IMPLEMENTACIÓN DE UN ECOSISTEMA IOT PARA EL CONTROL DE LA INGESTA DE CALORÍAS MEDIANTE MECANISMOS DE APRENDIZAJE PROFUNDO

IMPLEMENTATION OF AN IOT ECOSYSTEM FOR CONTROLLING CALORIE INTAKE THROUGH DEEP LEARNING MECHANISMS

TRABAJO FIN DE MÁSTER
CURSO 2020-2021

AUTOR
MARÍA VICTORIA BARYLAK ALCARAZ

DIRECTOR
GONZALO PAJARES MARTINSANZ

CONVOCATORIA: JULIO 2021
CALIFICACIÓN: 9 (SOBRESALIENTE)

MÁSTER EN INTERNET DE LAS COSAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

IMPLEMENTACIÓN DE UN ECOSISTEMA IOT PARA EL CONTROL DE LA INGESTA DE CALORÍAS MEDIANTE MECANISMOS DE APRENDIZAJE PROFUNDO

IMPLEMENTATION OF AN IOT ECOSYSTEM FOR CONTROLLING CALORIE INTAKE THROUGH DEEP LEARNING MECHANISMS

TRABAJO DE FIN DE MÁSTER EN INTERNET DE LAS COSAS
DEPARTAMENTO DE INGENIERÍA DE SOFTWARE E INTELIGENCIA ARTIFICIAL

CURSO ACADÉMICO 2020-2021

AUTOR
MARÍA VICTORIA BARYLAK ALCARAZ

DIRECTOR
GONZALO PAJARES MARTINSANZ

CONVOCATORIA: JULIO 2021
CALIFICACIÓN: 9 (SOBRESALIENTE)

MÁSTER EN INTERNET DE LAS COSAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

25 DE JUNIO DE 2021

DEDICATORIA

A mis padres, por el cariño y apoyo incondicional que me dan en todos los proyectos en los que me embarco.

Y a Jorge, por haber estado ahí en todo momento.

AGRADECIMIENTOS

Me gustaría agradecerle a mi tutor Gonzalo Pajares todo el apoyo ofrecido durante el curso, y su guía y ayuda durante la realización de este trabajo.

RESUMEN

El presente trabajo propone una solución inteligente dentro del paradigma IoT (*Internet of Things*). Se trata de un diseño que clasifica imágenes de alimentos capturadas con la cámara de un dispositivo móvil según el tipo de alimento a la vez que se obtienen las calorías asignadas al mismo con el fin de llevar un control sobre la ingesta de calorías en el tiempo.

La solución inteligente se centra en la utilización de dos modelos de redes neuronales convolucionales, concretamente AlexNet y GoogLeNet, que se adaptan y se rediseñan de acuerdo con las especificaciones de la aplicación, más concretamente para la clasificación de diez categorías de alimentos distintas, a partir de los cuales se puede determinar su nivel de calorías. Son modelos pre-entrenados, que se re-entrenan para las imágenes propias en lo que se conoce como transferencia de aprendizaje.

Los resultados de la clasificación se envían a ThingSpeak, que es una plataforma remota para almacenamiento de datos y procesamiento en la nube específicamente diseñada para aplicaciones IoT. De esta forma es posible monitorizar la trazabilidad de los datos almacenados, principalmente la ingesta de calorías.

La aplicación, basada en distintos componentes de Matlab, consta de un módulo de captura de imágenes a través de la cámara de un dispositivo móvil, que tiene a la vez instalada una aplicación con capacidad de comunicación on-line, tanto con un computador central como con los servicios en la nube de Matlab (Drive) o la mencionada plataforma ThingSpeak, desde donde se pueden enviar alarmas o avisos vía Twitter. Los distintos módulos, convenientemente integrados, constituyen la aplicación en su conjunto, que permite determinar su validez mediante el análisis de los resultados obtenidos.

Palabras clave

Internet de las Cosas, IoT, Aprendizaje Profundo, Redes Neuronales Convolucionales, AlexNet, GoogLeNet, Inteligencia Artificial, Clasificación alimentos.

ABSTRACT

The present project proposes an intelligent solution under the Internet of Things (IoT) paradigm. It is a design which classifies food images captured with the camera of a mobile device according to the type of food while obtaining the calories assigned to it in order to keep track of calorie intake over time.

This intelligent solution focuses on the use of two convolutional neural network models, particularly AlexNet and GoogLeNet, which are adapted, redesigned according to the application specifications, those being the classification of ten different categories of food, from which the level of calories can be determined. These are pre-trained models, which are re-trained with images in what is known as transfer learning.

The results of the classification are then sent to ThingSpeak, which is a remote platform for data storage and cloud processing specifically designed for IoT applications. Thus, it is possible to monitor the traceability of the stored data, mainly the calorie intake.

The application, based on different Matlab components, consists of an image capturing module by using the camera of a mobile device, which also has installed an application with on-line communication capacity, both with a central computer, Matlba's cloud services (Drive) and the aforementioned ThingSpeak platform, from which alarms or warnings can be sent via Twitter. The different modules, conveniently integrated, constitute the application as a whole, which makes it possible to determine its validity by analyzing the results obtained.

Keywords

Internet of Things, Iot, Deep learning, Convolutional Neural Networks, AlexNet, GoogLeNet, Artificial Intelligence, Food classification

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

1.1 Preliminares

El avance tecnológico está posibilitando a diario, de forma eficiente, el incremento de aplicaciones para la monitorización de la alimentación de las personas de forma instantánea. Son variadas las aplicaciones para dispositivos móviles desarrolladas en los últimos tiempos con tal finalidad. En efecto, por ejemplo, en Time2Feat (2020) se proporcionan detalles concretos sobre seis aplicaciones que permiten calcular el consumo de calorías diarias, tal es el caso de *My Fitness Pal*, *Fat Secret*, Yazio, *Lose it!*, Macros o *Easyfit*. Estas aplicaciones permiten, en términos generales, llevar el registro temporal a diario, semanal o mensual, del consumo de alimentos en términos de calorías. Además, algunas de ellas pueden conectarse con dispositivos de control de la actividad física para establecer la interrelación entre consumo de calorías y actividad física.

En el ámbito del control de ciertas enfermedades como la diabetes la nutrición constituye un elemento esencial para la salud. En esta línea también se han desarrollado aplicaciones móviles para el control nutricional (DiabeWeb, 2021), aquí aparece también *My Fitness Pal*, además de Foodmeter, Diabetes Menú, myDiabeticAlert o Diabetes a la Carta.

Por otra parte, es amplia la literatura respecto a la utilización de métodos de aprendizaje automático, y más concretamente, basados en Redes Neuronales Convolucionales (CNN, *Convolutional Neural Networks*) en la identificación y clasificación de alimentos. Tal es el caso del trabajo propuesto por Zhou y col. (2019) sobre la base del control de la alimentación con el fin de la mejora de la salud en la población, que realiza una revisión de trabajos al respecto, destacando entre otros los que se mencionan a continuación. Heravi y col. (2018) diseñaron una red modificada a partir de AlexNet (Russakovsky y col., 2015) logrando valores de precisión alrededor del 95% en un conjunto de 1.316 imágenes y 13 categorías. Mezgec y Seljak (2017) también desarrollaron un modelo, NutriNet, basado en AlexNet para la identificación de comidas y bebidas, realizando un entrenamiento de ésta con más de 225 mil imágenes, logrando

precisiones de alrededor del 94%. Fu y col. (2017) utilizaron como modelo ResNet (He y col., 2016) alcanzando precisiones del orden del 68%. Herruzo y col. (2016) utilizaron el modelo GoogLeNet (BVLC GoogLeNet Model, 2021) con resultados de precisión diferentes dependiendo del conjunto de datos utilizado, llegando al 97% para un conjunto de datos específico como FoodCAT (Wu y col., 2015).

Teniendo como referencia las aplicaciones mencionadas, y bajo la inspiración de los modelos de tipo CNN, en el presente trabajo se propone una aplicación inteligente basada en técnicas de Aprendizaje Profundo bajo el paradigma de Internet de las Cosas (*Internet of Things*, IoT). Aunque posteriormente se describe con un mayor nivel de detalle el modelo arquitectónico del sistema, en la Figura 1.1 se proporciona el esquema general con sus componentes principales, estableciendo así el punto de partida del presente trabajo.

1) Se dispone de un procesador central convenientemente equipado donde se realiza el entrenamiento de los modelos de redes neuronales, aunque los mismos también podrían ser entrenados en la nube (Matlab Drive, 2021), de forma que, en cualquier caso, se encuentran accesibles desde un dispositivo móvil.
2) Mediante el dispositivo móvil, conectado con la nube, se captura una imagen de un alimento, que se clasifica con el modelo de red CNN seleccionado en cada momento.
3) Con el resultado de la clasificación se realiza la pertinente consulta a la aplicación ThingSpeak (2021), específica para el tratamiento de datos en el ámbito de IoT. Aquí se almacenan distintos datos relativos a cada alimento (calorías, grasas, proteínas, carbohidratos), devolviendo el resultado vía Twitter.

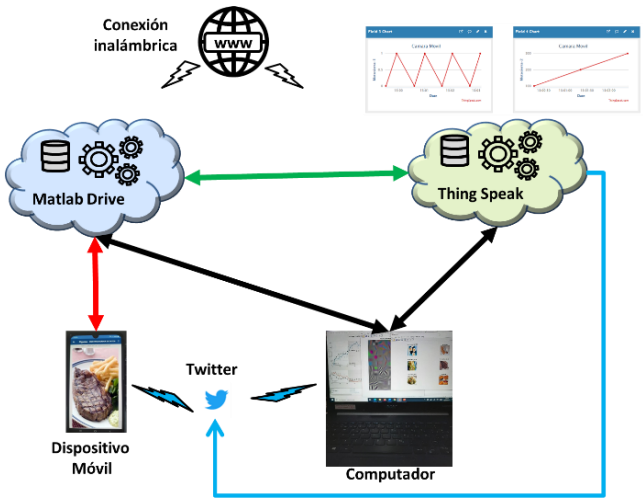


Figura 1.1. Esquema general del sistema

1.2 Objetivos

El objetivo general propuesto consiste en diseñar un sistema IoT, como una solución conceptual para la clasificación de alimentos, con capacidad de realizar un control informativo instantáneo sobre la naturaleza de cada alimento con relación a su potencial nutricional, a la vez que se puede mantener una traza de los componentes nutricionales a lo largo del tiempo. Bajo esta perspectiva, se realiza una propuesta de solución IoT.

Los objetivos específicos formulados son los siguientes,

- Diseño y adaptación de modelos CNN para su entrenamiento y toma de decisiones en clasificación.
- Configuración de la estructura de datos y procesamiento de estos en la nube, incluyendo el envío de mensajes informativos mediante Twitter.
- Integración de los módulos que implementan los procesos anteriores.
- Análisis y valoración de los resultados.

El plan de trabajo previsto tiene como base los objetivos específicos reseñados, contemplando el mismo número de fases que objetivos, y con una planificación equilibrada por semanas a lo largo del tiempo de desarrollo del proyecto.

1.3 Organización de la memoria

A partir de este punto la memoria seguirá una estructura divida por capítulos, cuyo contenido es el siguiente:

- Capítulo 2 - Redes Neuronales Convolucionales, donde se describe el funcionamiento del tipo de redes neuronales empleadas en el desarrollo del trabajo.
- Capítulo 3 - Diseño de la aplicación, donde se explica la arquitectura del sistema, junto con el proceso de entrenamiento y clasificación seguido.
- Capítulo 4 - Análisis de resultados, donde se presentan los dispositivos y programas utilizados, y los resultados obtenidos.
- Capítulo 5 - Conclusiones y trabajo a futuro, donde se exponen las conclusiones del proyecto, así como el trabajo que se podría realizar a futuro.

Capítulo 2 - Redes Neuronales Convolucionales

Una CNN es un tipo específico de red neuronal, cuyo fundamento base son las conocidas capas de convolución, que dan pie a su nombre (Pajares y col., 2021). Cualquier modelo de este tipo consta de dos fases: aprendizaje y clasificación. Durante la fase de entrenamiento se ajustan, y en esto consiste el aprendizaje, los denominados pesos del modelo, que se conocen como núcleos de convolución. Los modelos así ajustados son los que se utilizan para la clasificación de las diferentes imágenes.

En este trabajo se han utilizado dos modelos de CNN, concretamente AlexNet y GoogLeNet, que se describen a continuación. En cualquiera de estos modelos se realizan diferentes operaciones, con definición de una serie de parámetros, que constituyen la clave del proceso.

2.1 Operaciones aplicadas en las CNN

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,

$$C(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \qquad (2.1)$$

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.

