ReverseNutrition

Identificación de los ingredientes y la información nutricional de un plato en base a una fotografía

David Álvarez Fidalgo

Directores: Francisco Ortín Soler, José Quiroga Álvarez y Rubén Martínez Aragón

Universidad de Oviedo

Escuela de Ingeniería Informática
Universidad de Oviedo
2022

Agradecimientos

A mi familia, en especial a mis padres, por haberme animado siempre a aprender y apoyado en mis estudios.

A los tutores del trabajo, por haberme guiado durante su realización.

Al grupo de investigación Computational Reflection, por dejarme su hardware para el entrenamiento de los modelos.

Al Observatorio HP, por proponer un tema interesante y desafiante para el proyecto.

Resumen

La mayoría de los procesos existentes para calcular la información nutricional de un plato son complejos y tediosos, y requieren intervención humana. Por tanto, es interesante investigar en el desarrollo de nuevos métodos que permitan realizar esta tarea de forma automática utilizando inteligencia artificial. Sin embargo, los trabajos en esta área son escasos, a pesar del gran desarrollo que han experimentado en los últimos años campos como el aprendizaje automático o la visión artificial. Por ello, el presente trabajo propone dos formas distintas de estimar los valores nutricionales de un plato a partir de una imagen, utilizando aprendizaje profundo: una basada en una arquitectura de red neuronal profunda existente, y otra nueva que añade la capacidad de detectar los ingredientes del plato y estimar sus proporciones. Los resultados obtenidos muestran que estos enfoques son viables utilizando el *dataset* Nutrition5k. Además, la capacidad de detección de ingredientes y la estimación de proporciones ayudan a comprender los razonamientos realizados por estos modelos profundos.

Palabras clave: Nutrición, Ingredientes, Aprendizaje Profundo, Visión Artificial, Inteligencia Artificial

Abstract

Most of the existing processes for calculating the nutrition facts of a dish are complex and tedious, and require human intervention. Therefore, it is important to do research about how to achieve this task automatically by using artificial intelligence methods. Nevertheless, the existing work in this area is pretty limited, despite the great development that fields such as machine learning and computer vision have experienced in recent years. Therefore, this work proposes two different deep learning approaches to estimating the nutrition facts of a dish from an image. The first one is based on existing architectures, while the second one adds the ability to detect dish ingredients and their proportions. The results obtained show that both approaches are feasible using the Nutrition5k dataset. Moreover, the capability to detect ingredients and estimate their proportion help to understand the internals of these deep models

Keywords: Nutrition, Ingredients, Deep Learning, Computer Vision, Artificial Intelligence

Índice general

Capítulo 1

Introducción

La alimentación es una parte fundamental de nuestra vida diaria. La composición de los alimentos que comemos está estrechamente relacionada con nuestra salud y nuestro estado de ánimo [16, 27, 6, 2, 29]. Por esta razón, es muy interesante conocer la información nutricional los alimentos que consumimos. Cuando la comida está envasada, resulta sencillo conocer estos datos, ya que deben aparecer en la etiqueta de su envase. Sin embargo, cuando se trata de platos preparados, esta tarea se complica. Siempre se pueden pesar uno a uno los ingredientes del plato y calcular la información nutricional a partir de la composición de los ingredientes y de sus pesos, pero es un proceso tedioso y difícil, y que resultaría difícil realizar como comensal en un restaurante. También se pueden estimar los pesos de los ingredientes a simple vista, pero estas estimaciones suelen ser propensas a errores [22, 33].

Por estas razones, este trabajo estudia la posibilidad de estimar los valores nutricionales de un plato directamente a partir de una imagen del mismo, utilizando aprendizaje profundo (*deep learning*). De este modo, una vez se haya tomado la imagen, se eliminará la necesidad de que un humano intervenga en el proceso.

1.1 Aprendizaje profundo

El aprendizaje profundo es un área de estudio dentro del aprendizaje automático (*machine learning*), que, a su vez, está contenido en el área más general de la inteligencia artificial. El objetivo del aprendizaje automático es obtener programas, llamados modelos, que aprendan a extraer patrones a partir de datos, de modo que puedan resolver una tarea determinada. Para ello, los modelos almacenan una serie de parámetros internos que se van actualizando de acuerdo con los datos que observan. Generalmente, esto se logra minimizando una función de error, o de pérdida, determinada por la tarea que se desea resolver. Este proceso se conoce como entrenamiento o aprendizaje. Por otro lado, el proceso por el cual el modelo obtenido tras el entrenamiento trata de resolver la tarea a partir de datos nuevos se conoce como inferencia.

El rendimiento de estos modelos depende fuertemente de la representación

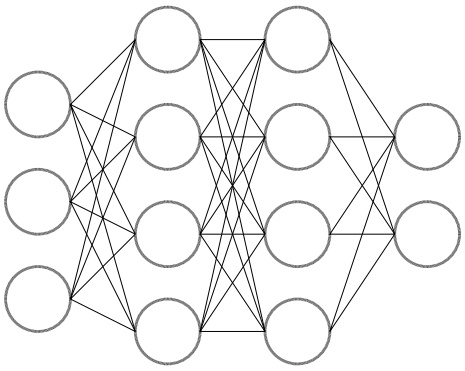

Figura 1.1: Esquema de la estructura de una red neuronal con 3 entradas y 2 salidas. Los círculos representan neuronas y las líneas, las conexiones entre ellas.

de los datos que reciben como entrada [14]. Para eliminar esta dependencia, se diseñan modelos que, además de ser capaces de resolver la tarea para la que fueron concebidos, también sean capaces de aprender la representación ideal de los datos. Esta corriente se conoce como aprendizaje de representación (*representation learning*) [3].

El aprendizaje profundo se caracteriza por aprender múltiples representaciones de los datos, de manera que las representaciones más complejas se construyan a partir de las más simples. Por ejemplo, para detectar un gato en una fotografía, un modelo de aprendizaje profundo partiría de los píxeles de la imagen, construiría los bordes entre los diferentes colores, detectaría las figuras a partir de estos bordes y, finalmente, decidiría si alguna de las figuras se corresponde con un gato. Por este motivo, el aprendizaje profundo está realmente contenido dentro del aprendizaje de representación.

Los modelos más característicos del aprendizaje profundo son las redes neuronales. Estas redes están formadas por capas, que a su vez están compuestas por neuronas. En la Figura 1.1 se muestra un esquema de la estructura de una red neuronal denominada perceptrón multicapa. Las neuronas de una capa están conectadas con todas las de la anterior, a excepción de la capa de entrada. De este modo, cada capa aprende una representación de los datos a partir de la representación aprendida por la capa anterior. El valor de las neuronas de la capa de entrada está determinado por los datos de entrada. El valor del resto de las neuronas es el resultado de sumar los productos de los valores de las neuronas de la capa anterior por una serie de pesos, y aplicar a estos valores una función de activación. En el caso de las redes neuronales, esos pesos son los parámetros que aprende el modelo en el proceso de entrenamiento. Para entrenar estos modelos se utiliza un algoritmo conocido como descenso de gradiente estocástico. Este algoritmo selecciona un conjunto de ejemplos de los datos de entrenamiento y se los pasa al modelo como entrada. A partir de la salida generada por el modelo, se calcula el valor de la función de pérdida y su gradiente respecto a los pesos. Utilizando este gradiente se actualizan los pesos de modo que se reduzca el error. La velocidad de este proceso se controla con un parámetro conocido como tasa de aprendizaje. A mayor tasa de aprendizaje, mayor efecto tendrán las actualizaciones.

1.2 Contribución

Después de esta breve introducción al aprendizaje profundo, vamos a presentar las dos aportaciones principales del trabajo:

a) Implementar un modelo basado en una arquitectura de aprendizaje profundo ya existente para estimar información nutricional a partir de una imagen y analizar el impacto de la selección de diferentes hiperparámetros en este modelo.

b) Presentar una nueva arquitectura basada en *transformers* para resolver el problema de la estimación nutricional, que además genere la lista de ingredientes del plato y estime sus proporciones. Los *transformers* se utilizan habitualmente en tareas de procesamiento de lenguaje natural, pero, como veremos, también consiguen un buen rendimiento en nuestro problema.

En el resto de este documento se fijarán los objetivos del proyecto, se realizará una revisión de los trabajos existentes más relacionados con estos objetivos, se presentarán en detalle las arquitecturas de modelos propuestas y se llevará a cabo un estudio experimental para evaluar su rendimiento usando el *dataset* Nutrition5k [40]. A continuación, se explica con más detalle la motivación y la finalidad del proyecto.

1.3 Motivación

Como se comentó al inicio de esta sección, los métodos para calcular la composición nutricional de un plato que requieren intervención humana no son ideales, sobre todo cuando se trabaja con alimentos ya preparados o cuando se necesita procesar grandes cantidades de platos. Por ello resulta de especial importancia desarrollar nuevos métodos que no necesiten la intervención de seres humanos, usando técnicas como el aprendizaje profundo.

