Clasificación de alimentos a partir de imágenes con dispositivos móviles

Bernabé Sánchez Sos[1] , Jorge Azorín López[2] y Andrés Fuster Guilló[2]

Resumen— **El presente artículo científico aborda la creación de un clasificador de imágenes, capaz de clasificar hasta 20 platos distintos y su incorporación a una aplicación móvil centrada en la gestión dietética y nutricional.**

Palabras clave— **Deep Learning, Visión por Computador, Clasificación de imágenes, Aplicación móvil.**

I. Introducción

En los últimos años, el avance de la Inteligencia Artificial y la Visión por Computador ha revolucionado numerosos campos de estudio, incluyendo la clasificación de objetos y reconocimiento de imágenes. En particular, la aplicación de técnicas de Aprendizaje Profundo (Deep Learning) ha demostrado un gran potencial en la resolución de tareas complejas relacionadas con la interpretación y comprensión de imágenes. Un campo en el cual el Aprendizaje Profundo ha encontrado una aplicación prometedora es la clasificación de platos en el ámbito de la nutrición.

La clasificación de platos es una tarea desafiante debido a la amplia variedad de formas, colores, texturas y presentaciones que pueden presentar los alimentos. Sin embargo, gracias a los avances en la tecnología y la disponibilidad de grandes conjuntos de datos etiquetados, los sistemas de Visión por Computador basados en Deep Learning han logrado abrirse camino en esta área.

El objetivo de este artículo científico es explorar y analizar diferentes enfoques y técnicas utilizadas para la clasificación de platos utilizando Deep Learning y Visión por Computador. Se compararán y seleccionarán modelos y arquitecturas de redes neuronales profundas ya existentes para abordar este tipo de problemas de clasificación de imágenes. Además, se mostrará la transformación e incorporación del modelo de red diseñado a una aplicación para dispositivos móviles, para facilitar su uso y acceso a los usuarios.

II. Estado del Arte

A. Food-101

El dataset Food-101 [1] es uno de los datasets más utilizados actualmente para la clasificación de platos alimenticios y, consiste en un conjunto de datos de 101 categorías de alimentos, que almacena y emplea 101.000 imágenes etiquetadas con el nombre del plato, 1.000 por plato. Las imágenes están compuestas por unidades de información digital conocidas como píxeles, formadas a su vez por 3 valores entre 0 y 255 (2^8 bits) que representan el modelo de color RGB. El ancho y alto varía, pero tienen en común que ninguno de los dos valores sobrepasa los 512 píxeles.

B. Red Neuronal Convolucional (CNN)

Una CNN es un tipo red neuronal profunda que se centra en la clasificación de información almacenada en imágenes sin tener en cuenta características espaciales. Verbi gracia, en la resolución del problema de clasificación de platos, la red no se centra en localizar la posición en la que se encuentra un determinado plato, sino que detecta el plato independientemente de la posición en la que se encuentre y, lo clasifica.

Como su nombre indica, la operación principal de este tipo de red es la convolución de matrices, un proceso que permite reducir la cantidad de parámetros que necesita aprender la red y resaltar y extraer características descriptivas de las imágenes [2]. Supongamos que el formato de entrada al modelo de red es de 6x6x3, es decir, 6 píxeles de ancho, 6 píxeles de alto y una profundidad de píxel de 3 (RGB). Sobre cada una de las imágenes se va a aplicar un kernel de 3x3x3 donde cada plano de este convolucionará con su plano de la imagen correspondiente, que dependiendo de los valores que contenga, suavizará la imagen, detectará bordes, realzará la imagen o eliminará ruido presente en la imagen.