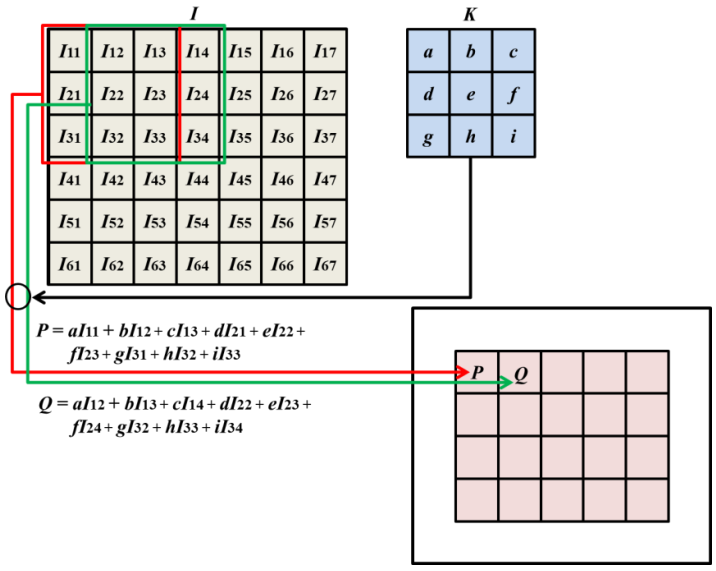


Figura 2.1. Ejemplo de convolución 2-D

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, $k = 3$.

$$\textit{Relación 5}:\ o = \left[\frac{i-k}{s}\right] + 1 \qquad \textit{Relación 6}:\ o = \left[\frac{i+2p-k}{s}\right] + 1 \tag{2.2}$$

2.1.2 ReLU

Esta función conocida como Unidad Lineal Rectificada (*ReLU*, *Rectified Linear Unit*) se define como $\mathrm{f(x)} = \max(0, \mathrm{x})$, 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.

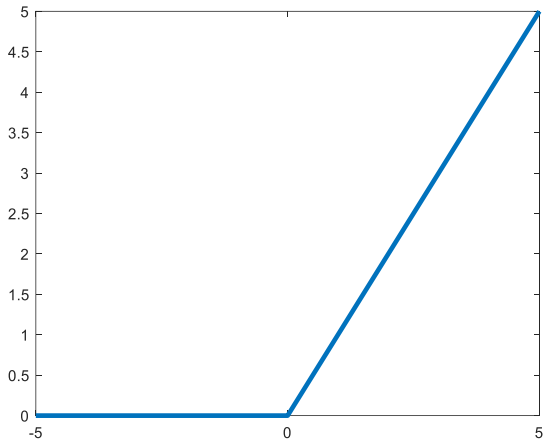


Figura 2.2. Representación de la función ReLU

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,

$$b_{x,y}^{i} = a_{x,y}^{i} \Big/ \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} \left(a_{x,y}^{i}\right)^{2}\right)^{\beta} \tag{2.3}$$

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 objeto 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.

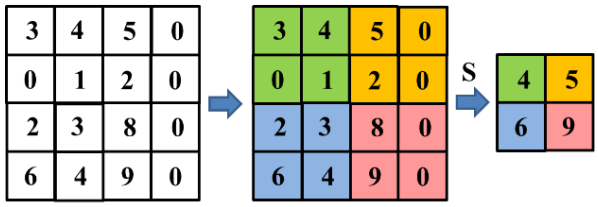


Figura 2.3. Max pooling

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

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

$$softmax(x)_i = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)} \tag{2.4}$$

2.1.7 Gradiente descendiente

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. 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 realizar 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 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 AlexNet

La red AlexNet (ImageNet, 2020; Russakovsky y col., 2015; Krizhevsky y col., 2021; BVLC AlexNet Model, 2021) como se ha mencionado previamente, es una de las utilizadas en el ámbito de las CNN, 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.

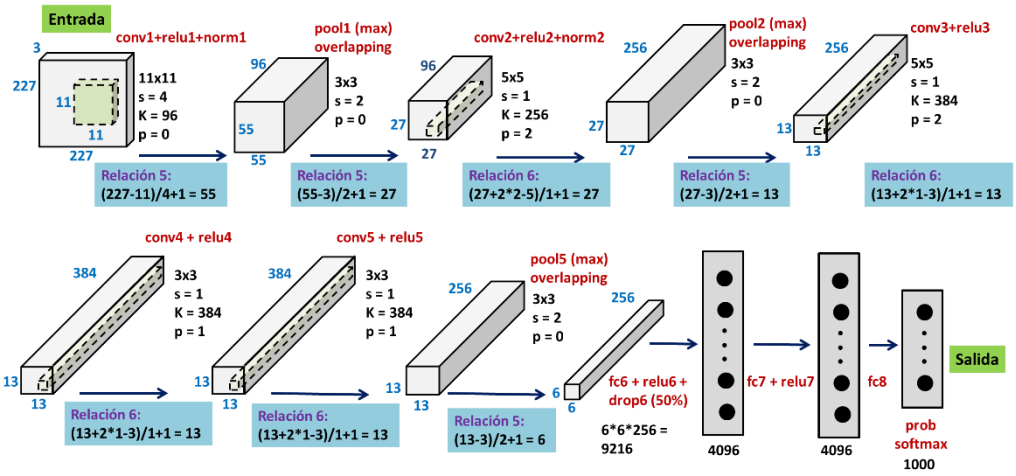


Figura 2.4. Modelo de red AlexNet

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 en 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), tal y como se explica posteriormente en la sección 4.2.2, donde se definen las distintas relaciones posibles, junto con su número correspondiente asociado.

En algunas implementaciones, Matlab (2021), tras la función *relu7* se realiza una operación de *dropout* del 50% modificando las dimensiones de la salida de esta capa.

2.3 GoogLeNet

Se trata de la red ganadora del concurso ILSVRC 2014. GoogLeNet (también conocida como *Inception V1*) de Google (BVLC GoogLeNet Model, 2021), proviene y se propuso en la publicación 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 que se usó se inspiró en LeNet, añadiendo un elemento novedoso denominado módulo *inception*. Un módulo *inception* (Inc) consta de varias convoluciones relativamente pequeñas para reducir drásticamente el número de parámetros. La arquitectura del modelo GoogleLeNet consiste en una CNN 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 otros 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 $4\times4\times512$ y $4\times4\times528$ 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).

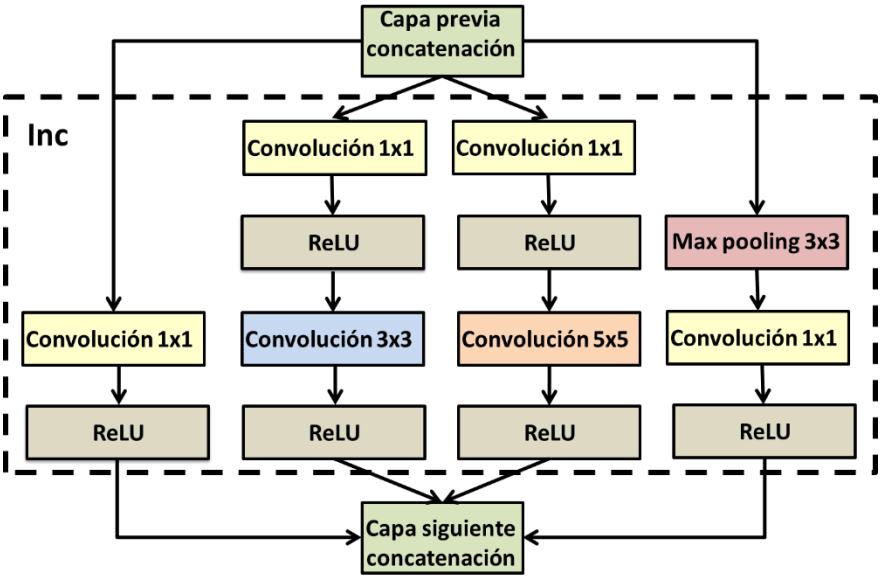


Figura 2.5. Módulo inception con incorporación de unidades ReLU

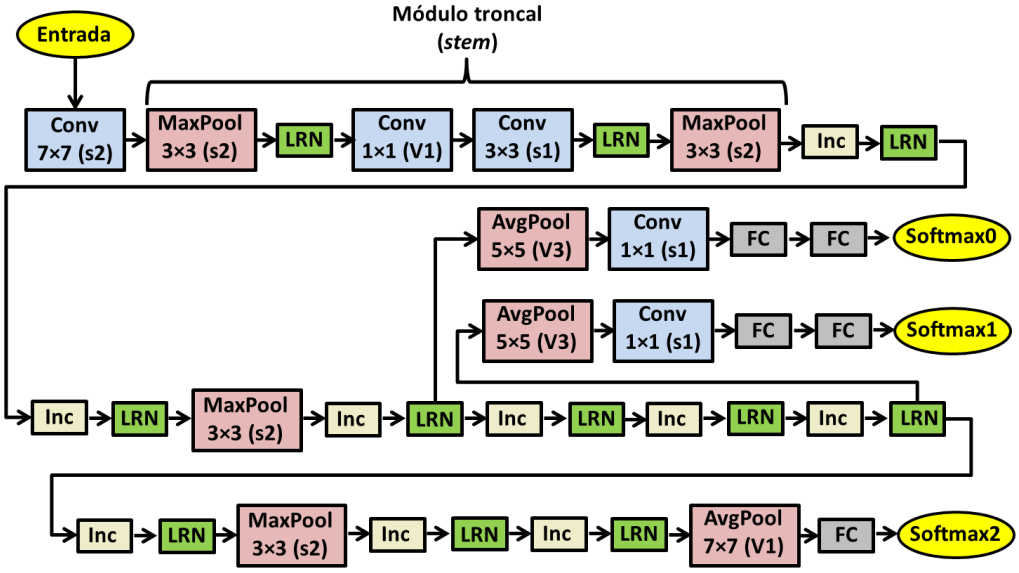


Figura 2.6. Modelo completo GoogLeNet (inception V1)

Capítulo 3 - Diseño de la aplicación

Una vez establecidas las bases teóricas del planteamiento formulado previamente, a continuación, se describe la arquitectura general de la propuesta, indicando los módulos y recursos utilizados en cada caso. Se muestran los flujos de datos entre los diferentes módulos, así como los correspondientes interfaces. Las funcionalidades de código desarrolladas y su ubicación se incluyen en el apéndice I.

3.1 Arquitectura del sistema

El modelo de arquitectura propuesto es el mostrado en la Figura 1.1, de suerte que para su funcionamiento es requisito indispensable disponer de conexión inalámbrica con la máxima calidad posible. Siguiendo el esquema de la mencionada figura, el dispositivo móvil captura las imágenes de los platos de comida para su evaluación. Por otra parte, el propio dispositivo móvil requiere tener por un lado activada la cámara asociada para la captura y comunicación con la nube, concretamente mediante Matlab Drive, que se identificará a partir de ahora simplemente como Drive. Además, se requiere una correcta sincronización entre el Computador y Drive.

Según la citada figura, existen conexiones entre los componentes de la arquitectura y en todos los casos bajo la mencionada cobertura inalámbrica.

Dentro de esta arquitectura se puede apreciar el módulo correspondiente a la plataforma ThingSpeak que es la plataforma específica de MathWorks (2021) para IoT. Las cabezas de flecha expresan la dirección del flujo de datos y, por tanto, las que aparecen en doble sentido indican posibilidad de transferencia en ambos sentidos.

A modo de resumen, los diferentes elementos que conforman la arquitectura son:

a) **Dispositivo Móvil:** es el elemento clave para la captura de imágenes de los alimentos a clasificar y conexión inalámbrica con Drive y ThingSpeak. También recibe los mensajes de Twitter.
b) **Matlab Drive:** repositorio para almacenar y procesar datos en la nube mediante los programas instalados en ella.

c) **ThingSpeak:** plataforma para servicios IoT en la nube, con el flujo de datos indicado.
d) **Computador:** plataforma para la realización de procesos con alta carga computacional, no soportados desde el móvil, ya que éste tiene limitado el tiempo de proceso a 240 segundos como máximo.

El flujo de datos y su procesamiento es el que se describe a continuación:

a) Bien en el Computador, a nivel local, o en Drive, a nivel remoto, se almacena el conjunto de imágenes para el entrenamiento de las CNN. Ambos deben estar perfectamente sincronizados para acceder al mismo espacio. Al Drive se accede desde el propio Computador. De la misma manera también se almacenan los programas para su ejecución.
b) Con los datos y los programas alojados en los espacios indicados, se realiza el entrenamiento de los modelos CNN, que puede ejecutarse en modo local o remoto en Matlab On-line (2021). Una vez realizado el entrenamiento, el modelo de red entrenado se almacena en Drive.
c) Desde el dispositivo móvil y a través de la app Matlab Mobile (2021), se solicita a través de Drive, la clasificación de la imagen captada con el dispositivo móvil. La imagen se clasifica mediante el modelo de red entrenado. A continuación, se establece la conexión con ThingSpeak para recuperar los datos nutricionales almacenados previamente en este espacio. Por otra parte, en este mismo espacio se almacenan los datos con la imagen clasificada, enviando el correspondiente mensaje vía Twitter, que se recibe en el dispositivo con acceso a esta red social.

3.2 Descripción de las unidades y módulos

Dentro de la arquitectura del sistema cada unidad o módulo tiene asignada una funcionalidad específica, que se describe a continuación.

3.2.1 Imágenes

Como paso previo al entrenamiento de las CNN es preciso estructurar las imágenes en las correspondientes clases o categorías, que se dividen en imágenes de entrenamiento y validación como se indica posteriormente.

En ambos casos, sobre ellas se aplica un proceso conocido como aumento de imágenes (*image augmentation*) consistente en añadir imágenes a las que se disponen en un momento determinado, aplicando operaciones de traslación, rotación y reflexión con el fin de mejorar la base de datos y preparar el sistema para ser entrenado con imágenes con el mayor número de variaciones posible. En este sentido, al conjunto de imágenes iniciales se le añade un 10% más de imágenes modificadas como se ha mencionado previamente.