Además, este trabajo también está motivado por la escasez de investigaciones en este campo, a pesar de la gran popularidad de la que ha gozado en los últimos años la Inteligencia Artificial y, en concreto, el aprendizaje automático y el aprendizaje profundo. Se han realizado bastantes trabajos en el área de la comprensión automática de alimentos, como la clasificación de imágenes de comida [23, 28, 25], la detección de ingredientes [4, 44], la generación de recetas [31] o la recuperación de recetas [32, 43, 7, 8, 9]. Sin embargo, casi no existen estudios relacionados con la información nutricional de los platos. De los existentes, la mayoría se centran en estimar solamente las calorías de un plato [19, 30]. El único trabajo relevante que, además de las calorías, estudie la estimación de macronutrientes, como grasas o proteínas, es el de Thames *et al.* [40]. Además, no existe ningún trabajo que combine estimación de calorías, macronutrientes, ingredientes y proporciones de ingredientes.

1.4 Finalidad del proyecto

Este trabajo tiene la finalidad de analizar las arquitecturas de modelos existentes para estimar la información nutricional de un plato a partir de una imagen, y

de proponer una nueva que realice esa estimación detectando los ingredientes del plato y sus proporciones.

Las arquitecturas existentes más relacionadas con nuestros objetivos son las propuestas por Babaeian *et al.* [19] y Thames *et al.* [40]. Nuestra nueva arquitectura mejora la de Babaeian *et al.*, puesto que permite estimar, además de calorías, los macronutrientes del alimento, todo ello usando un modelo mucho más simple que será más rápido en entrenamiento e inferencia. Con respecto a la propuesta por Thames *et al.*, nuestra arquitectura añade la detección de ingredientes y sus proporciones, manteniendo un error similar en las estimaciones de calorías y macronutrientes, lo que ayuda a explicar el razonamiento que sigue el modelo para producir sus predicciones. Además, si ya se conocen los ingredientes del plato, esta arquitectura permite estimar sus proporciones a partir de la imagen del plato y de los ingredientes reales.

Capítulo 2

Fijación de objetivos

El principal objetivo de este trabajo es estimar la información nutricional de un plato a partir de una imagen utilizando aprendizaje profundo. Para ello se diseñará la arquitectura de varios modelos de aprendizaje profundo y se entrenarán en un *dataset* de imágenes de alimentos. Como referencia para evaluar el rendimiento de estos modelos, se tomarán los resultados obtenidos por Thames *et al.* [40].

Otro de los objetivos del trabajo es estudiar un camino no explorado para estimar los datos nutricionales de un alimento, consistente en detectar los ingredientes que lo componen y sus proporciones. De este modo, se pueden conocer de forma más clara las razones por las que el modelo produce una estimación determinada, en comparación con un modelo que estime la información nutricional directamente a partir de la imagen.

Por último, este proyecto también tiene el objetivo de comparar los resultados obtenidos con los distintos modelos, además de analizar el impacto de la selección de hiperparámetros dentro de un mismo modelo.

2.1 Posibles ámbitos de aplicación

Como se comentó en la introducción, este trabajo trata de resolver un problema relevante y que ha sido relativamente poco investigado, por lo que existen un buen número de posibles aplicaciones innovadoras de sus resultados.

Por ejemplo, los resultados de esta investigación podrían ser útiles en el desarrollo de servicios que permitan a los usuarios subir fotos de los alimentos que consumen para ofrecerles estimaciones de su información nutricional.

También podrían ser útiles para empresas que procesen grandes cantidades de imágenes, como Google o Facebook, para que puedan clasificar aquellas en las que aparecen alimentos en base a su información nutricional, sin necesidad de intervención humana. Por ejemplo, un buscador podría aprovechar estos datos para mostrárselos a los usuarios que busquen información acerca de recetas, del mismo modo que muestran su tiempo de preparación y sus reseñas. Un ejemplo de cómo podría implementarse esta idea se muestra en la Figura 2.1.

Figura 2.1: Ejemplo de una posible aplicación de los resultados del trabajo en un buscador para mostrar la información nutricional de una receta.

Los sitios web que permiten publicar reseñas de restaurantes también podrían mostrar información acerca de la comida que se sirve en un restaurante obtenida a partir de las fotos subidas por los usuarios.

Ofrecido como un servicio web, su funcionalidad se podría ofrecer para crear diversos tipos de aplicaciones que permitan obtener información a partir de imágenes de platos.

Capítulo 3

Trabajo relacionado

El aprendizaje profundo y, en concreto, su aplicación en el área de la visión artificial, ha sido objeto de estudio de numerosas investigaciones en los últimos años. Sin embargo, su aplicación a la comprensión automática de imágenes de alimentos no ha sido tan extensa. No obstante, se han llevado a cabo algunos estudios acerca de detección de ingredientes y, en menor medida, estimación de información nutricional. También son importantes los avances en modelos de secuencia a secuencia [42], puesto que la detección de ingredientes y sus proporciones se puede interpretar como un problema de transformación de secuencias. A continuación se analizan las investigaciones en las áreas más relevantes con respecto a los objetivos de este trabajo.

3.1 Visión artificial

El principal hito en la historia de la aplicación del aprendizaje profundo a la visión artificial ha sido la introducción de las Redes Neuronales Convolucionales [14]. Las Redes Neuronales Convolucionales, o CNNs, son un tipo especial de red neuronal que se utiliza habitualmente cuando los datos de entrada tienen una estructura similar a una cuadrícula, donde cada posición está más relacionada con las posiciones cercanas que con las que se encuentran más lejos. Este es el caso de las imágenes, puesto que los píxeles que componen una figura se encuentran concentrados en la misma zona.

Estas redes se caracterizan por usar la operación de convolución en vez de las multiplicaciones de matrices usadas en las redes neuronales tradicionales. La convolución es una operación matemática entres dos funciones que produce como resultado otra función, y se denota por el símbolo $*$. En el caso de las CNNs, esta operación se aplica a tensores (matrices con un número arbitrario de dimensiones), puesto que un tensor se puede considerar como una función de varias variables (tantas como número de dimensiones tenga el tensor) cuyo dominio son los diferentes valores que pueden tomar sus índices. Por ejemplo, una convolución entre una imagen I, representada por una matriz bidimensional, y otra matriz K de dimensiones $m \times n$, compuesta por parámetros aprendidos por el modelo (a la que se suele llamar *kernel*), da como resultado la siguiente

matriz:

$$(I * K)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n) \tag{3.1}$$

Cada posición de esta matriz está conectada con un grupo de posiciones cercanas entre sí en la imagen. Esto contrasta con las redes neuronales tradicionales, en las que cada unidad de una capa está conectada con todas las unidades de la capa anterior.

Las ideas en las que se basan las CNNs modernas fueron introducidas por LeCun *et al.* en 1989 [21]. En los años posteriores se han refinado estos conceptos y han ido apareciendo CNNs que aplican nuevas técnicas para mejorar sus resultados. Algunos ejemplos de estos modelos son ResNet [17], que utiliza conexiones residuales entre sus capas, VGG [35], que utiliza redes muy profundas con *kernels* pequeños, e Inception [38], compuesto por bloques cuya salida es la concatenación de los productos de varias convoluciones con *kernels* de distintos tamaños. A día de hoy, las CNNs siguen siendo la opción predeterminada para crear modelos profundos que reciban como entrada una imagen, como es el caso de este trabajo.

3.2 Modelos de secuencia a secuencia

Los modelos de secuencia a secuencia son aquellos que transforman una secuencia de entrada en una secuencia objetivo, como es el caso de la traducción automática. Este tipo de problemas se ha resuelto tradicionalmente usando Redes Neuronales Recurrentes, o RNNs. Las Redes Neuronales Recurrentes mantienen un estado oculto que se va actualizando recurrentemente según se consumen elementos de la secuencia de entrada. Este enfoque presenta algunos problemas, entre los que se incluyen el coste de entrenamiento que implica la recurrencia y la dificultad de modelar dependencias a largo plazo entre los elementos de la secuencia. Entre las RNNs más utilizadas destaca el LSTM (*Long Short-Term Memory*) [18] y las GRUs (*Gated Recurrent Units*) [10].

Uno de los principales avances en el aprendizaje profundo en general, y en el área de los modelos de secuencia a secuencia en particular, ha sido la introducción de un nuevo modelo, el *transformer*, en 2017 [42]. La estructura del *transformer* está formada por dos componentes: el *transformer encoder* y el *transformer decoder*. El *encoder* transforma la secuencia de entrada y el *decoder* genera la secuencia objetivo de manera autorregresiva a partir de la salida del *encoder*. La principal novedad introducida con este modelo fue eliminar el uso de recurrencia o convoluciones para utilizar únicamente la autoatención, un mecanismo que permite relacionar distintas partes de una secuencia para obtener una representación de la misma [42]. De este modo, se da una solución a los principales problemas de las RNNs mencionados anteriormente.

Entre las aplicaciones del *transformer* destacan aquellas destinadas al procesamiento del lenguaje natural. Este hecho ha dado lugar a la aparición de una generación de modelos conocidos como Grandes Modelos de Lenguaje (*Large Language Models*), que buscan lograr la comprensión automática del lenguaje natural. Entre estos modelos destacan GPT-3 [5] y BERT [12].

El principal objetivo del presente trabajo, identificar la información nutricional de un alimento a partir de una imagen, no es una tarea de lenguaje. Sin embargo, este problema se puede descomponer en la identificación de los ingredientes del alimento y la estimación de las porciones de esos ingredientes. Las listas de ingredientes y sus proporciones se pueden considerar secuencias, siendo así posible resolver nuestro problema usando modelos de secuencia a secuencia.

