
Apéndice B - Resultados de ejemplo de la clasificación de alimentos

Las siguientes capturas de pantalla muestran el resultado obtenido tras la clasificación de platos de comida por parte de ambos modelos de red.

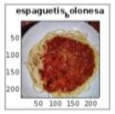
```
etiquetaAlex =
  categorical
     tarta_chocolate

etiquetaGoogle =
  categorical
     tarta_chocolate
```

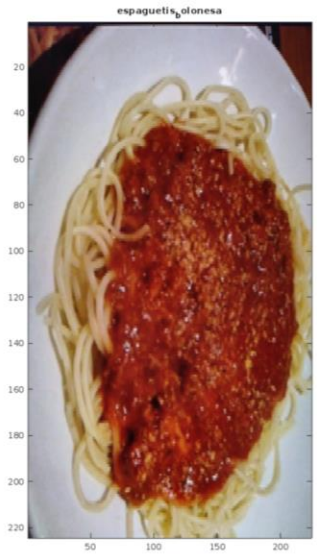



```
etiquetaAlex =
  categorical
     ensalada_cesar

etiquetaGoogle =
  categorical
     ensalada_cesar
```

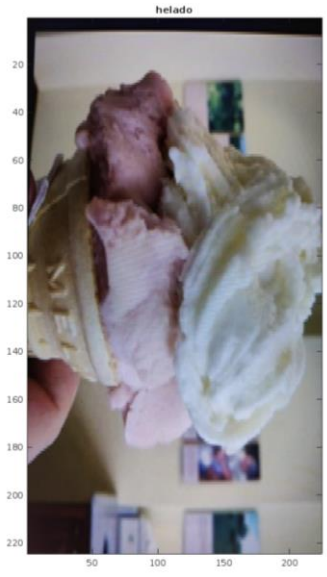


```
etiquetaAlex =
  categorical
      espaguetis_bolonesa

etiquetaGoogle =
  categorical
      espaguetis_bolonesa
```






```
etiquetaAlex =
  categorical
      helado

etiquetaGoogle =
  categorical
      helado
```



Apéndice C - Licencias de los iconos

La siguiente lista de iconos ha sido usada tanto en la interfaz gráfica como en la realización de diagramas:

- archivo-de-base-de-datos.png - Icono realizado por Uniconlabs disponible en https://www.flaticon.es/icono-gratis/archivo-de-base-de-datos_4233336

- balanced-diet.png - Icono realizado por paulalee disponible en https://www.flaticon.es/icono-gratis/dieta-equilibrada_6774898

- base-de-datos.png - Icono realizado por Freepik disponible en https://www.flaticon.es/icono-gratis/base-de-datos_5939181?related_id=5939259&origin=search

- brain-training.png - Icono realizado por piksart disponible en https://www.flaticon.com/free-icon/brain-training_7200519?term=brain+training&page=1&position=35&origin=tag&related_id=7200519

- classification.png - Icono realizado por Eucalyp disponible en https://www.flaticon.es/icono-gratis/clasificacion_3676513?term=clasificar&related_id=3676513

- cloud.png - Icono realizado por Freepik disponible en https://www.flaticon.es/icono-gratis/nube_565190

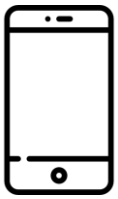
- computadora.png - Icono realizado por Freepik disponible en https://www.flaticon.es/icono-gratis/computadora_73443

- image-gallery.png - Icono realizado por Pixel perfect disponible en https://www.flaticon.es/icono-gratis/imagen_1829415

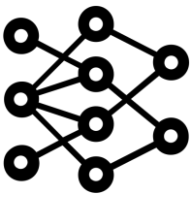
- letter-a.png - Icono realizado High Quality Icons disponible en https://www.flaticon.es/icono-gratis/carta-a_3665909

- letter-g.png - Icono realizado por High Quality Icons disponible en https://www.flaticon.es/icono-gratis/letra-g_3665939?term=letra+g&related_id=3665939

- mobile-phone.png – Icono realizado por Freepik disponible en https://www.flaticon.es/icono-gratis/telefono-movil_545245



- neural-network.png - Icono realizado por Vectors Tank disponible en https://www.flaticon.es/icono-gratis/red-neuronal_6461819



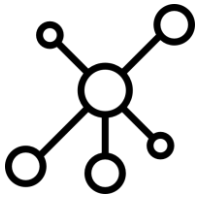
- photo.png - Icono realizado por Good Ware disponible en https://www.flaticon.es/icono-gratis/foto_685669

- picture.png - Icono realizado por Freepik disponible en https://www.flaticon.es/icono-gratis/imagen_1160358

- play-button.png - Icono realizado por Those Icon disponible en https://www.flaticon.es/icono-gratis/boton-de-play_2088565

- question-sign-png - Icono realizado por Dave Gandy disponible en https://www.flaticon.es/icono-gratis/signo-de-pregunta_25333

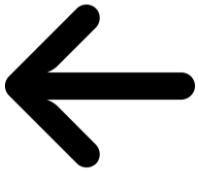
- red.png - Icono realizado por DinosoftLabs disponible en https://www.flaticon.es/icono-gratis/conexion_5366616



- reset.png - Icono realizado por Freepik disponible en https://www.flaticon.es/icono-gratis/reiniciar_5277847

- restaurant.png - Icono realizado por Nikita Golubev disponible en https://www.flaticon.es/icono-gratis/restaurante_1919608

- return.png - Icono realizado por Kirill Kazachek disponible en https://www.flaticon.es/icono-gratis/flecha_507257



- thingspeak-logo.png - Icono disponible en https://thingspeak.com/

- twitter.png - Icono realizado por Pixel perfecto disponible en https://www.flaticon.es/icono-gratis/gorjeo_2111819