3.2.2 Modificación de las capas de las CNN para el entrenamiento

Como se ha mencionado previamente, los modelos de redes neuronales utilizados son AlexNet y GoogLeNet, tratándose de modelos pre-entrenados para 1.000 categorías de objetos con la base de datos (ImageNet 2021) que consta de aproximadamente 1.3 millones de imágenes preparadas para el entrenamiento, 50 mil para la validación del modelo y 100 mil para test. En este sentido conviene resaltar dos aspectos fundamentales. Por un lado, el hecho de que el preentrenamiento implica que los pesos en las distintas capas de las redes poseen valores aprendidos con las imágenes indicadas y, por otro, que es necesario modificar la configuración de las redes para adaptarlas a las clases de salida que como se verá posteriormente son 10 en función del número de alimentos a identificar. Los modelos pre-entrenados tienen la ventaja de que, con un número relativamente bajo de ejemplos, se pueden obtener resultados satisfactorios.

En lo que respecta a la modificación, en la Figura 3.1 se muestran los dos ejemplos correspondientes a las últimas capas del modelo AlexNet, obtenidas mediante la herramienta *Deep Learning Network Analyzer* de Matlab observándose cómo la capa "*fc8*", se modifica a "*fc*", ambas totalmente conectadas, pasando de 1.000 conexiones a 10, coincidiendo en ambos casos con las categorías de clasificación para las que han sido entrenadas.

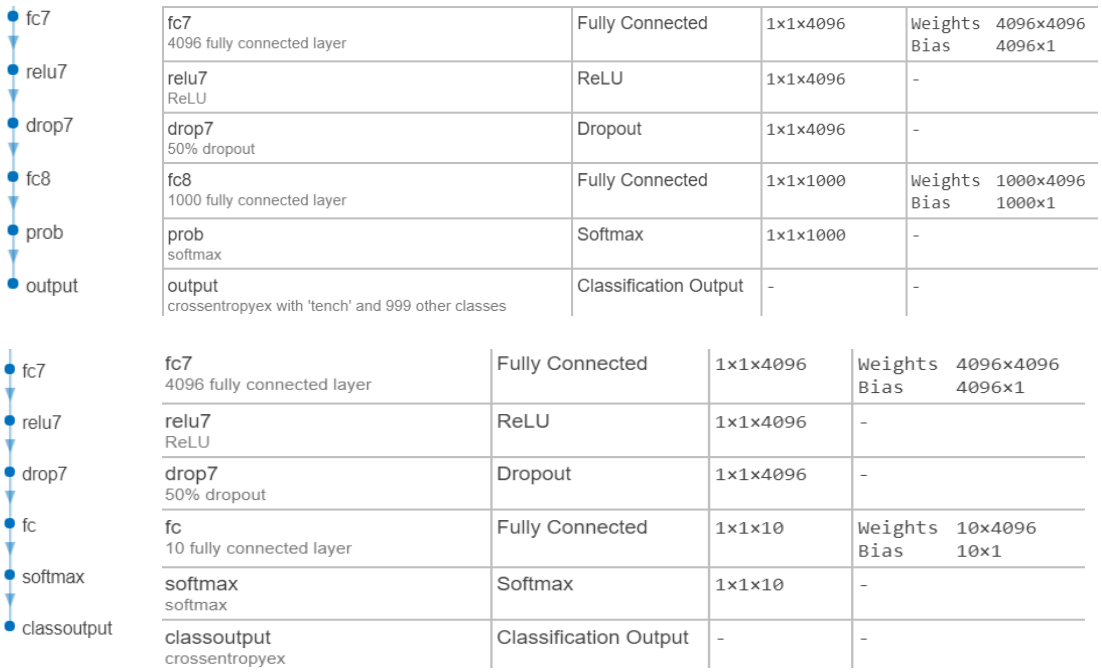
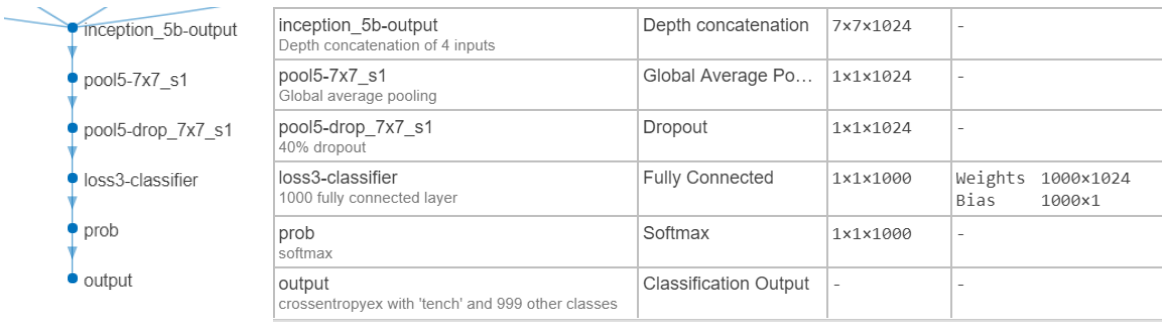
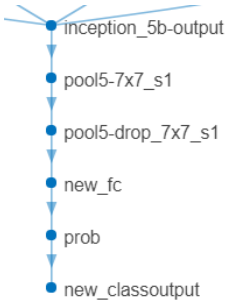


Figura 3.1. Modificación de las últimas capas de AlexNet

Por otra parte, en la Figura 3.2 se muestran los dos ejemplos equivalentes correspondientes a las últimas capas del modelo GoogLeNet, donde ahora se observa cómo la capa "*loss3-classifier*", se modifica a "*new_fc*" y, como en el caso anterior, ambas totalmente conectadas, pasando también de 1.000 conexiones a las 10 previstas.

En cualquiera de los dos casos, esta operación de re-entrenar una red con un nuevo conjunto de imágenes aprovechando los pesos ya ajustados y utilizar un nuevo conjunto de datos normalmente mucho más reducido, es lo que se conoce como transferencia de aprendizaje (*transfer learning*) en el ámbito del aprendizaje profundo.





inception_5b-output Depth concatenation of 4 inputs	Depth concatenation	7×7×1024	-
pool5-7x7_s1 Global average pooling	Global Average Po…	1×1×1024	-
pool5-drop_7x7_s1 40% dropout	Dropout	1×1×1024	-
new_fc 10 fully connected layer	Fully Connected	1×1×10	Weights 10×1024 Bias 10×1
prob softmax	Softmax	1×1×10	-
new_classoutput crossentropyex	Classification Output	-	-

Figura 3.2. Modificación de las últimas capas de GoogLeNet

3.2.3 Definición de los parámetros de entrenamiento

Para llevar a cabo el entrenamiento, es necesario establecer una serie de parámetros tal y como se indica en Pajares y col. (2021):

- ***Batch size:*** durante el proceso de actualización de los pesos de la red, mediante el cálculo del error y su retro-propagación mediante la técnica del gradiente descendente, se dispone de N imágenes de entrenamiento en un proceso iterativo. Esto significa que, al buscar el mínimo, se realizan una serie de pasos, siendo el objetivo de cada paso ajustar lo mejor posible los pesos. La dimensión de un lote (*batch*) define el número de imágenes utilizadas antes de llevar a cabo una actualización de los pesos de la red en el modelo.
- ***Epochs* e iteraciones:** define el número de veces que el conjunto total de muestras son procesadas por el algoritmo de optimización. Esto significa que cada muestra del conjunto N ha tenido una oportunidad de actualizar los pesos en cada *epoch*. También se puede fijar un número máximo de *epochs* para detener el proceso de entrenamiento cuando se alcanza dicho número, lo cual puede ser útil en el caso de que no se llegue a alcanzar la convergencia, esto es, que el error no sea aceptable.
- **Razón de aprendizaje (*learning rate*):** determina la rapidez con la que la red actualiza o ajusta los pesos involucrados. En el diseño propuesto se establece una razón de aprendizaje variable de forma que, sobre la razón fijada inicialmente, cada cierto número de *epochs* se aplica un determinado factor de reducción.

- **Método de optimización:** es el método aplicado para realizar la minimización de la función de coste.

3.2.4 Proceso de clasificación

Una vez finalizado el entrenamiento que, como se ha indicado previamente, se lleva a cabo en el computador o en Matlab On-Line, los diferentes pesos de las redes quedan actualizados para las nuevas imágenes de entrenamiento, materializándose el proceso de transferencia de aprendizaje. El modelo de red queda con estos pesos ajustados disponible y alojado en el Drive, que se utiliza para la clasificación de las nuevas imágenes captadas con la cámara del dispositivo móvil.

3.2.5 Descripción de ThingSpeak

Como se menciona en la sección 1.1, ThingSpeak es una plataforma específica para el tratamiento de datos en el ámbito de IoT. Permite visualizar los datos obtenidos de distintos dispositivos en tiempo real, así como llevar a cabo analíticas mediante la ejecución de código Matlab, o enviar alertas utilizando servicios web como Twitter, el cual es el caso del presente trabajo. Esta plataforma está dividida en canales de datos, los cuales se configuran para visualizar dichos datos en forma de gráficas.

Para la realización de este trabajo, se ha creado un canal en ThingSpeak, llamado "Nutrición'", para almacenar los datos nutricionales (grasas, proteínas, carbohidratos y calorías) de cada una de las clases de alimentos con las que han sido reentrenados los modelos de red utilizados. De esta forma, cuando una imagen es clasificada, se consultan en ThingSpeak los datos nutricionales correspondientes a su clasificación. Los datos almacenados son los que se muestran en la Tabla 3.1, donde se contienen los diez tipos de alimentos junto con sus aportes en grasas, proteínas, carbohidratos y calorías según las cantidades y magnitudes indicadas, en gramos (g) o kilocalorías (kcal). Este canal cuenta con 7 campos de datos. Los 5 primeros campos (*Field1 – Field5*) se corresponden con el nombre del alimento, y las grasas, proteínas, carbohidratos y calorías que aporta. Es decir, contienen los valores de los distintos componentes nutricionales de la Tabla 3.1.

Tabla 3.1. Contenido nutricional de distintos alimentos

Alimento	Grasas (g)	Proteínas (g)	Carbohidratos (g)	Calorías por 100g (kcal)
Churros	20.00	4.60	40.00	360
Ensalada-césar	2.10	3.20	4.30	44
Entrecot	11.44	29.50	0.00	222
Espaguetis-boloñesa	12.59	12.70	26.30	123
Flan	3.40	4.80	20.40	133
Helado	10.10	3.60	26.70	213
Huevos revueltos	20.72	10.20	0.80	229
Sushi	1.28	6.88	18.42	103
Tarta-chocolate	279.42	5.19	41.77	440
Tostada	3.7	7.30	45.70	249

En el campo número 6 (*Field6*), llamado "CaloriasAcumuladas", es donde se acumula el total de calorías de los alimentos cuyas imágenes han sido clasificadas, es decir, aquellos alimentos que se han ingerido. Por último, en el campo 7 (*Field7*), llamado "ActivarAcumulacion", se almacena un 1 cada vez que el campo de "CaloriasAcumuladas" es actualizado. En la Figura 3.3 pueden verse los campos del canal previamente mencionados.

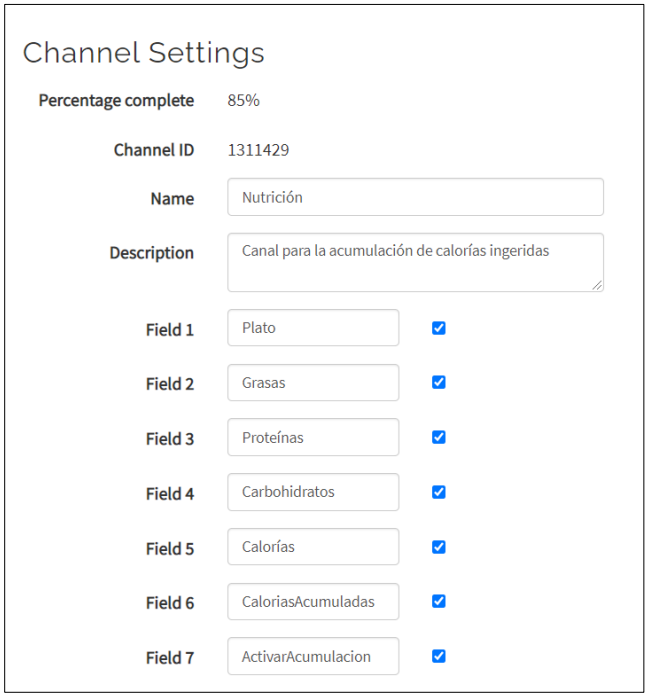


Figura 3.3. Configuración de los campos del canal de ThingSpeak

ThingSpeak no sólo se utiliza para consular los datos nutricionales de las distintas categorías de alimentos, sino que también se utiliza para visualizar en forma de gráfico

de líneas y de calibrador el total de las calorías acumuladas ingeridas, como puede verse en la siguiente Figura 3.4.