3.3 Detección de ingredientes

Se han realizado varias investigaciones acerca de cómo detectar los ingredientes que componen un plato a partir de una imagen del mismo. Salvador *et al.* [31] proponen una arquitectura para un modelo capaz de generar recetas a partir de imágenes de platos. Esta arquitectura tiene dos componentes. El primero detecta los ingredientes que aparecen en la imagen y el segundo genera la receta a partir de la lista de ingredientes detectados y la propia imagen. Para este trabajo resulta de interés el primer componente, puesto que la generación de recetas está fuera de nuestros objetivos. La arquitectura de este primer componente está basada en el *transformer* [42]. Como se comentó previamente, el *transformer* se utiliza principalmente para transformar una secuencia de entrada en una secuencia de salida, por ejemplo, en la traducción automática. En el caso de la detección de ingredientes a partir de una imagen, la entrada no es una secuencia, sino que es la propia imagen. Por ello, Salvador *et al.* sustituyen el *transformer encoder* por una Red Neuronal Convolucional (CNN) y usan la salida de esta red como entrada del *transformer decoder*. Esta arquitectura puede servir como base para la detección de ingredientes en nuestro trabajo. Sin embargo, para el cálculo posterior de la información nutricional también es necesario conocer las proporciones de cada ingrediente.

Bolaños *et al.* [4] proponen un enfoque distinto para realizar la detección de ingredientes basándose en la clasificación multiclase. Los problemas de clasificación multiclase se diferencian de los problemas de clasificación tradicionales de visión artificial en que una imagen puede pertenecer a más de una clase. En el caso de la detección de ingredientes, las clases a las que puede pertenecer una imagen son los ingredientes que componen el plato retratado en ella. Para resolver los problemas de clasificación tradicionales se utilizan CNNs. Al final de la CNN se añade una capa densa (cada unidad conectada con todas las unidades de la capa anterior) con una función de activación softmax. El resultado de aplicar esta activación representa la distribución de probabilidad $P(y_i|x)$ de que la imagen pertenezca a cada una de las clases y_i, dada la imagen, x. La predicción $\hat{y}_x$ se obtiene buscado la clase para la que esa probabilidad es máxima, $\hat{y}_x = arg\,max_{y_i} P(y_i|x)$. Para los problemas de clasificación multiclase se sustituye la función de activación softmax por una función sigmoide:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

De este modo, cada unidad de la salida del modelo representa la probabilidad $P(y_i \in \mathbb{Y}|x)$ de que la clase y_i pertenezca al conjunto $\mathbb{Y}$ de clases de la imagen, dada la imagen, x. Así, se puede estimar un conjunto de ingredientes

a partir de una imagen, seleccionando aquellos que superen una cierta probabilidad de estar presentes en el plato de la imagen. Este enfoque también puede resultar interesante para detectar ingredientes en nuestro trabajo en vez de usar un *transformer*, puesto que, como el orden no importa a la hora de detectar ingredientes, dicha lista se puede modelar como un conjunto en vez de como una secuencia. No obstante, como ya se comentó anteriormente, sigue siendo necesario estimar las proporciones de cada ingrediente detectado para poder calcular la información nutricional del plato.

En esta área también es relevante el trabajo de Zhu *et al.* [44], que presentan otros dos métodos para identificar ingredientes:

- El primero consiste en utilizar un clasificador para detectar el ingrediente principal del plato. Después se aplica otro modelo para estima el número de ingredientes de la imagen y se utiliza una matriz de correlación (obtenida a partir del *dataset* usado para el entrenamiento) para determinar los ingredientes con más probabilidad de aparecer junto al principal. Finalmente se aplican clasificadores binarios para cada ingrediente en el orden indicado por la matriz de correlación, hasta detectar el número de ingredientes estimado.
- El segundo método estima primero el número de ingredientes en la imagen. Después se usa *k-means clustering* para dividir la imagen en segmentos de acuerdo con el número de ingredientes estimado. Finalmente, cada segmento se pasa como entrada a un clasificador para determinar el ingrediente al que corresponde.

Estos dos métodos presentan varias carencias que los hacen poco atractivos para nuestro trabajo. El principal problema del primero es que requiere entrenar tantos clasificadores binarios como ingredientes distintos haya en el *dataset*, un número que podría ser muy elevado para *datasets* lo suficientemente variados. Por otro lado, el principal problema del segundo método es que, al estar basado en dividir la imagen en segmentos correspondientes a un ingrediente, no es posible detectar ingredientes que no puedan identificarse con un área concreta de la imagen, como el aceite, la sal o el azúcar. En muchas ocasiones este tipo de ingredientes son los que más contribuyen a la información nutricional del plato.

3.4 Estimación de información nutricional

La estimación de información nutricional a partir de una imagen es el principal objetivo de este trabajo. Sin embargo, existe poca literatura acerca de la aplicación del aprendizaje profundo a esta área. Las investigaciones más relevantes se presentan a continuación.

Babaeian *et al.* [19] presentan una arquitectura completa para estimar las calorías de un plato a partir de una imagen. Esta arquitectura tiene dos componentes principales:

- El primer componente detecta los ingredientes de la imagen usando un *transformer*, de una manera muy similar a la propuesta por Salvador *et al.* [31].

- El segundo componente está a su vez dividido en dos partes:
  - Un *transformer decoder* que estima las calorías correspondientes a cada ingrediente a partir de las características de la imagen y de la lista de ingredientes generada por el primer componente.
  - Tres *transformers decoders* aplicados secuencialmente. Los dos primeros estiman la unidad en la que se debe medir cada ingrediente y la cantidad de esa unidad en el plato, respectivamente. El tercero alinea las calorías estimadas por la primera fase con las predicciones realizadas por los dos primeros *decoders*. Cada uno de estos *transformers* recibe como entrada la imagen, la lista de ingredientes y la salida del *transformer* anterior.

  Los resultados de cada una de las partes se combinan para obtener la estimación final de calorías del plato.

Esta arquitectura es bastante compleja y se podría prescindir de algunos elementos. En lugar de estimar las unidades y la cantidad de cada unidad, se pueden estimar las proporciones de los ingredientes por 100 gramos en el plato. Una vez estimadas las proporciones de los ingredientes, se puede calcular directamente la información nutricional del plato a partir de bases de datos públicas sobre composición de ingredientes, como FoodData Central [41]. De este modo se evita añadir más *transformers* y se pueden estimar todos los datos nutricionales del plato, no solo las calorías.

El otro trabajo relevante en esta área es el realizado por Thames *et al.* [40]. Aunque su aportación principal es la creación del *dataset* Nutrition5k, que se analizará más adelante, también se demuestra su utilidad entrenando un modelo multitarea para estimar información nutricional a partir de las imágenes del *dataset*. Los modelos multitarea, a diferencia de los tradicionales, se entrenan para resolver varias tareas a la vez. Parte de los parámetros del modelo son compartidos por todas las tareas, mientras que el resto son independientes de cada una de ellas. En este caso se considera que las diferentes tareas se corresponden con la estimación de cada uno de los datos nutricionales del plato (calorías, grasas, proteínas e hidratos de carbono). La arquitectura del modelo propuesto está formada por una CNN a la que se añaden al final varias capas densas compartidas por todas las tareas, seguidas de más capas densas independientes para cada tarea. Con esta arquitectura se puede lograr el objetivo principal del trabajo, pero no sirven para la detección de ingredientes y proporciones. No obstante, puede ser una buena referencia para evaluar los resultados de otros modelos.

Capítulo 4

Descripción del sistema

Para lograr el principal objetivo del trabajo, que es estimar la información nutricional a partir de la imagen de un plato, proponemos dos arquitecturas de modelos distintas:

1. Un modelo multitarea basado en la arquitectura presentada por Thames *et al.* [40]. A este modelo nos referiremos como *FCNutr*.
2. Un modelo basado en *transformers* compuesto por una fase de detección de ingredientes y otra de estimación de las porciones de los ingredientes. Nos referiremos a este modelo como *TFIngPort*. Las estimaciones de este modelo se utilizarán para calcular la información nutricional del plato a partir de los datos nutricionales de sus ingredientes.

En el resto de esta sección se presentan en detalle ambas arquitecturas.

4.1 *FCNutr*

FCNutr es un modelo multitarea basado en la arquitectura presentada por Thames *et al.* [40]. Como se comentó anteriormente, los modelos multitarea se entrenan para resolver varias tareas a la vez, de tal manera que parte de los parámetros del modelo sean compartidos por todas las tareas. En nuestro caso, cada tarea se corresponderá con la estimación de un único dato nutricional del alimento, como las calorías o las grasas.

Este modelo está compuesto por una CNN seguida de varias capas densas. Una parte de las capas densas es compartida por todas las tareas, mientras que el resto son independientes para cada una de ellas. En la figura 4.1 se muestra esta arquitectura de manera esquematizada.

Como CNN se ha decidido usar InceptionV3 [39], preentrenada en el *dataset* ImageNet [37], puesto que es una red que consigue buenos resultados en *benchmarks* populares, como el propio ImageNet, manteniendo un número relativamente reducido de parámetros.