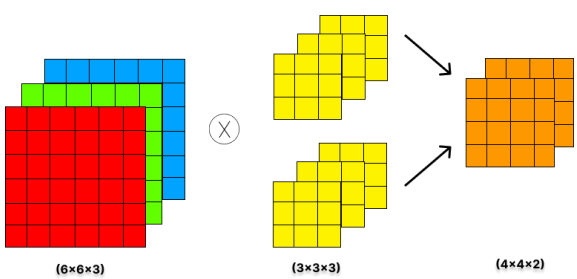


Fig. 1: Operación de convolución

En el caso de la figura 1, se puede apreciar de manera visual como se produce la operación de convolución de una imagen RGB y, como tras aplicar dos kernel 3x3x3, se reduce el número de parámetros de 108 (6x6x3) a 32 (4x4x2). El número de kernels es arbitrario en función de qué características de la imagen se deseen extraer para resolver un problema. Si el número de kernels aumentase, cambiaría la estructura de la información que va a procesar de la red, ya que pasaríamos de un dato con un ancho y alto considerable pero con una profundidad baja (3) a un dato con menor ancho y alto, pero con una profundi-

[1]Ingeniería Multimedia, Universidad de Alicante, e-mail: bss42@gcloud.ua.es.

[2]Dpto. de Tecnología Informática y Computación, Universidad de Alicante, e-mail: {jazorin,fuster}@ua.es.

dad superior que corresponderá al número de kernel aplicados.

B.1 Average Pooling

La operación de agrupación promedio es una técnica utilizada en Deep Learning para reducir las dimensiones de un tensor tridimensional y reducir el sobreajuste durante la fase de entrenamiento [3]. El funcionamiento de este método se basa en agrupar y realizar el promedio matrices en función del hiperparámetro stride. En la figura 2, se realiza la función de agrupación sobre una matriz 4x4x3 utilizando un filtro 2x2 y un stride de valor 2.

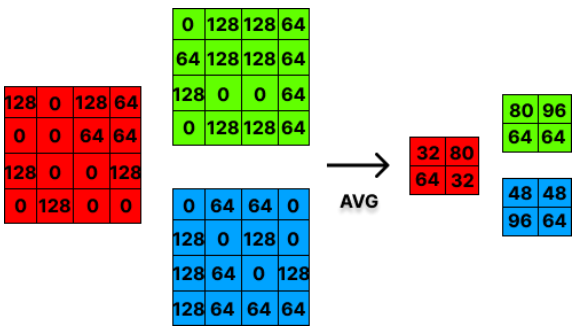


Fig. 2: Agrupación promedio

Este método sumado a las operaciones de convolución de la red acabarán dando como resultado un mapa de características con un ancho y alto con valor 1 y una profundidad que dependerá del número de convoluciones y agrupaciones promedio realizadas. Posteriormente, este resultado será enviado a las últimas capas para poder predecir la clase a la que pertenece la imagen de entrada.

B.2 Dropout

Dropout es un algoritmo relativamente nuevo para entrenar redes neuronales que se basa en abandonar estocásticamente las neuronas durante el entrenamiento, con el fin de evitar la coadaptación en la detección de características [4]. De esta manera, se fuerza a las neuronas a que trabajen de manera independiente, sin tener que depender de la información aportada por las neuronas más próximas. Además, evita el fenómeno conocido como sobreajuste, que hace que la red neuronal se acostumbre a unos datos de entrada determinados sin mejorar durante el entrenamiento o al usar otros datos para entrenamiento, pruebas o predicción.

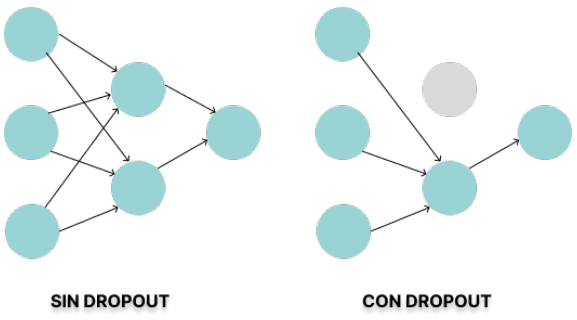


Fig. 3: Dropout

B.3 Dense

La adición de una o varias capas densas a la red hace que mejore el rendimiento si la red no está sobreajustada, ya que intenta reducir la pérdida durante el entrenamiento. Cada capa Dense empleará una función de activación denominada relu, que se añade a las capas ocultas, ésta transforma los valores negativos en cero y es usada para obtener mejores resultados durante la fase de entrenamiento. La última capa de la red también suele ser una capa Dense, conocida como capa totalmente conectada y en los problemas de clasificación en los que intervienen más de dos categorías, se utiliza junto a una función de activación softmax, la cual calcula la probabilidad para cada una de las categorías del problema propuesto [5].