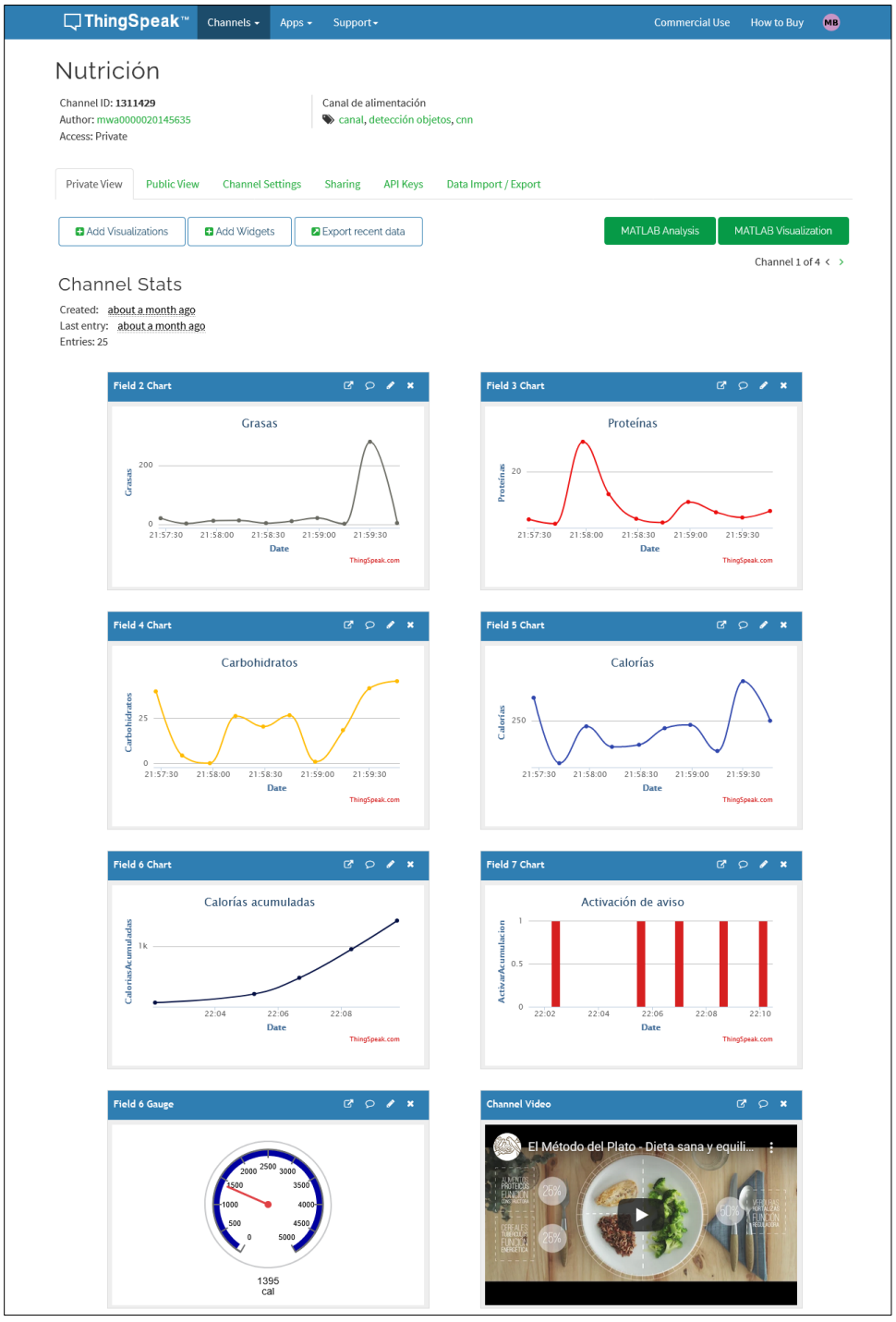


Figura 3.4. Canal de ThingSpeak con datos nutricionales

Para la publicación del mensaje de aviso mediante Twitter, se configuró lo que en ThingSpeak se conoce como "React"'. Un "React" es una acción que se dispara

cuando los datos del canal cumplen una determinada condición. En este caso, se ha configurado concretamente un "React" para el campo de las calorías acumuladas de forma que, cuando el valor de dicho campo supere, por ejemplo, las 2.000 calorías, se publique un tweet con un mensaje determinado. Esta configuración puede verse en la siguiente Figura 3.5.

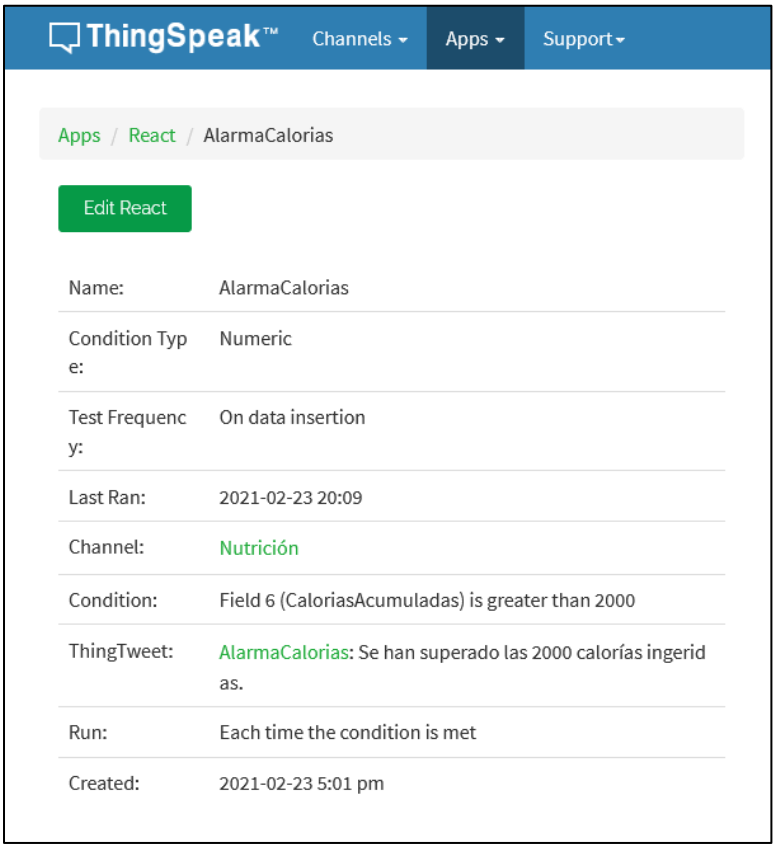


Figura 3.5. Configuración de la publicación de un tweet al superar una determinada cantidad de calorías

La cuenta en la que se publica el tweet se configura con lo que, como se ve en la Figura 3.5, se conoce como "ThingTweet". ThingTweet es un mecanismo interno de ThingSpeak que permite enlazar una cuenta de Twitter con una cuenta de ThingSpeak. De esta forma, a la hora de indicar la acción que realizará el React cuando la condición se cumpla, en este caso la publicación de un tweet, se podrá elegir una de las cuentas de Twitter que se hayan enlazado utilizando este mecanismo. En concreto, como se muestra en la Figura 3.6, la cuenta de Twitter con la que se enlazó el ThingSpeak es @AlarmaCalorias.

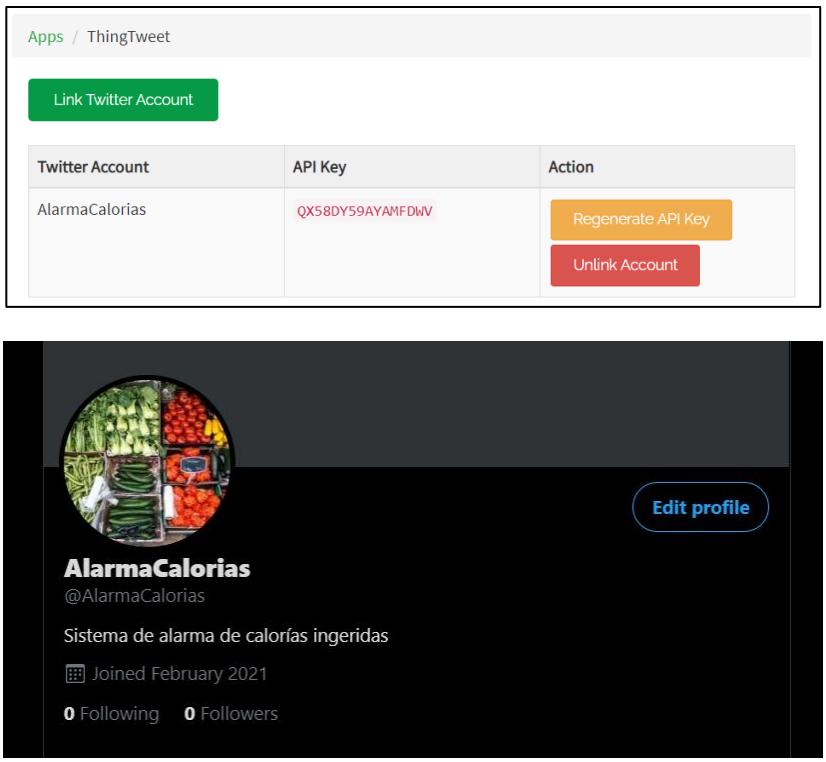


Figura 3.6. Configuración de ThingTweet

Capítulo 4 - Análisis de resultados

Seguidamente se describen los resultados obtenidos en los diferentes módulos que componen la aplicación desarrollada. Se comienza con la enumeración de los recursos materiales y herramientas utilizadas.

4.1 Recursos y herramientas

4.1.1 Hardware

- **Computador**, utilizado para el reentrenamiento de los modelos de red, así como para la carga de los datos nutricionales en ThingSpeak. En concreto, se ha utilizado un portátil Lenovo Z50-70 Tipo 80E7, con un procesador Intel Core i7 de 4ª generación y 16GB de RAM.
- **Dispositivo móvil**, para la captura de imágenes de los alimentos ingeridos. Se ha utilizado un Samsung Galaxy A6, con un procesador Octa Core de 64 bits y 3GB de RAM, con Android 10 como sistema operativo.

4.1.2 Software

- **App Mobile**, que se instala en el dispositivo móvil compartiendo espacio de almacenamiento con Matlab On-Line y con el computador través de Drive, además de con ThingSpeak. En todos los casos con conexión inalámbrica.
- **Matlab de escritorio** (2021), en la versión 2020b correspondiente a la entrega de septiembre de 2020, instalado de forma local en el computador, con sus correspondientes *Toolboxes*, incluyendo *Deep Learning* y *Computer Vision* como los más destacados, además de los modelos de redes AlexNet y GoogLeNet pre-entrenados.
- **Matlab On-Line** (2021), versión en la nube con la última versión siempre actualizada a la última entrega, en este caso la 2021a, correspondiente a marzo de 2021.

- **ThingSpeak**, plataforma IoT basada en la nube, soportando protocolos tales como API MQTT o REST, entre otros, y con capacidad de almacenamiento de datos y visualización de estos a través de cualquier navegador web con conexión inalámbrica o mediante la correspondiente App móvil instalada al efecto en el dispositivo. Permite el despliegue de programas Matlab para el procesamiento de datos con fines de análisis y monitorización de eventos o nuevos datos. También permite la creación de alertas y mecanismos de activación reactivos programados.
- **ThingView** (2021), App para dispositivos móviles que permite la visualización y monitorización de los datos almacenados en ThingSpeak.

4.1.3 Preparación de las imágenes

Las imágenes utilizadas para el reentrenamiento se obtuvieron del *dataset* contenido en la plataforma Kaggle (2021). En concreto, se utilizaron las imágenes de la competición *iFood – 2019 at FGVC6* (2019). Este *dataset* incluía imágenes de 251 clases distintas de alimentos, divididas en tres conjuntos, de entrenamiento, de validación y de *test*. El conjunto de entrenamiento contaba con 118.475 imágenes en total, dividas en unas 500 imágenes por clase. Por otro lado, el conjunto de *test* contaba con un total de 28.377 imágenes, y el conjunto de validación con 11.994.

Del conjunto de imágenes disponible se ha creado una base propia para el re-entrenamiento disponiendo en total de un conjunto de 300 imágenes distribuidas en diez clases, exactamente las que se indican en la Figura 4.1(a), de forma que el 70% se dedican al entrenamiento y el 30% restante a la validación durante el proceso de entrenamiento. En la Figura 4.1(b) se muestran algunas imágenes representativas del conjunto de datos mencionado, seleccionadas aleatoriamente del total. En ellas se pueden ver los distintos alimentos tal y como se presentan en una situación real, lo que permite valorar y estimar su naturaleza nutricional una vez clasificadas por el sistema. En este sentido conviene señalar la validez de la propuesta en este trabajo para su aplicación y utilización en un entorno real sin más que disponer de la configuración convenientemente adaptada y en cualquier caso, de cara a la propuesta de una solución de concepto, que permitiría desarrollar la misma o similar estrategia en un

nuevo ecosistema IoT bajo diferentes herramientas y aplicaciones sin más que sustituir los elementos desarrollados e integrados aquí trasladándolas al nuevo ecosistema.