La estructura de Inception [38] se diferencia de la de las CNNs tradicionales en que utiliza bloques, llamados módulos Inception, compuestos por convolucio-

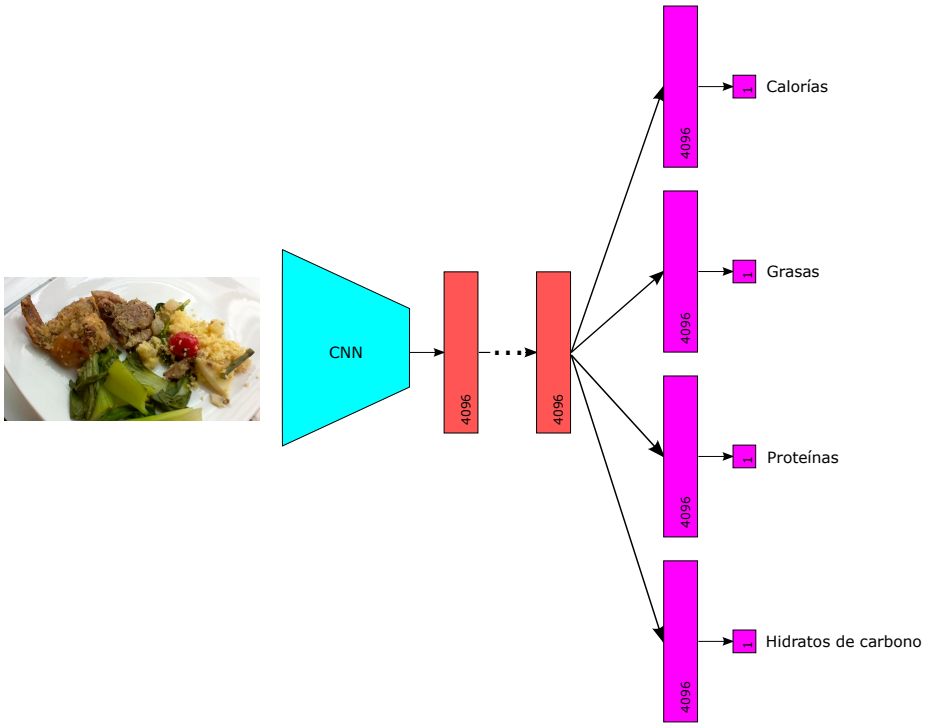

Figura 4.1: Esquema de la arquitectura del modelo *FCNutr*, compuesto por una CNN (en azul), varias capas densas compartidas por todas las tareas (en rojo) y dos capas densas independientes para cada tarea (en fucsia).

nes con *kernels* de distintos tamaños y operaciones de *pooling*, y cuya salida es la concatenación de sus productos. En la Figura 4.2 se muestra la estructura de un ejemplo de módulo Inception. La principal ventaja del uso de estos módulos es que permiten utilizar *kernels* más pequeños, lo que reduce el número de parámetros, manteniendo un rendimiento similar al de otras CNNs. Para el presente trabajo utilizaremos InceptionV3, que es la última revisión de esta arquitectura.

El hecho de utilizar una CNN preentrenada implica que no será necesario que el modelo aprenda algunas de las representaciones de los datos necesarias para resolver nuestra tarea. Esta técnica se conoce como *transfer learning* [45] y consiste en aprovechar los parámetros aprendidos en una tarea determinada para resolver otra distinta. En nuestro caso aprovecharemos los pesos obtenidos tras entrenar la CNN para reconocer objetos en las imágenes de ImageNet y los utilizaremos como punto de partida en nuestro entrenamiento. Como se comentó en la Sección 1.1, el aprendizaje profundo se basa en obtener modelos capaces de aprender varias representaciones de los datos de entrada, de modo que las más complejas se construyan a partir de las más simples. Al usar modelos preentrenados, no es necesario aprender parte de esas representaciones, puesto que el modelo ya las aprendió en el entrenamiento previo. En nuestro caso, por ejemplo, la CNN ya conocerá cómo detectar los contornos de las figuras, reduciendo así el tiempo de entrenamiento. Generalmente se suelen congelar los parámetros transferidos para evitar que se actualicen durante el entrenamien-

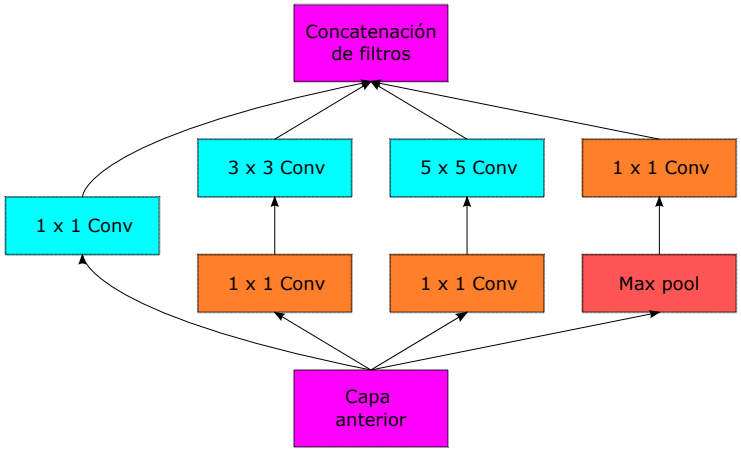

Figura 4.2: Esquema de la arquitectura de un ejemplo de módulo Inception. Las convoluciones que aparecen en naranja se utilizan como un método de reducción de dimensiones.

to. Sin embargo, como nuestro dataset es bastante distinto de ImageNet, no congelaremos estos parámetros.

Como el producto de la CNN es tridimensional ($ancho \times alto \times 2048$) y las capas densas que le siguen necesitan una entrada de una dimensión, se aplica *average pooling* a lo largo de los ejes de anchura y altura. Esto consiste en, por cada posición en el último eje, calcular el valor medio a lo largo de los otros dos ejes. De este modo se obtiene un vector unidimensional de longitud 2048 que puede servir como entrada para las capas densas.

A la CNN le siguen varias capas densas de anchura 4096 compartidas por todas las tareas. Estas capas densas usan una función de activación ReLU (*Rectified Linear Unit*):

$$f(x) = max\{0, x\} \tag{4.1}$$

Se realizarán varios experimentos para determinar el número óptimo de capas compartidas, aumentando progresivamente la capacidad del modelo hasta llegar a obtener errores derivados del sobreajuste.

Después de las capas compartidas se utilizan 2 capas densas independientes para cada tarea. La primera capa tiene una anchura de 4096 unidades y una función de activación ReLU. La segunda capa es la capa de salida de esa tarea, por lo que contiene una única unidad. La última capa también utiliza una función de activación ReLU, puesto que las salidas del modelo deberán ser siempre positivas (indican la cantidad de cada valor nutricional por 100 g del plato).

Para todas las tareas se usará el error absoluto medio (*mean absolute error*, MAE) como función de pérdida. El error absoluto medio entre una predicción $\hat{y}$ y el valor verdadero y, para un tamaño de *batch* N, se calcula de la siguiente manera:

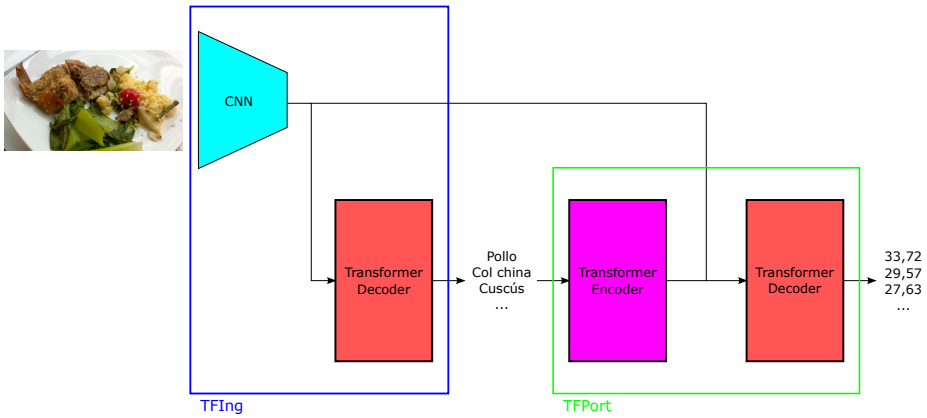

Figura 4.3: Esquema de la arquitectura del modelo *TFIngPort*, compuesto por dos fases: *TFIng* (recuadrada en azul) y *TFPort* (recuadrada en verde).

$$MAE(\hat{y}, y) = \frac{\sum_{i=1}^{N} |\hat{y}_i - y_i|}{N} \tag{4.2}$$

4.2 *TFIngPort*

El segundo modelo, *TFIngPort*, es un modelo basado en *transformers* [42] que está dividido en dos fases:

1. La primera fase recibe como entrada la imagen del alimento y genera una lista de ingredientes ordenados de mayor a menor proporción en el plato. A esta fase nos referiremos como *TFIng*.

2. La segunda fase recibe la lista de ingredientes ordenados producida por *TFIng* y genera una secuencia de proporciones, en la que el elemento n se corresponde con la proporción del ingrediente n de la lista de ingredientes en el alimento. Nos referiremos a esta segunda fase como *TFPort*.