B.4 BatchNormalization

Esta capa de normalización por lotes es capaz de aumentar el rendimiento, haciendo que el tiempo de entrenamiento de la red se reduzca y mejore su precisión. Esto es gracias a la centralización y normalización de cada lote de imágenes a partir de una media y una desviación que se calcula con ese lote, después se vuelve a reescalar y descentrar la información contenida en las imágenes [6].

III. Metodología

A. *Experimentos iniciales*

La primera tarea en el desarrollo de la red neuronal convolucional para clasificar imágenes era encontrar un modelo existente que se pudiese utilizar para resolver el problema. Los modelos de red seleccionados para estudio y comparación fueron: Xception, VGG16, VGG19, ResNet50, InceptionV3 y MobileNetV2. Para ello, se decidió someter a estos modelos a entrenamientos con pocas épocas y datos. Con este objetivo en mente, se escogieron 10.000 imágenes correspondientes a 10 categorías del dataset Food-101. Se realizó un proceso de redimensión de las imágenes a un tamaño de 224x224 con una división de entrenamiento con un 75 % de las imágenes y de validación del 25 %. El tamaño de lote, es decir, el número de imágenes que iba a procesar la red neuronal fue 32. En cada una de estas pruebas no se incluye la última capa del modelo, hay que tener en cuenta que estas arquitecturas de red fueron utilizadas para clasificar 1.000 categorías distintas, por lo que fue necesario cambiar como mínimo la última capa para especificar el nuevo número de categorías a clasificar. Se utilizaron los pesos resultantes de la evaluación de las arquitecturas de red con el conjunto de datos ImageNet y se congelaron todas las capas del modelo, evitando así que pudiesen ser entrenadas.

Con toda la información y técnicas identificas y estudiadas, se diseño un modelo de red con las siguientes características:

1. El conjunto de capas de la red predefinida (Xception, VGG16, ...).
2. Una capa Dropout con rate=0,2, es decir, se desconectan un 20 % de las neuronas.

3. Una capa Dense con 512 unidades y la función de activación relu.
4. Una capa BatchNormalization para normalizar y descentralizar la información de las capas anteriores.
5. Otra capa Dropout con un rate=0,1.
6. Otra capa Dense con 256 unidades junto a la función de activación relu.
7. Otra capa BatchNormalization.
8. Otra capa Dropout con rate=0,1.
9. Una capa Dense con 10 unidades que corresponden al número de categorías posibles junto a la función de activación softmax.

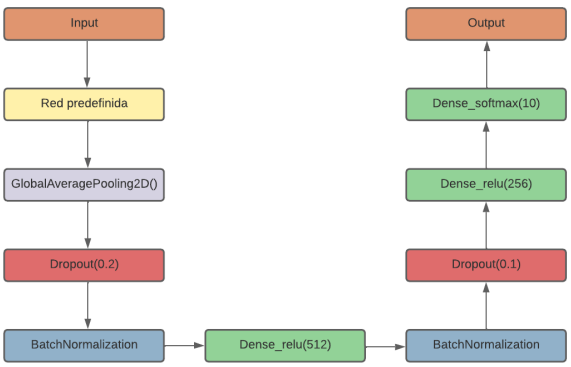


Fig. 4: Arquitectura de red

Cada una de las redes neuronales convolucionales diseñadas a partir de una red ya existente fue entrenada durante 5 épocas, un número relativamente bajo, pero se realizó con el objetivo de descartar rápidamente aquellas arquitecturas de red que no obtuviesen buenos resultados en un corto período de entrenamiento y tiempo, ya que el tiempo medio de entrenamiento de cada red tuvo una duración de 5 minutos aproximadamente. De manera adicional, se utilizó una función de parada temprana que monitorizaba la precisión de validación obtenida durante el entrenamiento con una paciencia de 2, esto se traduce en que si la red no obtenía una precisión de validación superior tras 2 épocas, se detenía el entrenamiento. Además, se incluyó la recuperación de los mejores pesos obtenidos durante la fase de entrenamiento, sin importar la época en la que se alcanzaron.