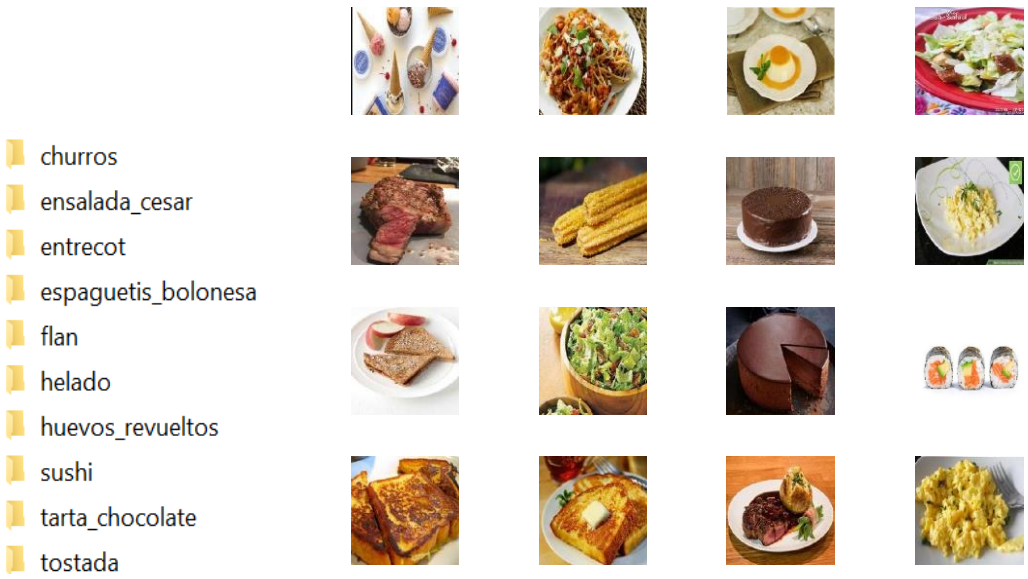


Figura 4.1. Categorías de alimentos y ejemplos seleccionados aleatoriamente de imágenes para el re-entrenamiento

En la Figura 4.2 se muestra un subconjunto representativo de las imágenes resultantes del proceso de aumento de imágenes como se ha indicado previamente, donde pueden observarse las deformaciones, rotaciones, el replicado de los alimentos o la sustitución del fondo original por el fondo negro entre otros.

Figura 4.2. Ejemplos de imágenes aumentadas

4.2 Entrenamiento de las CNN

Seguidamente se analizan los resultados correspondientes a la fase de entrenamiento en sendos modelos AlexNet y GoogLeNet.

Una cuestión relevante, antes de iniciar el proceso de entrenamiento, consiste en validar los nuevos modelos de red re-diseñados. Con tal finalidad se utiliza la herramienta *Deep Learning Network Analyzer* de Matlab que indica el número de capas del modelo, así como el número de avisos o errores. Cabe mencionar que, como es obvio, para la validación del modelo estos dos últimos deben de ser cero, en particular los errores. En la Figura 4.3(a) se observa el resultado del análisis para el modelo AlexNet donde se informa de las 25 capas analizadas de este modelo sin errores y sin *warnings*. La Figura 4.3(b) se corresponde con el modelo GoogLeNet analizado, donde las 144 capas rediseñadas aparecen igualmente libres de *warnings* y errores.

(*a*) (*b*)

Figura 4.3. Validación de los modelos re-diseñados: (a) AlexNet; (b) GoogLeNet

4.2.1 Modelo GoogLeNet

En primer lugar, se analiza el modelo GoogLeNet por su mejor comportamiento frente a AlexNet. En la Figura 4.4 se muestra la evolución del proceso de entrenamiento, en este caso del modelo de red GoogLeNet, concretamente habiendo llegado a la *epoch* 3 de las 6 totales que definen el proceso con un número máximo de 128 iteraciones. Como puede visualizarse, en el avance del proceso se muestra la evolución de la precisión (*accuracy*) y pérdida (*loss*) tanto en lo que respecta a la parte de entrenamiento (*training*) como de validación (*validation*). Se observa que el proceso comienza con valores muy desfavorables en lo que respecta a la precisión tanto en el entrenamiento como en la validación, con una clara y manifiesta tendencia positiva hacia el máximo en precisión y mínimo en pérdida, como cabe esperar de una buena evolución de cualquier proceso de aprendizaje.

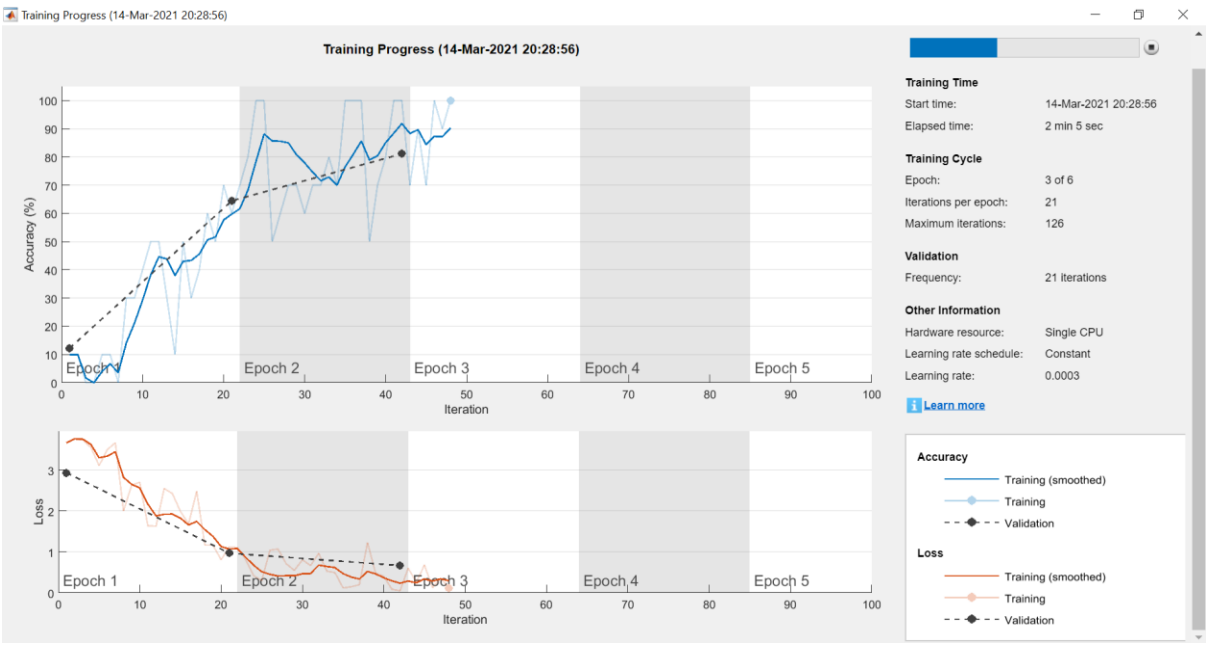


Figura 4.4. Visualización del progreso de entrenamiento del modelo GoogLeNet

El proceso de entrenamiento culmina tal y como aparece en la Figura 4.5, logrando una precisión en validación del 88.89%, que resulta ser suficientemente aceptable, frente al 100% que sería el ideal, en el contexto realizado y teniendo en cuenta el número relativamente reducido de imágenes utilizadas para el entrenamiento. Esto pone de manifiesto la importancia del proceso de transferencia de aprendizaje, de forma que a pesar del reducido número de imágenes utilizadas para el reentrenamiento, el resultado final puede considerarse válido. En la misma figura se observa que el tiempo total empleado en el proceso de entrenamiento ha sido de 5 minutos y 20 segundos, que es más que aceptable bajo las condiciones específicas del proceso y teniendo en cuenta que se ha llevado a cabo en una CPU de propósito general. El proceso de aprendizaje se ha realizado bajo una razón de aprendizaje de 0.0003 manteniéndose constante durante todo el proceso. En cuanto a la evolución de la precisión se observan ciertas oscilaciones aproximándose al 100% en determinados tramos del proceso, para al final decaer parcialmente hasta situarse en el valor indicado previamente. En cuanto a la evolución de la función de pérdida se observa un comportamiento más estable, acercándose rápidamente a valores cercanos a cero, valor ideal, y manteniendo este comportamiento prácticamente hasta el final, lo que en términos generales supone un proceso global aceptable.

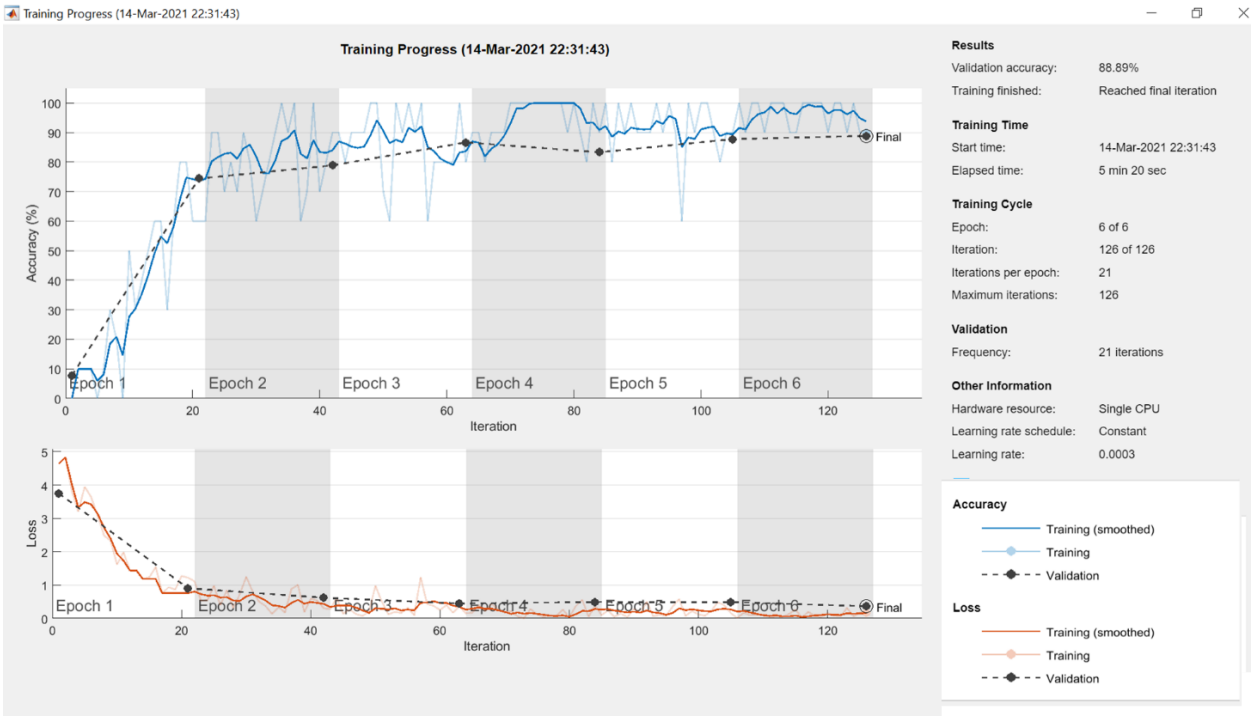


Figura 4.5. Finalización del entrenamiento de GoogLeNet

El método de optimización utilizado ha sido el Gradiente Estocástico Descendente (SGD), con un *batch size* de 10 y 6 *epochs* con 21 iteraciones por cada *epoch*, haciendo un total de 126 iteraciones. En cada *epoch* se realiza lo que se conoce como *shuffle* que consiste en entremezclar (barajar) las imágenes que participan en esa *epoch*. La razón de aprendizaje es la indicada previamente. No se ha establecido la detención del proceso tras un determinado número de *epochs* o iteraciones cuando no se consiguen resultados de optimización aceptables, por lo que en este caso se ha dejado evolucionar hasta alcanzar el límite de iteraciones prefijado, a este parámetro se le suele conocer como *patience*, que en este caso se ha fijado a infinito.

En la Figura 4.6 se muestra la matriz de confusión en lo que respecta a los resultados de validación observándose, de nuevo, unos resultados aceptables logrando varios porcentajes de aciertos del 100% en cuanto a las clases predichas y verdaderas.

En la diagonal principal se muestran los resultados correspondientes a las decisiones correctas, que son ciertamente mayoritarias en todos los casos. En cualquier caso, los resultados son aceptables. Analizando los resultados por filas, y en orden a clarificar los resultados, en la fila 1, por ejemplo, de 8 platos correspondientes a la

categoría de "churros", el sistema predice 6 como tal y 2 como "helado", siendo el peor caso de clasificación el de la fila 3, donde de los 14 posibles platos de "entrecot" sólo 9 han sido clasificados como tal. Por otra parte, aparecen 6 categorías con porcentajes logrados del 100%.

El análisis por columnas se refiere al estudio de las categorías predichas frente a las verdaderas, observándose que en este caso son cuatro las categorías que consiguen un 100%. En el caso, por ejemplo, en la categoría de "churros" se ha predicho un total de 9 datos, de los cuales sólo 6 han sido correctamente clasificados, y los otros tres han sido predichos incorrectamente. Lo mismo ocurre en lo que se refiere a la categoría de "helado". Estos dos últimos casos se corresponden con los peores resultados del modelo.