La principal ventaja de este diseño en dos fases es que permite entrenar cada una de ellas de forma independiente. Además, cuando se conozcan los ingredientes del plato pero no sus proporciones, se podría usar únicamente la segunda fase para estimar las proporciones.

Una característica del *transformer* es que utiliza *embeddings* para representar los elementos de las secuencias. Los *embeddings* son representaciones vectoriales de, en este caso, valores escalares. El modelo aprende estas representaciones de tal modo que la distancia entre los *embeddings* de dos elementos similares sea menor que entre los de otros dos menos relacionados.

El uso de *embeddings* es muy habitual en tareas de lenguaje natural, donde cada *token* está identificado por un número entero a partir del cual se genera el *embedding*, de manera que, por ejemplo, la distancia entre los *embeddings* de *tokens* como "perro" y "gato" sea menor que la distancia entre los *embeddings* de esos *tokens* y el generado a partir del *token* "coche". Uno de los principales trabajos de investigación en esta área es Word2vec [26].

En el caso de *TFIngPort*, usaremos *embeddings* de longitud 256 en todas las partes del modelo. También se generarán *embeddings* a partir del orden de las posiciones de la secuencia, que se sumarán a los *embeddings* obtenidos del contenido de esas posiciones, de tal modo que los *embeddings* resultantes contengan también información acerca del orden de los elementos. Estos *embeddings* posicionales también serán aprendidos por el modelo.

Además, cabe destacar que se usará un tamaño de secuencia fijo, rellenando con ceros el resto de la secuencia cuando su tamaño sea menor que esta cantidad fija. En los *transformers* se utilizará una máscara para evitar que los ceros se tengan en cuenta a la hora de producir salidas.

En la Figura 4.3 se muestra un esquema de la arquitectura general del modelo. En las secciones siguientes se detallará la estructura interna de cada una de las fases que lo componen.

4.2.1 *TFIng*

La primera fase del modelo, *TFIng*, está basada en la arquitectura propuesta por Salvador *et al.* [31] para detectar ingredientes. Esta fase estará compuesta por una CNN que recibe la imagen de entrada y un *transformer decoder* que recibe las características de la imagen extraídas por la CNN y genera, de manera autorregresiva, la lista de ingredientes ordenados de mayor a menor proporción en el plato de la imagen.

Salvador *et al.* proponen modelar la lista de ingredientes como un conjunto, puesto que los ingredientes no pueden repetirse y, en principio, su orden no importa. Sin embargo, este enfoque implica que sea necesario entrenar el *decoder* de forma autorregresiva, perdiendo una de las principales ventajas del *transformer*. Además, en nuestro caso, el orden de los ingredientes sí que importa, puesto que, al generar los ingredientes ordenados por proporciones, facilitamos el trabajo de la segunda fase del modelo. Por estas razones hemos decidido modelar los ingredientes como una lista ordenada en vez de un conjunto. En la Figura 4.4 se muestra un esquema más detallado de la arquitectura de esta fase.

En cuanto a la CNN, se ha decidido usar InceptionV3 [39], del mismo modo que para *FCNutr*. En este caso, la salida de la CNN se usará como entrada del *decoder*, por lo que será necesario transformar el producto tridimensional de la CNN en una secuencia cuyos elementos sean vectores de la misma longitud que los *embeddings*, es decir, que sean de longitud 256. Para ello se aplicará una convolución con 256 *kernels* de dimensiones 1×1. El resultado de esta convolución sigue siendo un tensor tridimensional con forma $ancho \times alto \times 256$. Para que sea compatible con la entrada del *decoder*, se reducirán sus dos primeras dimensiones para obtener un tensor con forma $ancho \cdot alto \times 256$.

El *transformer decoder* sigue la arquitectura propuesta por Vaswani *et al.* [42]. La salida del *decoder* es también una secuencia de *embeddings*. Esta secuencia se pasa como entrada a una capa densa con tantas unidades como el tamaño del vocabulario de ingredientes, y con una función de activación softmax. El resultado de esta capa es una secuencia cuyo elemento n es la distribución de probabilidad de que el elemento n de la lista de ingredientes s, $s^{(n)}$, sea de la clase y_i, dada la imagen x y los elementos previos de la lista de ingredientes:

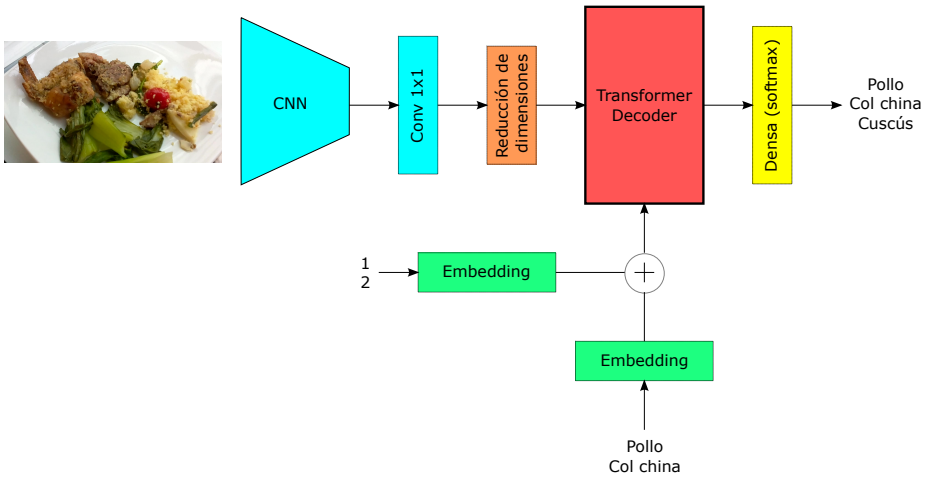

Figura 4.4: Esquema de la arquitectura de la primera fase, *TFIng*, del modelo *TFIngPort*.

$P(s^{(n)} = y_i|x, s^{(1)}, ..., s^{(n-1)})$.

Aunque hayamos decidido modelar la secuencia de ingredientes como una lista en vez de un conjunto, no vamos a permitir que haya elementos repetidos. Para ello, antes de aplicar la función de activación softmax y por cada posición de la secuencia de salida, se sumará $-\infty$ a las unidades correspondientes a los ingredientes que ya aparezcan en las posiciones anteriores. De este modo, el modelo siempre estimará una probabilidad de 0 para los ingredientes que ya hayan aparecido, evitando las repeticiones.

En el entrenamiento, el *decoder* recibirá como entrada las características de la imagen producidas por la CNN y la secuencia completa de ingredientes desplazada una posición a la derecha. En la primera posición de la secuencia, que habrá quedado vacía después del desplazamiento, se insertará un *token* especial que represente el inicio de la secuencia. Al final de la secuencia se añadirá otro *token* especial que represente el fin de la secuencia. Como el *decoder* recibe la secuencia completa, se aplicará una máscara con el objetivo de que, para cada posición de la salida, solo se use la información de las posiciones anteriores a la hora de predecir su clase. De este modo se entrenan todas las posiciones en un solo paso.

Durante la inferencia, el *decoder* recibe inicialmente el *token* de inicio de secuencia y la salida de la CNN. Se toma la predicción del ingrediente de la primera posición y se concatena con el *token* de inicio de secuencia, para volver a pasárselo al *decoder* como entrada. Este proceso se repite con cada una de las posiciones hasta que el *decoder* prediga que el siguiente elemento es el *token* de fin de secuencia. De este modo el *decoder* genera la lista de ingredientes de manera autorregresiva.

Para entrenar el modelo se usará como función de pérdida, para cada posición de la secuencia, la entropía cruzada, H. La entropía cruzada entre la distribución de probabilidad real P (aquella con una probabilidad de 1 para el ingrediente

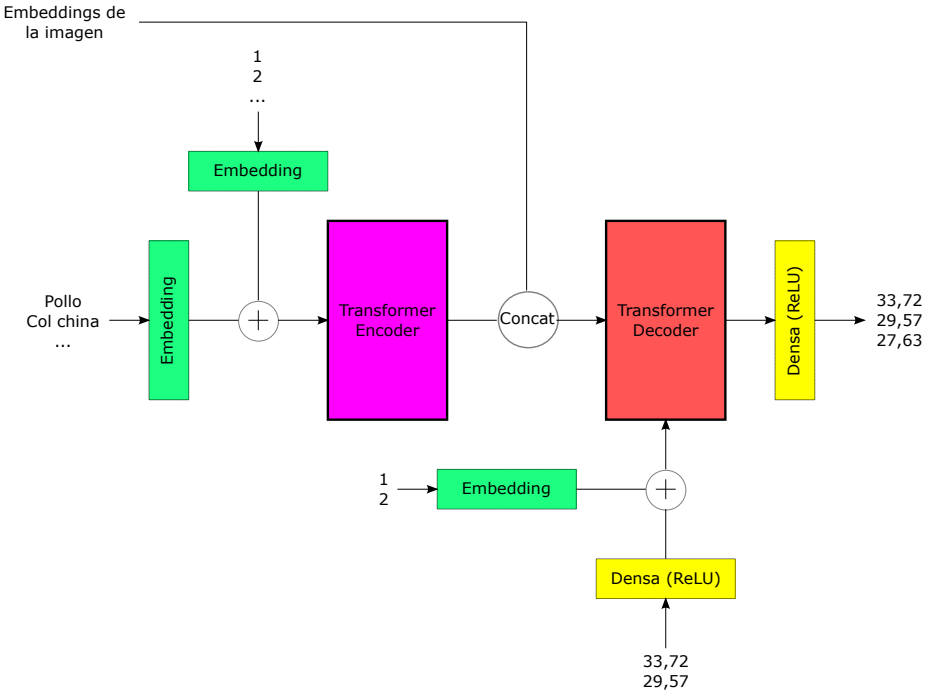

Figura 4.5: Esquema de la arquitectura de la segunda fase, *TFPort*, del modelo *TFIngPort*.

real y 0 para el resto), y la distribución Q estimada por el modelo se calcula de la siguiente manera:

$$H(P,Q) = -\sum_{i} P(y_i) \log Q(y_i) \tag{4.3}$$

El concepto de entropía cruzada procede de la teoría de la información y mide cómo de distintas son dos distribuciones. En nuestro caso, mide cómo de diferentes son la distribución real y la estimada por el modelo, por lo que es un valor que nos interesa minimizar.