Un factor determinante para obtener buenos resultados durante el entrenamiento y validación de una red neuronal convolucional, es el uso de un optimizador adecuado en función de las características del conjunto de datos y los parámetros de entrenamiento. En el caso de estos experimentos iniciales, se ha empleado Adam (Adaptative Moment Estimation), el algoritmo de optimización más utilizado en Deep Learning [7], el cual posee las siguientes ventajas:

- Resulta fácil de implementar.
- Es computacionalmente eficiente y requiere de pocos requisitos de memoria.
- Es adecuado para la resolución de problemas compuestos por gran cantidad de datos y/o parámetros.
- Por lo general, no es necesario cambiar los hiperparámetros de la función de optimización porque suelen tener una interpretación intuitiva.

De forma resumida, los parámetros configurados para los entrenamientos corresponden a los de la siguiente tabla:

Tabla I: Parámetros experimentales.

Parámetro	**Valor**
Optimizador	Adam
Tamaño de lote	32
Tasa de aprendizaje	0,001
Número de épocas	5

Los resultados obtenidos tras el entrenamiento de cada red fueron analizados detenidamente con el objetivo de descartar aquellas redes con un aprendizaje lento, estas redes no obtuvieron valores de precisión de validación superiores al 20 %. Estas redes en concreto, son las que utilizaron las arquitecturas de red predefinidas Xception, InceptionV3 y MobilenetV2. Esto no quiere decir que los modelos de red señalados sean poco óptimos para la resolución de problemas de Aprendizaje Profundo, si no que no obtienen buenos resultados con los parámetros y conjuntos de datos propuestos. Por otro lado, las redes neuronales convolucionales desarrolladas a partir de VGG16, VGG19 y ResNet50 obtuvieron mejores resultados.

Tabla II: Resultados de modelos de red.

Red	**Prec. Entr.**	**Prec. Val.**	**Pérd. Val.**
VGG16	93,7 %	75,8 %	1,035
VGG19	95,9 %	74,9 %	0,972
ResNet50	95,8 %	67,1 %	1,571

Los resultados varían en función de la arquitectura de red entrenada. El valor referente a la precisión de entrenamiento indica que no existe una alta diferencia entre los tres modelos de red, esto es debido a las características y parámetros usados. Por otro lado, el valor de precisión de validación sí que presenta diferencias significativas entre las dos redes VGG y ResNet50, llegando a haber una diferencia de un 8,7 % de precisión entre el modelo basado en VGG16 y ResNet50. Respecto a la pérdida de validación ocurre lo mismo, una alta disimilitud entre las arquitecturas de red fundamentadas en cualquiera de las dos versiones VGG y ResNet50. La arquitectura de red elegida para el experimento final de clasificación de imágenes de platos de alimentos fue VGG16, porque a pesar de presentar un número significativamente mayor de parámetros que el modelo justificado en la arquitectura ResNet50, es decir, un mayor coste computacional y temporal de entrenamiento, ha obtenido una precisión de validación notablemente superior. Además, ha obtenido una precisión de validación superior al modelo apoyado en VGG19 que posee un número superior de parámetros.

B. Experimento final

Como ya se ha mencionado anteriormente, la arquitectura de red en la que se fundamentó la red neu-

ronal es en VGG16. En este experimento, el número de categorías es 20 (el doble que en los experimentos anteriores), con el objetivo de aumentar el número de posibilidades y abarcar más tipos de platos. Los platos seleccionados son: apple_pie, caesar_salad, cheesecake, chicken_curry, churros, donuts, escargots, fish_and_chips, french_fries, greek_salad, hamburger, ice_cream, macarons, omelette, paella, pizza, ramen, spring_rolls, sushi y tacos.