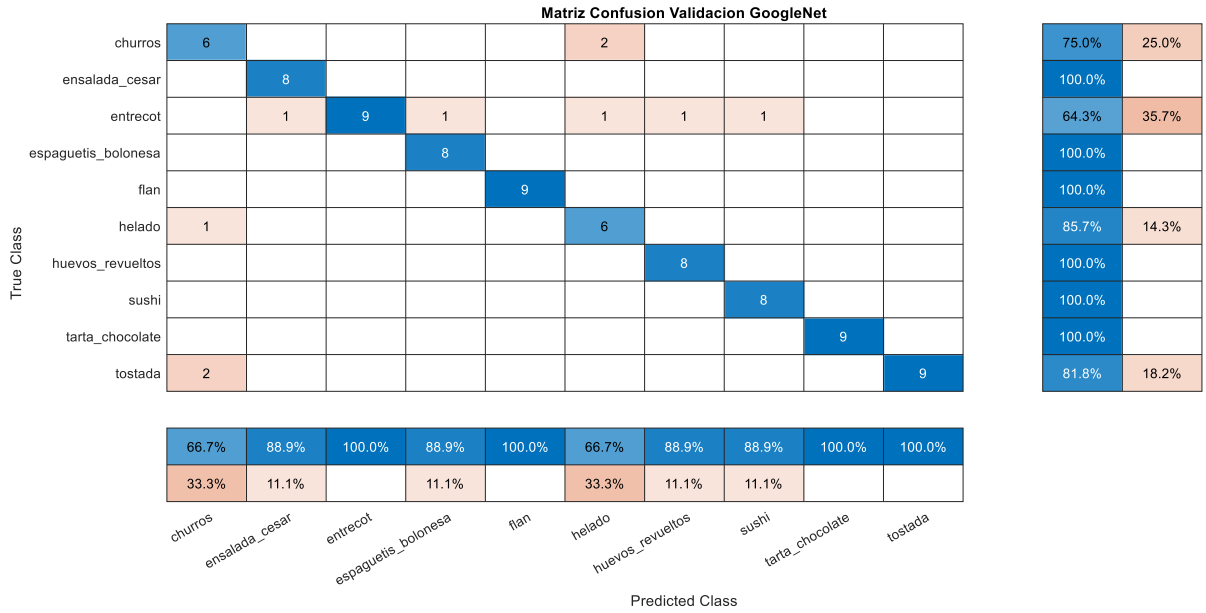


Figura 4.6. Matriz de confusión del modelo GoogLeNet

En la Figura 4.7 se muestran algunos resultados de clasificación en los ejemplos de validación indicando las probabilidades con las que se han seleccionado frente al resto. Se observan unos resultados ciertamente satisfactorios, siendo el menor de ellos de 71.2%, mientras que existen varios porcentajes muy próximos al 100%, lo cual es altamente satisfactorio.

Los errores finales de entrenamiento y validación han resultado ser respectivamente de 1,7% y 12.5%.

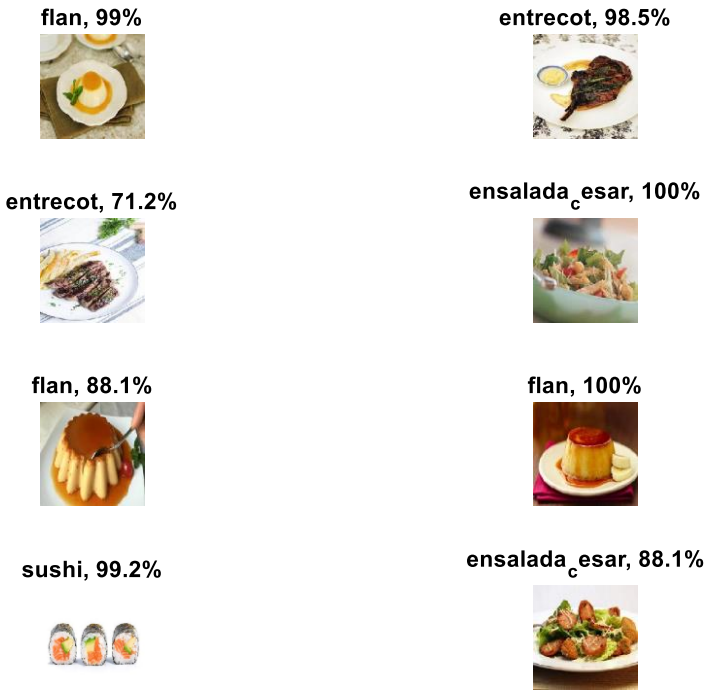


Figura 4.7. Probabilidades de clasificación en GoogLeNet

En cualquier caso y como conclusión general, los resultados de validación del modelo son ciertamente aceptables, lo que confirma de nuevo la eficiencia de la transferencia de aprendizaje.

Por otra parte, conviene señalar que se han llevado a cabo diversos procesos de entrenamiento, con variación de parámetros habiendo conseguido resultados similares por lo que se reporta uno de los más favorables en este caso.

Como es bien sabido, tras la fase de entrenamiento los pesos del modelo se han actualizado para conseguir los resultados indicados previamente. En el modelo GoogLeNet la primera capa de convolución posee 64 filtros, de forma que $K = 64$, en todos los casos de dimensión 7x7x3. Los pesos de esta primera capa se visualizan en la Figura 4.8, los 64 filtros se muestran situados en una matriz de 8x8 filtros, observándose cómo en cada uno de ellos aparecen representados los valores de activación, de forma que a mayores niveles de intensidad mayores son los valores representados y por tanto mayores los niveles de actuación en cada filtro. Fijándose en uno de ellos, por ejemplo, en el de la fila 1 y columna 3, se observan claramente los mencionados niveles blancos y negros con claridad, lo mismo que en otros varios de ellos.

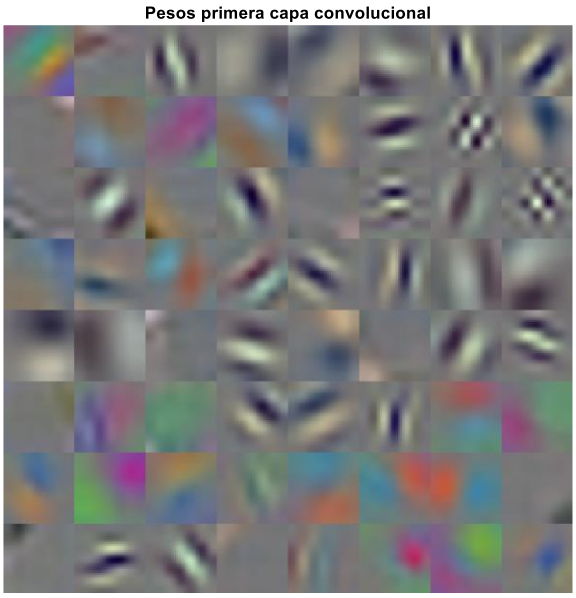


Figura 4.8. Representación de los pesos en la primera capa de convolución del modelo GoogLeNet

4.2.2 Modelo AlexNet

En el modelo AlexNet el proceso de entrenamiento finaliza según se muestra en la Figura 4.9, logrando una precisión en validación del 85.56%, también relativamente aceptable como en el caso anterior, consecuencia igualmente del proceso de transferencia de aprendizaje.

En este caso, el tiempo total de entrenamiento ha sido de 3 minutos y 20 segundos en el mismo equipo en el que se realizó el entrenamiento con GoogLeNet. El conjunto de imágenes utilizado para el entrenamiento y la validación es el mismo que en el modelo GoogLeNet, habiéndose aplicado también el mismo conjunto de parámetros durante el aprendizaje, lo cual permite de alguna manera la posibilidad de análisis comparativo entre ambos modelos.

Durante la evolución del proceso no se llega en ninguna *epoch* al 100% con ciertas oscilaciones, aunque bien es cierto que desde el inicio se progresa en la buena dirección, tanto en cuanto a la precisión como a la función de pérdida se refiere. En el primer caso tendiendo hacia el 100% y en el segundo hacia el valor cero.

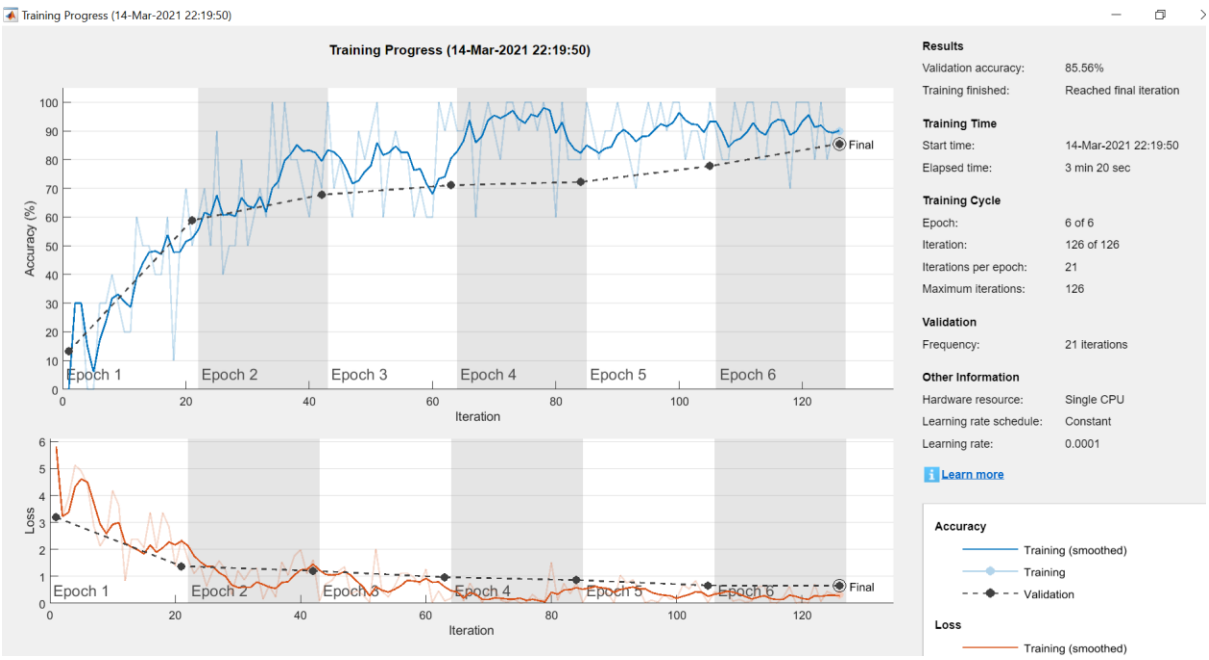


Figura 4.9. Finalización del entrenamiento de AlexNet

En la Figura 4.10 se muestra la matriz de confusión correspondiente al modelo AlexNet. Se observan también varios porcentajes de aciertos del 100% en cuanto a las clases predichas y verdaderas. La valoración de la matriz admite comentarios similares que en el modelo GoogLeNet. Los errores globales de entrenamiento y validación son en este caso del 1.9% y del 14.4%, algo más altos que en el modelo anterior.

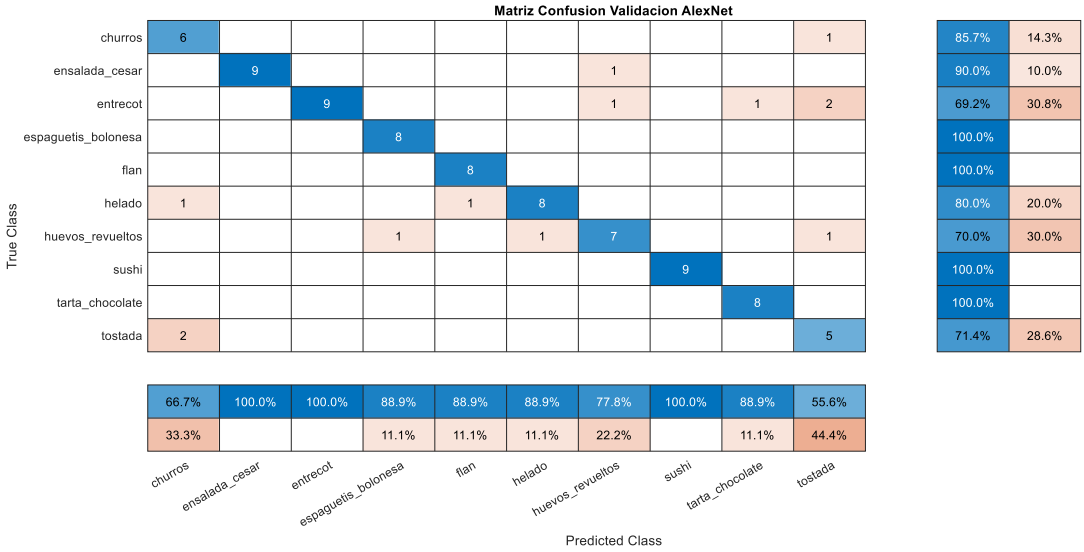


Figura 4.10. Matriz de confusión del modelo AlexNet

En relación con los resultados de clasificación en los ejemplos de validación, en la Figura 4.11 se muestran las probabilidades de selección frente al resto. Destaca en negativo el hecho de la clasificación de la imagen de la fila 2 en la columna 1, donde un plato de "churros" ha sido clasificado como "flan" con un 39.5% de probabilidad, lo que indica que en este caso "flan" ha sido la categoría ganadora, pero con un porcentaje muy bajo. No obstante, el resto de los porcentajes son más que aceptables, consiguiendo que cinco categorías alcancen porcentajes superiores al 95%.