4.2.2 *TFPort*

La segunda fase del modelo, *TFPort*, está compuesta por un *transformer encoder* y un *transformer decoder*. El *transformer encoder* recibe como entrada la lista de ingredientes producida por la primera fase y la transforma en una secuencia de *embeddings*. El *decoder* recibe las salidas del *encoder* y los *embeddings* de la imagen generados en la primera fase, y genera la secuencia de proporciones de los ingredientes, de tal manera que el elemento n de la secuencia se corresponda con la proporción por 100 g en el alimento del ingrediente n de la lista de ingredientes producida por *TFIng*. La Figura 4.5 muestra un esquema más detallado de la arquitectura de esta fase.

Tanto la arquitectura del *encoder* como la del *decoder* están basadas en la

propuesta por Vaswani *et al.* [42]. Sin embargo, el *transformer decoder* original está diseñado para resolver problemas de clasificación, ya que se estima la probabilidad de que un elemento de la secuencia objetivo pertenezca a cada una de las clases posibles, como era el caso de la detección de ingredientes. En nuestro caso, el *decoder* se utilizará para una tarea de regresión, ya que se trata de estimar el número real que ocupará cada posición de la secuencia objetivo. Por esta razón se van a realizar algunos cambios respecto a la arquitectura original del *decoder*:

- Primero, se necesita obtener los *embeddings* de los elementos de la secuencia de proporciones que se han generado en un momento determinado y que el *decoder* utiliza para predecir el siguiente elemento de la secuencia. El método tradicional para realizar esta tarea está diseñado para elementos que sean números enteros y utiliza una matriz de *embeddings*. En nuestro caso, los elementos de la secuencia son número reales, por lo que este método no funcionará. Este problema lo resolvemos utilizando una capa densa con 256 unidades y activación ReLU para general los *embeddings*.
- Por otro lado, en lugar de pasar la salida del *decoder* a una capa densa con una función de activación softmax, se pasará a una capa densa con una sola unidad y función de activación ReLU. De este modo, la salida de la fase es una secuencia cuyos elementos son los números reales positivos resultantes de aplicar esta última capa.

Otra diferencia respecto al *transformer* original es que el *decoder* recibe, además de la salida del *encoder*, los *embeddings* de la imagen, a parte de la secuencia objetivo generada hasta ese momento. Salvador *et al.* [31] también se encontraron con este problema a la hora de generar recetas a partir de una imagen y una lista de ingredientes. Como solución, observaron que la mejor opción es concatenar las dos entradas para obtener una única secuencia que servirá como entrada del *decoder*. Nosotros también seguiremos ese enfoque.

Al igual que en *TFIng*, el *decoder* entrena todas las posiciones de la secuencia objetivo en un solo paso y, durante la inferencia, genera la secuencia de proporciones de forma autorregresiva. En este caso no se utilizará ningún *token* especial para marcar el final de la secuencia, sino que, durante la inferencia, se muestrearán tantas proporciones como ingredientes haya en la entrada de la fase.

Como esta fase resuelve un problema de regresión, al igual que en *FCNutr*, se usa el error absoluto medio como función de pérdida, en lugar de la entropía cruzada. La fórmula para calcular el error absoluto medio es la que aparece en la Ecuación 4.2.

Capítulo 5

Metodología

A partir de las arquitecturas presentadas en el Capítulo 4 se van a realizar los siguientes experimentos:

1. Se entrenarán cuatro modelos distintos basados en la arquitectura de *FC-Nutr*, variando el número de capas compartidas (2 o 3) y usando o no *dropout* en esas capas. La técnica de *dropout* [36] consiste en, durante el entrenamiento, poner a cero aleatoriamente algunas de las unidades del modelo para regularizarlo, es decir, para controlar su capacidad y evitar sobreajuste. Por tanto, se probarán las siguientes combinaciones:

   (a) 2 capas compartidas sin *dropout*.

   (b) 2 capas compartidas con *dropout*.

   (c) 3 capas compartidas sin *dropout*.

   (d) 3 capas compartidas con *dropout*.

   Después, se medirá el error absoluto medio entre las predicciones de cada uno de los modelos y los valores nutricionales reales de los platos.

2. Se entrenará un modelo con la arquitectura de *TFIngPort*. Después, se realizarán varias mediciones para evaluar el rendimiento del modelo. Para la predicción de ingredientes se calculará la tasa de acierto media y el coeficiente de Jaccard medio entre el conjunto de las predicciones y el de los ingredientes reales. El coeficiente de Jaccard mide la similitud entre dos conjuntos y se calcula utilizando la siguiente fórmula:

   $$\mathcal{J}(\mathbb{A}, \mathbb{B}) = \frac{|\mathbb{A} \cap \mathbb{B}|}{|\mathbb{A} \cup \mathbb{B}|} \tag{5.1}$$

   De la estimación de porciones se medirá los errores absoluto y cuadrático medios. Finalmente, a partir de las predicciones de ingredientes y proporciones, se calculará la información nutricional del plato y se medirá el error absoluto medio entre los resultados y los valores reales.

Todos los modelos se entrenarán con el conjunto de entrenamiento del *dataset* seleccionado, mientras que sus resultados se medirán en el conjunto de evaluación[1].

A continuación se detalla el *dataset* que se ha seleccionado, cómo se entrenaron los modelos y el entorno de ejecución en el que se han realizado los experimentos.

5.1 *Datasets*

Para entrenar y evaluar los modelos se han considerado dos *datasets* distintos: Recipe1M [32, 24] y Nutrition5k [40]. A continuación, se analiza la composición de cada uno de estos *datasets*, así como sus ventajas e inconvenientes para aplicarlos al problema de estimar la información nutricional de alimentos a partir de imágenes.

5.1.1 Recipe1M

El *dataset* Recipe1M [32, 24] contiene aproximadamente 1 millón de recetas extraídas de páginas web, incluyendo su título, lista de ingredientes e instrucciones de preparación. Además, el *dataset* incluye unas 800.000 imágenes de estas recetas. Algunas de las recetas del *dataset* tienen asociadas más de una imagen, mientras que de otras no se incluye ninguna. De ese millón de recetas, unas 50.000 también incorporan información nutricional: calorías, grasas, grasas saturadas, proteínas, azúcar y sal. Para este trabajo se va a considerar el subconjunto de esas 50.000 recetas para las que, además de información nutricional, se incluyen una o más imágenes. A continuación, se muestra el número exacto de imágenes y recetas que contiene este subconjunto, así como su distribución en las particiones de entrenamiento, validación y evaluación.

- 20.232 recetas:
  - 14.127 recetas para entrenamiento (70 %).
  - 3.003 recetas para validación (15 %).
  - 3.102 recetas para test (15 %).
- 50.718 imágenes:
  - 35.293 imágenes para entrenamiento (69 %).
  - 8.005 imágenes para validación (16 %).
  - 7.420 imágenes para test (15 %).

En la Figura 5.1 se muestran algunos ejemplos de imágenes de este subconjunto, acompañadas por los nombres de las recetas a las que corresponden. Las distribuciones del número de ingredientes por receta y del número de imágenes por receta se muestran en la Figura 5.2. Respecto a la distribución del número de imágenes por receta cabe resaltar que existen numerosos valores atípicos que no se muestran en la Figura 5.2. En la Figura 5.3 se muestra un diagrama de caja de esta distribución en la que se pueden apreciar estos valores. La receta

[1] El número de instancias de cada conjunto se detalla en la Sección 5.1.2.