Para ello se utilizaron 20.000 imágenes de 20 categorías distintas del dataset Food-101. Las imágenes fueron divididas en tres grupos: entrenamiento, validación y evaluación. El grupo de entrenamiento contenía el 72 % de las imágenes (14.400), el grupo de validación el 18 % (3.600) y el grupo de evaluación un 10 % (2.000). Sobre las imágenes de entrenamiento se aplicó aumento de datos, esta técnica resulta de gran utilidad para conjuntos de datos con pocas muestras o para la simulación de la captura de una imagen en determinadas condiciones. Un caso de uso de esta técnica es el cambio en la rotación de una imagen, ya que es posible que al capturar una imagen de un plato, la cámara este rotada, de esta manera la red se entrena con una mayor cantidad de datos diferentes de una misma categoría, aumentando así la capacidad de predicción de ésta.

Los filtros aplicados para el aumento de datos seleccionados son los siguientes:

- Rotación de la imagen en un rango de -90 y 90 grados.
- Generación de brillo en la imagen entre 0,1 y 0,7.
- Desplazar la imagen hacia la derecha o hacia la izquierda un 50 %.
- Desplazar la imagen hacia arriba o abajo un 50 %.
- Invertir los píxeles de las filas de la imagen.
- Invertir los píxeles de las columnas de la imagen.

En la siguiente figura se puede observar el resultado de aplicar la técnica de aumento de datos con los filtros antes mencionados a una imagen:

Fig. 5: Aumento de datos

El tamaño de las imágenes se estableció en 224x224, con un tamaño de lote de 32 y se mezclaron las imágenes, de forma que el orden cambie para que la red no reciba todas las imágenes de una misma categoría de manera seguida en el conjunto de entrenamiento y validación. Para el desarrollo de este experimento, se parte de los parámetros y resultados obtenidos en otros experimentos que también emplean como base el modelo de red VGG16. Los experimentos en los que se fundamenta la arquitectura de red desarrollada, sus características y parámetros, son los siguientes:

- Experimento 1: en este experimento, se realiza la clasificación de imágenes sobre las 101 categorías de platos del dataset Food-101, con un total de 101.000 imágenes, según una división de entrenamiento y evaluación del 75 % y 25 % respectivamente. Las imágenes fueron redimensionadas a un tamaño de 224x224 píxeles. El tamaño de lote elegido fue de 10, con un valor de 10 épocas durante el entrenamiento. El optimizador escogido para el experimento fue SGDM, con una tasa de aprendizaje del 0,001. Los resultados obtenidos durante este experimento, corresponden a un valor de precisión durante el proceso de evaluación de un 79,86 % [8].
- Experimento 2: en este experimento, se realiza la clasificación de imágenes sobre 10 categorías de platos del dataset Food-101, con un total de 10.000 imágenes según una división de entrenamiento, validación y evaluación del 63,8 %, 1,12 % y 25 % respectivamente. Las imágenes fueron redimensionadas a un tamaño de 224x224 píxeles. El tamaño de lote elegido fue de 64, con un valor de 50 épocas durante el entrenamiento. El optimizador elegido para el experimento fue Adam con una tasa de aprendizaje del 0,001. Además, se aplica el método de Ajuste Fino en el modelo, es decir, se congelan la mayor parte de las capas preentrenadas del modelo VGG16 y solamente se descongelan las últimas capas del modelo, que en este caso fueron 2, para una mejor adaptación del modelo al problema de clasificación de imágenes de platos. Los resultados obtenidos durante este experimento corresponden a un valor de precisión durante el proceso de evaluación de un 81,56 % [9].
- Experimento 3: en este experimento, se realiza la clasificación de imágenes sobre 10 categorías de platos del dataset Food-101, con un total de 5000 imágenes, según una división de entrenamiento y validación del 70 % y 30 % respectivamente. Las imágenes fueron redimensionadas a un tamaño de 224x224 píxeles. El tamaño de lote elegido fue de 64 con un valor de 408 épocas durante el entrenamiento. El optimizador seleccionado para el experimento fue SGDM con una tasa de aprendizaje del 0,001. Los resultados obtenidos durante este experimento corresponden a un valor de precisión durante el proceso de validación de un 85,07 % [10].