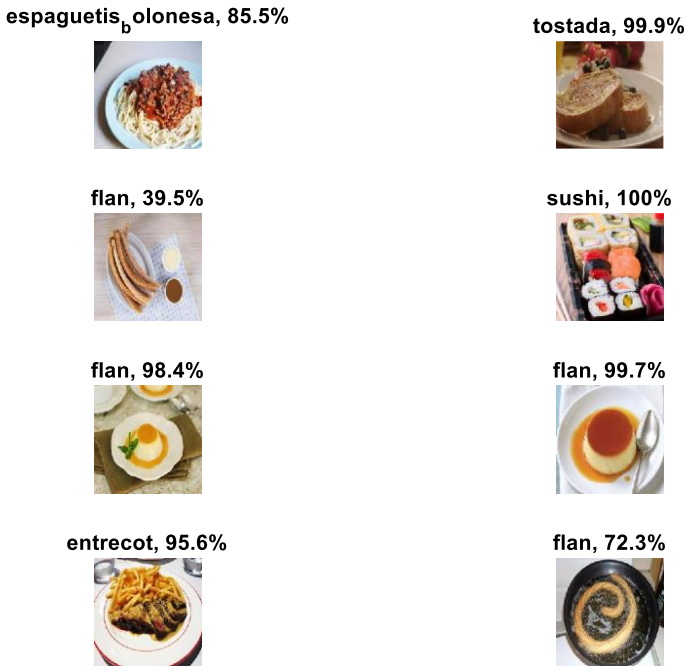


Figura 4.11. Probabilidades de clasificación en AlexNet

En el modelo AlexNet la primera capa de convolución posee 96 filtros, de forma que el tamaño del núcleo es $K = 96$, en todos los casos de dimensión $11x11x3$, tal y como se muestra en la Figura 2.4. Los pesos de esta primera capa se muestran en la Figura 4.12, donde se pueden hacer observaciones similares a los resultados mostrados para GoogLeNet relativos a los niveles de activación de los filtros en función de las intensidades mostradas en los distintos núcleos. Es decir, valores representados con tonalidades claras representan mayores niveles de activación que los valores con tonalidades oscuras e incluso negras, que no tienen efecto alguno sobre el resultado filtrado de las imágenes una vez que éstas pasan por los correspondientes núcleos de los filtros.

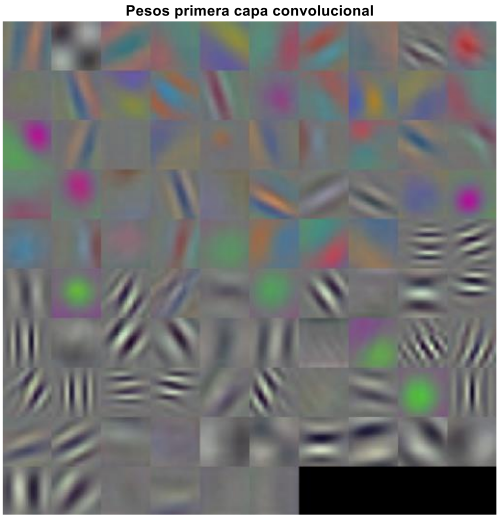


Figura 4.12. Representación gráfica de los pesos en la primera capa de convolución del modelo AlexNet

4.3 Clasificación

Una vez finalizado el proceso de entrenamiento para ambos modelos se puede llevar a cabo el proceso de clasificación de imágenes.

Para ello se sigue el esquema mostrado en la Figura 1.1. La captura de las imágenes a clasificar se lleva a cabo mediante el dispositivo móvil ejecutando la aplicación de Matlab. En Matlab se llama a la función `snapshot(cam, 'manual')`, la cual activa la cámara del dispositivo para la captura manual de imágenes. Una vez capturada la imagen, se clasifica mediante los modelos previamente entrenados, obteniéndose una clasificación como la que puede verse en la Figura 4.13.

Como puede verse, en la clasificación de la imagen se incluye el nombre de la categoría de alimento a la que pertenece, junto con la información nutricional asociada a dicha categoría. Esta información se compone de los gramos de grasas (G), proteínas (P) y carbohidratos (Ch), y de las calorías (Ca) que aporta la categoría de alimento especificada.

Cabe mencionar, que no todas las clasificaciones llevadas a cabo son correctas, pudiéndose obtener clasificaciones erróneas como es el ejemplo de la Figura 4.14. En este ejemplo puede verse cómo un plato de "huevos revueltos" ha sido clasificado como un plato de "espaguetis boloñesa". Otros ejemplos diferentes de clasificación se muestran en el Apéndice II.

Figura 4.13. Clasificación correcta de un plato de sushi

Figura 4.14. Clasificación errónea de un plato de huevos revueltos

4.4 ThingSpeak y Twitter

Tras la clasificación de la imagen captada por el dispositivo móvil se procesan las calorías correspondientes al resultado de la clasificación. Tras mostrar la imagen junto con el resultado de la clasificación y su información nutricional, tal y como se ha descrito en la sección anterior, se lleva a cabo la actualización del campo de las calorías acumuladas y la activación del campo "ActivarAcumulacion" de ThingSpeak.

En la siguiente Figura 4.15 puede verse el número de calorías acumuladas y el número de veces en las que las calorías acumuladas fueron actualizadas, respectivamente.

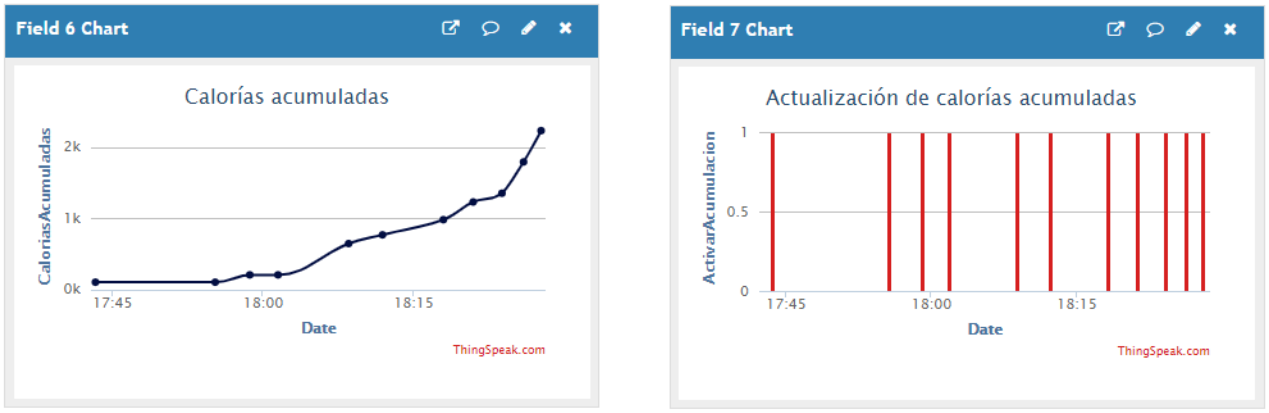


Figura 4.15. Representación de las calorías acumuladas y el número de veces que fueron actualizadas en el canal de ThingSpeak

Una vez superado el umbral de calorías acumuladas configurado en el React mencionado en la sección 3.2.5, en la Figura 4.16 puede verse la última ejecución de éste.

Name	Created	Last Ran
AlarmaCalorias View Edit	2021-02-23	2021-06-01 5:07 pm

Figura 4.16. Datos de la última ejecución del React de Matlab

Esto hace que, tal y como aparece en la Figura 4.17, se genere un tweet indicando que se han superado, en este caso, las 2.000 calorías ingeridas, junto con la fecha en la que se generó dicha alarma.

Figura 4.17. Perfil de la cuenta @AlarmaCalorias con los mensajes de alarma generados por ThingTweet

Capítulo 5 - Conclusiones y trabajo futuro

5.1 Conclusiones

El presente trabajo propone el diseño y la implementación de una solución que permite la monitorización de la ingesta de calorías mediante la clasificación de imágenes de comida, utilizando técnicas de aprendizaje profundo dentro de un contexto de *Internet of Things*.

La solución presentada supone una estructura completa de captura y clasificación de imágenes, así como de visualización de la información extraída de dicha clasificación, mediante la publicación de los resultados obtenidos en *dashboards*; y monitorización de la misma mediante la generación de alarmas. De este modo, se ha comprobado la viabilidad de la solución dentro de un contexto de control de ingesta de calorías.

Para el desarrollo de este trabajo, en primer lugar, se creó una base de datos de imágenes de distintas categorías de alimentos con las que llevar a cabo el entrenamiento de los modelos de redes CNN.

Una vez creada la base de datos, se llevó a cabo el reentrenamiento de dos modelos de redes CNN (AlexNet y GoogLeNet) para la clasificación de diez categorías de alimentos distintas, demostrando la utilidad del método de transferencia de aprendizaje para casos en los que se disponen de un número reducido de imágenes de entrenamiento.

Por último, se llevó a cabo el diseño y la configuración de un entorno de procesamiento de datos utilizando la plataforma ThingSpeak para la visualización de los mismos, con el envío de alarmas mediante Twitter.

El balance final del presente trabajo es positivo, habiéndose cumplido todos los objetivos establecidos al principio de esta memoria, y habiéndose obtenido una solución completa que ofrece la posibilidad de monitorizar de forma satisfactoria la ingesta de calorías.

5.2 Trabajo futuro

Una vez demostrada la viabilidad de la solución planteada, a continuación, se enumeran una serie de mejoras a realizar a futuro:

a) Implementación, mediante la funcionalidad de *Matlab analysis*, la generación de un reporte diario, semanal o mensual que incluya la cantidad de las calorías ingeridas junto con su fecha y hora de ingesta, un promedio de calorías ingeridas, días con mayor ingesta de calorías, por ejemplo.

b) Implementación del formateo de las calorías acumuladas, de forma que cada día se comience con un total de 0 calorías acumuladas.

c) Modificación del proceso de envío de datos a ThingSpeak para que se actualicen no sólo las calorías acumuladas, sino también las grasas, proteínas y carbohidratos acumulados.

d) Entrenamiento de modelos de predicción de ingesta de calorías.

e) Aumento del número de categorías de alimentos que los modelos son capaces de clasificar, así como el aumento de la base de datos de imágenes para mejorar el proceso de entrenamiento de los modelos de red.

f) Investigación de la integración de Matlab con tecnologías para el desarrollo de aplicaciones móviles, de forma que se pueda contar con una interfaz más *user friendly* a la hora de llevar a cabo la captura de los alimentos a clasificar.

Chapter - Introduction

Preliminaries

Every day, technological progress is making it possible to efficiently increase the number of applications for monitoring people's nutrition instantaneously. There are several applications for mobile devices developed in recent years for this purpose. In fact, for example, in Time2Feat (2020) specific details are provided on six applications that allow the calculation of daily calorie intake, such as My Fitness Pal, Fat Secret, Yazio, Lose it!, Macros or Easyfit. These applications allow, in general terms, to keep a temporary daily, weekly or monthly record of food consumption in terms of calories. In addition, some of them can be connected to physical activity tracking devices to establish the interrelation between calorie intake and physical activity.

In the field of the control of certain diseases such as diabetes nutrition is an essential element for health. Along this line, mobile applications have also been developed for nutritional control (DiabeWeb, 2021) this is the case of My Fitness Pal, Foodmeter, Diabetes Menu, myDiabeticAlert or Diabetes a la Carta.

On the other hand, there is extensive literature on the use of machine learning methods, and more specifically, those based on Convolutional Neural Networks in the identification and classification of food. Such is the case of the work proposed by Zhou et al. (2019) on the basis of food control with the aim of improving the health of the population, who reviews the works in this regard, highlighting among others those mentioned next. Heravi et al. (2018) designed a modified network based on AlexNet (Russakovsky et al., 2015) achieving accuracy values around 95% in a set of 1,316 images and 13 categories. Mezgec and Seljak (2017) also developed a model based on AlexNet, NutriNet, for food and beverage identification, training it with more than 225 thousand images, achieving accuracies of around 94%. Fu et al. (2017) used ResNet (He et al., 2016) as a model, achieving accuracies of about 68%. Herruzo et al. (2016) used the GoogLeNet model with different accuracy results depending on the dataset used, reaching 97% for a specific dataset such as FoodCAT (Wu et al., 2015).

Based on the aforementioned applications, and under the inspiration of CNN type models, this paper proposes an intelligent application based on Deep Learning techniques under the Internet of Things (IoT) paradigm. Although the architectural model of the system is described in greater detail later, **¡Error! No se encuentra el origen de la referencia.**.1 provides the general scheme with its main components, thus establishing the starting point of the present work.

1) There is a central processor conveniently equipped where the training of the neural network models is performed, although they could also be trained in the cloud Matlab Drive (2021), so that, in any case, they are accessible from a mobile device.
2) Using the mobile device, connected to the cloud, an image of a food item is captured and classified with the CNN network model selected each time.
3) With the result of the classification, the relevant query is made to the application ThingSpeak (2021), specific for data processing in the field of IoT. Here, different data related to each food (calories, fats, proteins, carbohydrates) is stored, and the result is returned via Twitter.

Objectives

The proposed overall objective is to design an IoT system, as a conceptual solution for food classification, with the ability to perform an instant informative control on the nature of each food in relation to its nutritional potential, while a trace of the nutritional components can be maintained over time. Under this perspective, an IoT solution proposal is made.

The specific objectives formulated are the following,

- Design and adaptation of CNN models for training and decision making in classification.
- Configuration of the data structure and data processing in the cloud, including the sending of informative messages via Twitter.
- Integration of the modules that implement the above processes.
- Analysis and evaluation of the results

The work plan foreseen is based on the specific objectives outlined above, contemplating the same number of phases as objectives, and with a balanced planning by weeks throughout the development time of the project.

Chapter - Conclusions and future work

Conclusions

This paper proposes the design and implementation of a solution that allows the monitoring of calorie intake by classifying food images, using deep learning techniques within an Internet of Things context.

The solution presented involves a complete structure for capturing and classifying images, as well as for visualizing the information extracted from this classification through the publication of the results obtained in dashboards, and monitoring of the same by generating alarms. In this way, the viability of the solution has been tested within a calorie intake control context.

For the development of this work, first, a database of images of different food categories was created with which to carry out the training of the CNN network models.

Once the database was created, re-training of two CNN network models (AlexNet and GoogLeNet) was carried out for the classification of ten different food categories, demonstrating the usefulness of the transfer learning method for cases where a small number of training images are available.