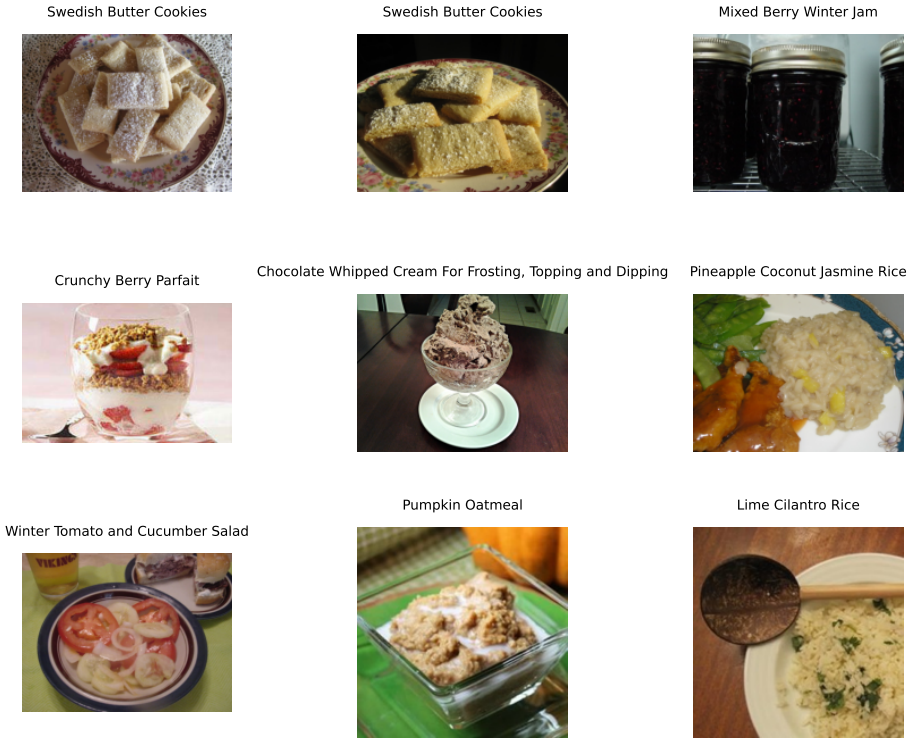

Figura 5.1: Ejemplos de imágenes, junto con los nombres de las recetas a las que corresponden, del subconjunto de Recipe1M.

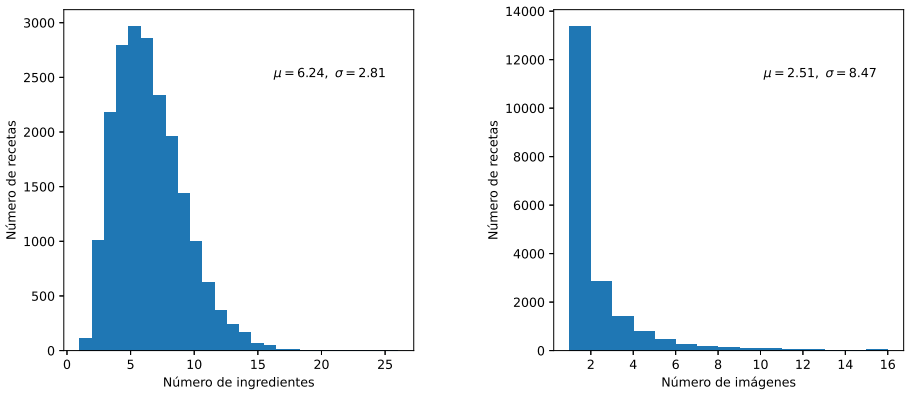

Figura 5.2: Distribuciones del número de ingredientes por receta (izquierda) y del número de imágenes por receta (derecha) en el subconjunto de Recipe1M.

que más imágenes tiene asociadas es “Amish White Bread”, con 506 imágenes. Las imágenes de esta receta representan alrededor del 1 % del total de imágenes del subconjunto seleccionado.

En cuanto a ingredientes, las recetas del subconjunto contienen un total de 355 ingredientes diferentes. En la Figura 5.4 se muestran los 30 ingredientes

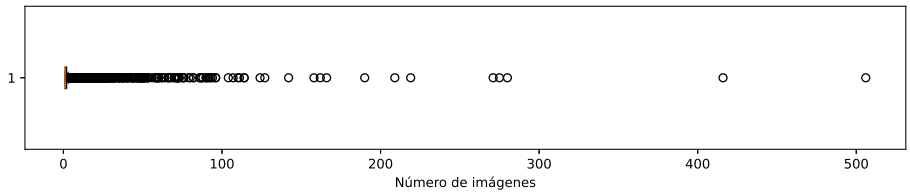

Figura 5.3: Diagrama de caja que muestra la distribución del número de imágenes por receta.

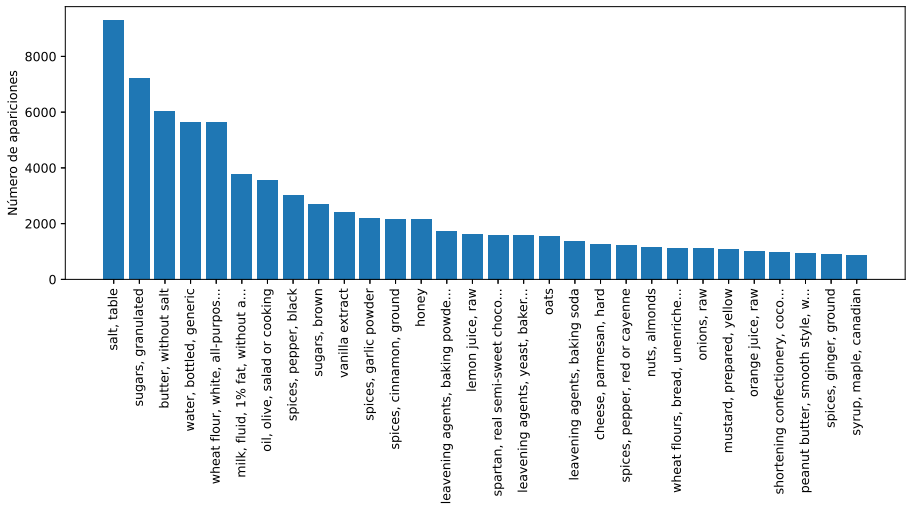

Figura 5.4: 30 ingredientes más populares en el subconjunto de Recipe1M.

más populares junto con su número de apariciones. Todos los ingredientes están identificados por el nombre que tienen asignados en la base de datos FoodData Central [41].

Respecto a la información nutricional, tal como se comentó al comienzo de esta sección, el *dataset* incluye calorías, grasas, grasas saturadas, proteínas, azúcar y sal. Una ausencia notable en esta lista es la de los hidratos de carbono. Para completar los datos nutricionales de las recetas, se realizaron consultas a la base de datos FoodData Central [41] con el objetivo de obtener la cantidad de hidratos de carbono por 100 gramos de cada ingrediente. A continuación, se procedió a multiplicar las proporciones de los ingredientes de cada receta por los resultados obtenidos de la consulta para calcular la cantidad de hidratos de carbono presente en cada plato. En la Figura 5.5 se muestra la distribución de cada uno de los datos nutricionales por 100 gramos entre las recetas del *dataset*.

Este *dataset* tiene algunas ventajas e inconvenientes. Su principal ventaja es su variedad, pues incluye primeros platos, segundos, postres e incluso bebidas. Además, sus imágenes están sacadas en distintos entornos y con diferentes ángulos e iluminaciones, tal como se puede observar en los ejemplos presentados en la Figura 5.1. Por otro lado, al estar obtenidas de sitios web de recetas, es muy probable que algunas imágenes no se correspondan con las recetas reales,

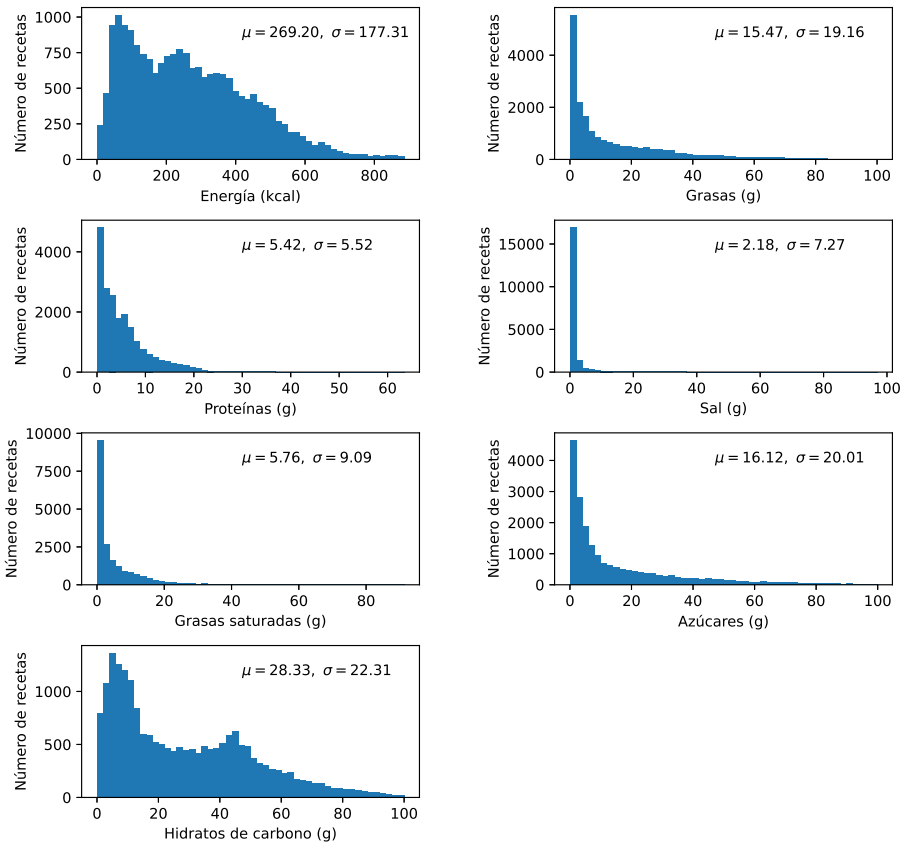

Figura 5.5: Distribución de la información nutricional por 100 gramos en el subconjunto de Recipe1M: energía (primera fila, izquierda), grasas (primera fila, derecha), proteínas (segunda fila, izquierda), sal (segunda fila, derecha), grasas saturadas (tercera fila, izquierda), azúcares (tercera fila, derecha) e hidratos de carbono (cuarta fila, izquierda).