En la siguiente tabla, se pueden observar las características principales de cada experimento:

Tabla III: Experimentos de otras fuentes.

Exp.	Optim.	Época	Tam. Lot.	Prec.
1	SGDM	10	10	79,86 %
2	Adam	50	64	81,56 %
3	SGDM	408	64	85,07 %

El conjunto de experimentos formado por experimentos analizados de otras fuentes y los experimentos previos, detallados en el apartado anterior, sirvieron como base para la creación del experimento final para la resolución del problema de clasificación de imágenes. Los parámetros utilizados se obtuvieron a partir de los experimentos anteriores:

- Del primer experimento, se recoge el uso del optimizador SGDM, a diferencia de los experimentos previos en los cuales se había utilizado Adam como función de optimización. SGDM (Sthocastic Gradient Descent with Momentum) es una función de optimización que acelera los vectores de gradientes en las direcciones correctas, lo que se traduce en una convergencia más rápida. El impulso es un promedio móvil de los gradientes que es utilizado para actualizar los pesos de la red. Como resultado, se generan promedios ponderados exponencialmente que facilita la obtención de mejores resultados al proporcionar una estimación alejada de los datos ruidosos durante el entrenamiento [11]. Un estudio que compara las funciones de optimización más utilizadas para la clasificación multicategoría Adam y SGDM, indica que Adam obtiene unos mejores resultados durante el entrenamiento con pocas épocas, pero según éstas se van incrementando, SGDM obtiene mejores resultados de precisión [12].
- Del segundo experimento, se utilizaron los filtros aplicados a la técnica de aumento de datos, ya que el número de filtros era mayor que en los otros dos experimentos. Un número mayor de filtros en la aplicación de aumento de datos favorece la obtención de mejores resultados, pero hay que tener en cuenta que es un proceso costoso computacionalmente que puede alargar la fase de entrenamiento y validación. También se recoge el número de épocas equivalente a 50, en los otros dos experimentos el número de épocas era 10 y 408. Un número de épocas bajo suele ser empleado en conjuntos de datos escasos y con pocas categorías a clasificar, en caso contrario y dependiendo del modelo de red propuesto, los resultados de precisión obtenidos serán bajos comparados con un incremento del número de épocas en el mismo modelo de red. Por otro lado, el incremento del número de épocas es recomendable hasta cierto punto en función de la arquitectura de red, el conjunto de datos y sus características, ya que traspasado ese límite, se estará consumiendo tiempo y recursos sin obtener mejores resultados. Por último, se imita el método de descongelación para las últimas tres capas para adaptar el modelo de red a la resolución del problema de clasificación de imágenes de platos.
- Del tercer experimento, además del uso de la función de optimización SGDM, se recogen las capas de agrupación para reducir la dimensionalidad entre capas y así acelerar el entrenamiento, y la capa dropout para mejorar los resultados del entrenamiento tras cada época.

Tabla IV: Parámetros del experimento final.

Parámetro	Valor
Optimizador	SGDM
Tasa de aprendizaje	0,001
Impulso	0,9
Tamaño de lote	32
Épocas	50
Capas descongeladas	3

Tras determinar todas las características del experimento, se inició el proceso de entrenamiento, validación y evaluación, que posteriormente fueron almacenados y guardados con el fin de analizar y preservar los resultados obtenidos.

IV. Resultados

A. *Resultados experimentales*

La duración total del experimento fue de 4 horas teniendo en cuenta las características del sistema en el que se realizó el experimento, las cuales son:

- Procesador Intel Core i7 2.20 GHz.
- Tarjeta gráfica NVIDIA GeForce RTX 2060.
- Memoria de acceso aleatorio (RAM) de 16 GB.
- Sistema Operativo Windows 10 con arquitectura de 64 bits.