Finally, a data processing environment was designed and configured using the ThingSpeak platform for data visualization, with the sending of alarms via Twitter.

The final performance of the present approach is positive, having met all the objectives established at the beginning of this report, and having obtained a complete solution that offers the possibility of satisfactorily monitoring calorie intake.

Future work

Once the feasibility of the proposed solution has been demonstrated, several improvements to be applied in the future are listed below:

a) Implementation, through Matlab analysis functionality, of the generation of a daily, weekly or monthly report that includes the amount of calories

ingested with the date and time of intake, an average of calories ingested, or days with higher calorie intake, for example.

b) Implementation of the formatting of accumulated calories, so that each day starts with a total of 0 accumulated calories.

c) Modification of the process of sending data to ThingSpeak so that not only accumulated calories, but also accumulated fat, protein and carbohydrates are updated.

d) Training of calorie intake prediction models.

e) Increasing of the number of food categories that the models are able to classify, as well as increasing the image database to improve the training process of the network models.

f) Investigation of the integration of Matlab with technologies for the development of mobile applications, in order to have a more user friendly interface for the capture of the food to be classified.

BIBLIOGRAFÍA

1. Apps Diabetes (2021). Apps para personas con diabetes. Disponible on-line: https://www.diabeweb.com/blog/21/las-5-mejores-apps-sobre-nutricion-para- (último acceso: Marzo 2021).
2. Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*, Springer, NY, USA.
3. BVLC AlexNet Model (2021). Disponible on-line: https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet (último acceso: Marzo 2021).
4. BVLC GoogleNet Model (2021). Disponible on-line: https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet (último acceso: Marzo 2021).
5. Fu, Z. H., Chen, D., Li, H. Y. (2017). ChinFood1000: A large benchmark dataset for Chinese food recognition. In D. S. Huang, V. Bevilacqua, P. Premaratne, & P. Gupta (Eds.), Intelligent Computing Theories and Application, ICIC 2017, Pt I (Vol. 10361, pp. 273–281).
6. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proc. of the IEEE conference on computer vision and pattern recognition, pp. 770-778.
7. Heravi, E. J., Aghdam, H. H., Puig, D. (2018). An optimized convolutional neural network with bottleneck and spatial pyramid pooling layers for classification of foods. Pattern Recognition Letters, 105, 50–58.
8. Herruzo, P., Bolaños, M., Radeva, P. (2016). Can a cnn recognize Catalan diet? In AIP Conference Proceedings (Vol. 1773).
9. iFood – 2019 at FGVC6 (2019). Disponible on-line: https://www.kaggle.com/c/ifood-2019-fgvc6 (último acceso: Noviembre 2020).
10. ImageNet LSVRC (2012). Large Scale Visual Recognition Challenge 2012 (ILSVRC2012). Disponible on-line: http://www.image-net.org/challenges/LSVRC/2012/ (último acceso: Marzo 2021).
11. Imagenet (2021). Disponible on-line: http://www.image-net.org (último acceso: Marzo 2021).
12. Kaggle. 2021. https://www.kaggle.com/ (último acceso: Mayo 2021).

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. Mathworks (2021). Makers of Matlab and Simulink. Disponible on-line: https://es.mathworks.com/ (ultimo acceso: Marzo 2021)
15. Matlab (2021). Deep Learning Toolbox. Disponible on-line: https://es.mathworks.com/products/deep-learning.html (último acceso: Junio 2021).
16. Matlab Drive (2021) Disponible on-line: https://es.mathworks.com/products/matlab-drive.html. (último acceso: Junio 2021).
17. Matlab Mobile (2021). Disponible on-line: https://play.google.com/store/apps/details?id=com.mathworks.matlabmobile&hl=es&gl=US (ultimo acceso: Junio 2021).
18. Matlab On-line (2021) Disponible on-line: https://es.mathworks.com/products/matlab-online.html (último acceso: Marzo 2021).
19. Mezgec, S., & Seljak, B. K. (2017). NutriNet: A deep learning food and drink image recognition system for dietary assessment. Nutrients, 9(7), 657.
20. Pajares, G., Herrera, P.J., Besada, E. (2021). Aprendizaje Profundo. RC-Libros.
21. Ruder, S. (2017). An overview of gradient descent optimization algorithms. arXiv:1609.04747v2 [cs.LG].
22. Russakovsky, O., Deng, J., Su, H., Krause, J. (2015). Imagenet large scale visual recognition challenge. International Journal of Computer Vision 115.3: 211-252.
23. 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].
24. ThingSpeak (2021). ThingSpeak for IoT Projects. Disponible on-line: https://thingspeak.com/. (último acceso: Junio 2021).
25. ThingView (2021). ThingSpeak viewer. Disponible on-line: https://play.google.com/store/apps/details?id=com.cinetica_tech.thingview&hl=es_419&gl=US. (último acceso: Junio 2021).

26. Time2Feat (2021). App para calcular las calorías consumidas. Disponible on-line: https://time2feat.com/contador-calorias-alimentos-apps/ (último acceso: Marzo 2021).
27. Wu, R., Yan, S., Shan, Y., Dang, Q., Sun, G. (2015). Deep image: Scaling up image recognition. CoRR, arXiv:1501.02876, 2015. URL http://arxiv.org/abs/1501.02876.
28. Zhou, L., Zhang, C., Liu, F., Qiu, Z., He, Y. (2019). Application of Deep Learning in Food: A Review. Comprehensive Reviews in Food Science and Food Safety, 18(6), 1793-1811.

APÉNDICE I: PROGRAMAS EN MATLAB

El código desarrollado junto con las imágenes empleadas para el reentrenamiento de los modelos de red se encuentra en el siguiente repositorio: https://github.com/mvicbar/MMIOT_TFM. La carpeta raíz del proyecto, llamada "MMIOT_TFM", contiene a su vez dos carpetas "Codigo" y "Dataset". A continuación, se explica el contenido de cada una de estas carpetas, junto con el proceso a seguir para ejecutar los programas.

A. Carpeta "Dataset"

En esta carpeta se incluyen las imágenes con las cuales han sido reentrenados los modelos de red. La estructura de carpetas en la que las imágenes están distribuidas puede verse en la Figura 4.1(a). Cada una de las carpetas representa una de las clases de alimentos con las que los modelos de red son entrenados para clasificar, conteniendo un total de 30 imágenes cada una. Además de esta estructura de carpetas, la carpeta "Dataset" incluye un fichero llamado *info_nutricional.csv* en el que se incluye la información nutricional (grasas, proteínas, carbohidratos y calorías) de cada una de las clases representadas en la estructura de carpetas mencionada anteriormente.

B. Carpeta "Codigo"

En esta carpeta se incluyen los programas escritos en Matlab, que son los siguientes:

a) *CamaraClasificacionAlimentos.m* que realiza las siguientes operaciones:

- Captura una imagen a través de la cámara del dispositivo móvil de forma manual.
- Redimensiona la imagen capturada al tamaño requerido por el modelo de red del que se trate (227x227 en el caso de AlexNet y 224x224 en el caso de GoogLeNet).
- Clasifica la imagen capturada mediante el método `classify`.

- Muestra la clasificación de la imagen por pantalla.

b) *CamaraClasificacionAlimentosTS.m* que realiza el mismo proceso que el programa anterior, pero recibe los datos nutricionales almacenados en ThingSpeak y devuelve el resultado de la clasificación de la imagen junto con sus datos nutricionales correspondientes.

c) *CargaDatosThingSpeak.m* que lleva a cabo la carga de los datos nutricionales de las distintas clases de alimentos en ThingSpeak mediante las siguientes operaciones:
   - Definición de las claves de acceso del canal de ThingSpeak, tanto para la escritura como para la lectura.
   - Carga de los datos nutricionales mediante múltiples ejecuciones de la función `thingSpeakWrite`.
   - Lectura de los datos cargados.

d) *CNNClasificacionAlimentos.m* que contiene una prueba de clasificación en la que, tras cargar en el *workspace* el modelo de red y elegir una imagen a clasificar, muestra por la ventana de comandos el resultado de la clasificación y los porcentajes de pertenencia a cada una de las clases de alimentos.

e) *CNNTrainingAlimentos.m* que lleva a cabo el entrenamiento de los modelos de red mediante las siguientes funcionalidades:
   - Carga del sistema de carpetas que contienen las imágenes para el entrenamiento.
   - División del conjunto de imágenes en conjunto de entrenamiento y conjunto de validación con una relación del 70%.
   - Sustitución de las últimas capas del modelo manteniendo las demás capas para el traspaso de conocimiento.
   - Modificación de las imágenes de entrenamiento llevando a cabo aumentos, redimensionados y volteados de las imágenes para evitar un sobreajuste del modelo al conjunto de entrenamiento.
   - Entrenamiento del modelo de red mediante la función `trainNetwork`.
   - Almacenamiento del modelo de red entrenado en una variable para su posterior uso.

- Clasificación del conjunto de validación mostrando cuatro imágenes clasificadas juntos con su clase predicha.
- Mostrado de los pesos de la primera capa convolucional.

f) *InicioAlimentos.m* que crea los objetos para el manejo de la cámara del dispositivo móvil, y carga el modelo de red reentrenado en el *workspace*.

g) *LeerDatosThingSpeak.m* que lleva a cabo la lectura de los datos nutricionales almacenados en el canal de ThingSpeak y que fueron previamente cargados mediante el programa *CargaDatosThingSpeak.m*.

h) *netTransferAlimentos.mat* que contiene el modelo de red neuronal reentrenado.

i) *Nutricion.m* que centraliza la ejecución del proceso completo de captura, clasificación y envío de resultados a ThingSpeak mediante las siguientes operaciones:

- Ejecución del fichero *CamaraClasificacionAlimentosTS* para capturar una imagen con la cámara del dispositivo móvil y obtener el resultado de la clasificación junto con su información nutricional.
- Mostrado de la imagen junto con el resultado de la clasificación y su información nutricional por pantalla.
- Lectura de las calorías acumuladas almacenadas en el canal de ThingSpeak.
- Aumento de dichas calorías acumuladas con las calorías correspondientes a la clase en la que se ha clasificado la imagen capturada.
- Actualización del valor de las calorías acumuladas del canal de ThingSpeak.
- Activación del campo de actualización de calorías acumuladas en ThingSpeak.

C. Proceso de ejecución

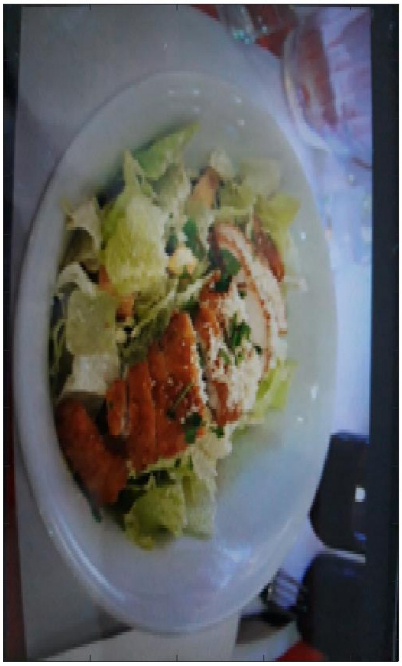
Para llevar a cabo la ejecución del sistema implementado se han de seguir los siguientes pasos:

- Subir a Drive los programas *InicioAlimentos.m*, *CamaraClasificacionAlimentosTS.m*, *LeerDatosThingSpeak.m* y *Nutricion.m*, de forma que estos puedas accederse desde el dispositivo móvil.
- Ejecutar el programa *CargaDatosThingSpeak.m* para cargar los datos nutricionales de las distintas clases de alimentos en el canal de ThingSpeak.
- Abrir la aplicación de Matlab Mobile y conceder permiso de acceso a la cámara en el menú Sensores > Registro de sensores > Configurar.
- Desde la línea de comandos de Matlab Mobile ejecutar el fichero *InicioAlimentos.m*.
- Ejecutar el fichero *LeerDatosThingSpeak.m* para cargar en el *workspace* los datos previamente almacenados en ThingSpeak.
- Ejecutar el fichero *Nutricion.m* el cual activará la cámara para capturar una imagen de un plato de comida para clasificarla.
- Una vez capturada la imagen, en la línea de comandos aparecerá la imagen del resultado de la clasificación junto con su información nutricional.
- Acceder a ThingSpeak mediante la página web o mediante ThingView para ver la actualización de las calorías acumuladas.
- En el caso de haber superado las 2.000 calorías acumuladas visualizar el tweet correspondiente.

APÉNDICE II: EJEMPLOS DE CLASIFICACIONES DE ALIMENTOS

Las siguientes imágenes clasificadas por la aplicación indican los alimentos identificados, junto con sus valores nutricionales.

ensalada_cesar G = 2.1 P = 3.2 Ch = 4.3 Ca = 44



churros G = 20 P = 4.6 Ch = 40 Ca = 360

helado G = 10.1 P = 3.6 Ch = 26.7 Ca = 213

flan G = 3.4 P = 4.8 Ch = 20.4 Ca = 133

espaguetis_boloneSa G = 12.59 P = 12.7 Ch = 26.3 Ca = 123

sushi G = 1.28 P = 6.88 Ch = 18.42 Ca = 103

tarta_chocolate G = 279.42 P = 5.2 Ch = 41.77 Ca = 440

tostada G = 3.7 P = 7.3 Ch = 45.7 Ca = 249