Los resultados obtenidos son los representados a continuación:

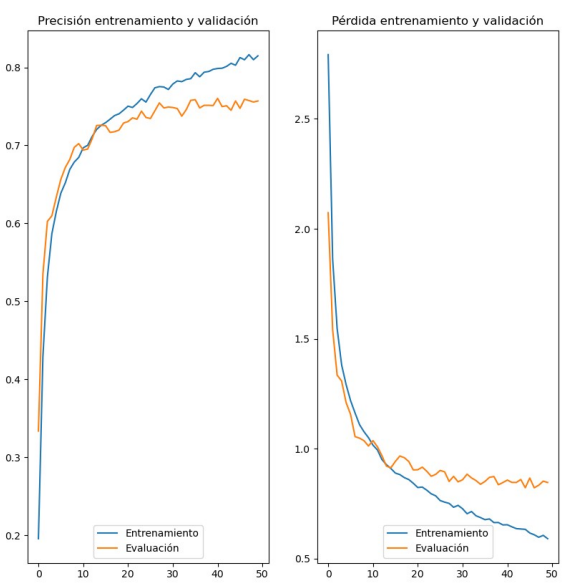


Fig. 6: Resultados del experimento final

Los resultados analizados corresponden al valor de precisión de cada división: entrenamiento, validación

y evaluación. Los resultados se analizan en función de las métricas Top 1 y Top 5. Top 1 es el valor obtenido a partir de la comparación de la respuesta de la red y el valor real. Top 5 corresponde a la comparación de las 5 mejores probabilidades obtenidas y el valor real, si alguna de las categorías cuya probabilidad se encuentra en el Top 5 corresponde al valor real o esperado, se refleja en la precisión Top 5 del modelo.

Tabla V: Resultados Top 1 y Top 5.

Top	Prec. Entr.	Prec. Val.	Prec. Eval.
1	81,63 %	76,02 %	84,38 %
5	97,19 %	94,75 %	97,27 %

Respecto al valor de pérdida, el modelo obtuvo un resultado durante el proceso de validación de 0,8214, un dato situado entre el valor de pérdida del Experimento 2 (0,907) y el Experimento 3 (0,7435).

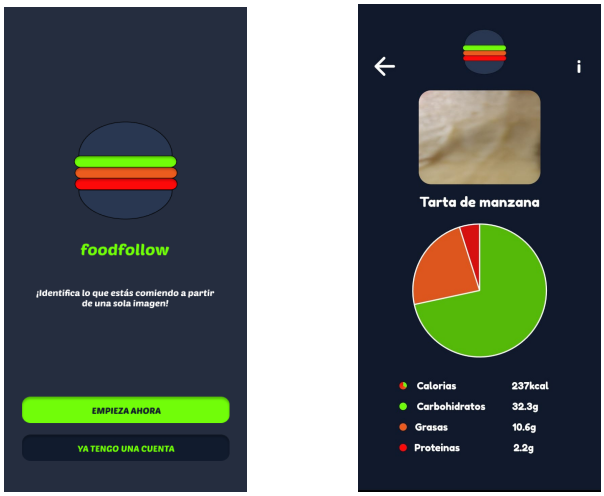
Por último, cabe destacar que se ha obtenido un modelo de red cuyos resultados son capaces de competir con los de los experimentos realizados por otras fuentes. Además, el código generado para realizar el experimento final se encuentra alojado en un repositorio de GitHub para que cualquier persona interesada en la investigación pueda acceder a través del siguiente enlace: `Código del experimento`.

B. Transformación y despliegue de la red

Para que los usuarios pudiesen probar el funcionamiento de la red neuronal convolucional obtenida, se diseñó y creó una aplicación móvil a partir del uso de tecnologías ampliamente conocidas en el desarrollo web. Un paso importante en la implementación de la red en la aplicación, fue el de transformación de formatos.

Para poder utilizar el modelo de predicción, fue preciso convertirlo a un formato compatible con las herramientas que se iban a utilizar. El formato en el que se guardó el modelo desarrollado con el lenguaje de programación Python corresponde a un archivo con extensión .h5, un formato poco amigable con un entorno de desarrollo basado en tecnologías web. El formato elegido para poder utilizar el modelo en la aplicación fue JSON, esta conversión se pudo realizar gracias a la librería tensorlflowjs, que posee una función para convertir un modelo .h5 en un modelo JSON, con las capas que lo componen almacenadas en formato .bin.

De esta forma, el modelo de red convolucional se ejecuta y funciona directamente en el dispositivo móvil del usuario, sin necesidad de un servidor con características hardware elevadas para procesar las imágenes que se capturan y con un tiempo de predicción una vez que el modelo está cargado en memoria de aproximadamente solo 2 segundos. A continuación, se muestran algunas de las interfaces de la aplicación móvil:



(a) Página de Inicio (b) Página de Plato

Fig. 7: Interfaces de la aplicación móvil

V. Conclusiones

En conclusión, este estudio ha proporcionado una visión significativa sobre el uso de redes neuronales convolucionales para la clasificación de imágenes de platos y ha revelado varios hallazgos importantes. A través del análisis exhaustivo de los datos y la aplicación rigurosa de los métodos, se ha llegado a las siguientes conclusiones clave:

1. Las redes neuronales convolucionales son un método excelente para resolver tareas de clasificación de imágenes, aunque resulta complicado dar con el modelo perfecto. También cabe destacar que los resultados de predicción dependen de muchos factores, como funciones de optimización, número de capas, técnicas y/o capas para la mejora del rendimiento y resultados, aumento de datos e incluso, la naturaleza de los datos de entrada.
2. El presente artículo aporta una sólida contribución al problema de clasificación de platos, mostrando distintas técnicas que pueden ser utilizadas para mejorar factores como el rendimiento durante el entrenamiento y la reducción de parámetros de la red para reducir su dimensión. Se espera, que este experimento sirva como base de futuras investigaciones y experimentos, que consigan obtener mejores resultados y una arquitectura de red con cada vez menos parámetros, con el objetivo de que pueda estar presente en el máximo número de sistemas embebidos posibles sin importar sus características hardware.

Agradecimientos

Este trabajo fue apoyado por la Agencia Estatal de Investigación (AEI) de España bajo la subvención PID2020-119144RB-I00 financiada por MCIN/AEI/10.13039/501100011033.

Referencias

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool, "Food-101 – mining discriminative components with random forests," in *European Conference on Computer Vision*, 2014.

[2] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[3] Min Lin, Qiang Chen, and Shuicheng Yan, "Network in network," 2014.

[4] Pierre Baldi and Peter Sadowski, "Understanding dropout," 2013.

[5] Alireza M. Javid, Sandipan Das, Mikael Skoglund, and Saikat Chatterjee, "A relu dense layer to improve the performance of neural networks," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2021-June, pp. 2810–2814, 2021.

[6] Vignesh Thakkar, Suman Tewary, and Chandan Chakraborty, "Batch normalization in convolutional neural networks - a comparative study with cifar-10 data," *Proceedings of 5th International Conference on Emerging Applications of Information Technology, EAIT 2018*, 9 2018.

[7] Diederik P. Kingma and Jimmy Lei Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 12 2014.

[8] Abdulkadir Sengur, Yaman Akbulut, and Umit Budak, "Food image classification with deep features," *2019 International Conference on Artificial Intelligence and Data Processing Symposium, IDAP 2019*, 9 2019.

[9] James McDermott, "Hands-on Transfer Learning with Keras and the VGG16 Model - LearnDataSci," 2021.

[10] Sapna Yadav, Alpana, and Satish Chand, "Automated food image classification using deep learning approach," *2021 7th International Conference on Advanced Computing and Communication Systems, ICACCS 2021*, pp. 542–545, 3 2021.

[11] Sebastian Ruder, "An overview of gradient descent optimization algorithms," 9 2016.

[12] Pujo Hari Saputro, Dhina Puspasari Wijaya, Musthofa Galih Pradana, Dyah Listianing Tyas, and Wahyuni Fithratul Zalmi, "Comparison adam-optimizer and sgdm for classification images of rice leaf disease," *Proceedings - 4th International Conference on Informatics, Multimedia, Cyber and Information System, ICIMCIS 2022*, pp. 348–353, 2022.