dado que muchos usuarios que publican recetas toman las imágenes de otras fuentes, como buscadores web, en lugar de sacarlas ellos mismos. Además, el procedimiento que se siguió para transformar los ingredientes en lenguaje natural a ingredientes de FoodData Central no garantiza que esta transformación sea completamente correcta [24]. Por último, también se debe considerar que el tamaño del subconjunto seleccionado del *dataset* es relativamente pequeño comparado con otros *datasets* usados habitualmente en visión artificial, como por ejemplo ImageNet, que cuenta con más de 14 millones de imágenes [37]. Existe una versión ampliada de este *dataset* llamada Recipe1M+ que incluye unos 13 millones de imágenes [24]. Sin embargo, al no disponer de una conexión a Internet suficiente rápida como para descargar todas estas imágenes en un tiempo razonable, ni de medios de almacenamiento con el tamaño necesario, se optó por considerar únicamente la versión reducida.

5.1.2 Nutrition5k

El *dataset* Nutrition5k [40] contiene aproximadamente 5.000 platos con ingredientes, proporciones de los ingredientes e información nutricional. De cada plato se incluyen 4 vídeos tomados desde diferentes ángulos con una resolución de 1920×1080 píxeles. El *dataset* también incluye imágenes RGB-D para estimar el peso de los platos, pero no se usarán en este trabajo, puesto que las estimaciones de los valores nutricionales se realizarán siempre por 100 g del plato, sin importar su peso.

Todos estos platos se prepararon en 2 cafeterías distintas de California, donde también se grabaron los vídeos. Para calcular la información nutricional de cada plato se pesaron sus ingredientes y se multiplicaron los pesos medidos por los datos nutricionales del ingrediente publicados en la base de datos FoodData Central [41]. Para este trabajo se van a considerar únicamente los platos preparados en la primera cafetería, puesto que los datos de la segunda contienen mucho ruido, llegando incluso a no verse el plato en una buena parte de los vídeos—Thames *et al.* [40] tampoco usaron los platos de la segunda cafetería en sus experimentos.

También se han excluido todos los platos para los que no había vídeos o para los que la suma de sus nutrientes era mayor que su peso. A continuación, se muestra el número total de platos de este *dataset* que hemos considerado, así como su distribución en las distintas particiones de entrenamiento, validación y evaluación:

- 4.600 platos:
  - 3.237 platos para entrenamiento (70 %).
  - 689 platos para validación (15 %).
  - 674 platos para test (15 %).

Para generar las imágenes que se utilizaron en el entrenamiento de los modelos se extrajo cada quinto fotograma de los vídeos. Después, se revisaron manualmente aquellos platos para los que se obtuvieron más de 200 imágenes y se eliminaron aquellas en las que no se viese el plato o apareciese otro distinto. A continuación, se muestra el número total de imágenes que se obtuvieron y su distribución entre las particiones:

- 252.934 imágenes:
  - 178.436 imágenes para entrenamiento (70 %).
  - 38.152 imágenes para validación (15 %).
  - 36.346 imágenes para test (15 %).

En la Figura 5.6 se muestran algunos ejemplos de imágenes extraídas de los vídeos. Como se puede apreciar en los ejemplos de la primera fila, algunas de las imágenes del *dataset* están volteadas verticalmente. Sin embargo, esto no es un problema, puesto que es deseable que los modelos entrenados sean invariantes a la rotación. En la Figura 5.7 se muestran las distribuciones del número de ingredientes por plato y del número de imágenes por plato. Las cantidades de

Figura 5.6: Ejemplos de imágenes extraídas de los vídeos del *dataset* Nutrition5k.

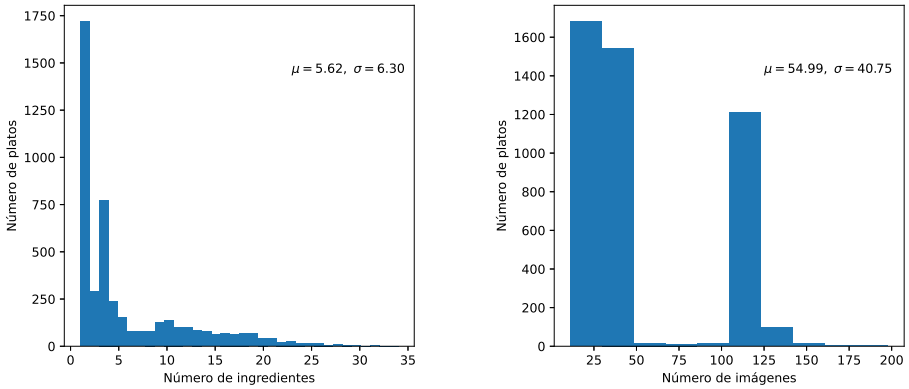

Figura 5.7: Distribuciones del número de ingredientes por plato (izquierda) y del número de imágenes por plato (derecha) en el *dataset* Nutrition5k.

imágenes por plato se concentran en torno a unos valores concretos porque la mayoría de vídeos tiene números de fotogramas similares.

Respecto a los ingredientes, los platos del *dataset* contienen un total de 555 ingredientes distintos. A diferencia del *dataset* Recipe1M, estos ingredientes no están identificados con ningún nombre estándar. Además, se incluyen ingredientes como "pizza" o "ensalada césar" que realmente no son ingredientes propiamente dichos y sería más razonable considerarlos como platos compuestos de otros ingredientes. Los 30 ingredientes más populares del *dataset*, junto con su número de apariciones, se muestran en la Figura 5.8.

En cuanto a la información nutricional, el *dataset* incluye, para cada plato, las calorías, las grasas, las proteínas y los hidratos de carbono. A diferencia de

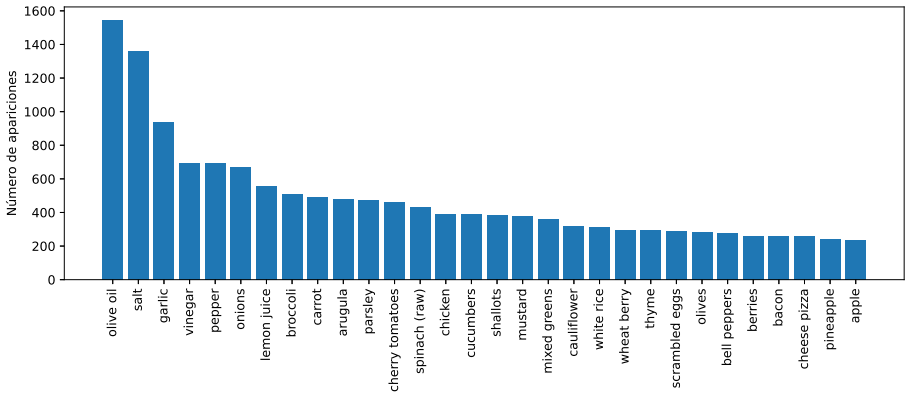

Figura 5.8: 30 ingredientes más populares en el *dataset* Nutrition5k.

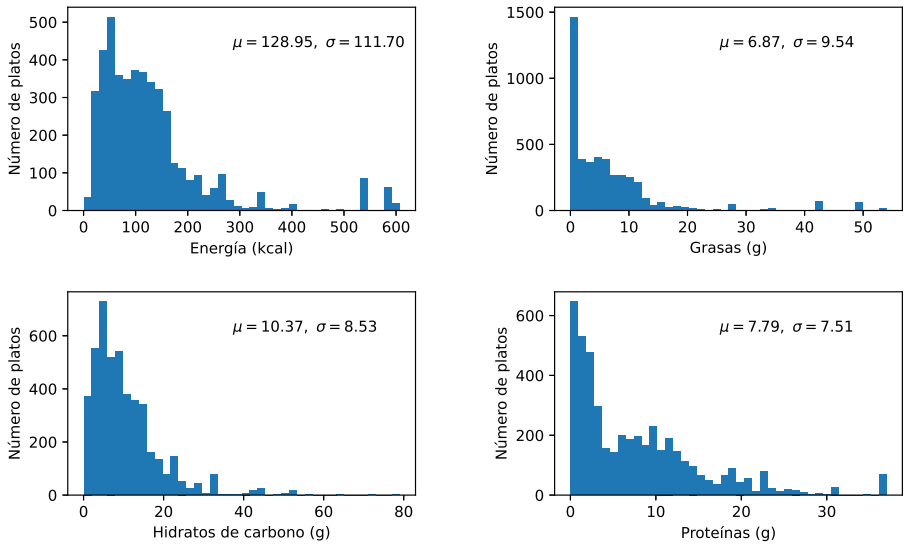

Figura 5.9: Distribución de la información nutricional por 100 gramos en el *dataset* Nutrition5k: energía (arriba, izquierda), grasas (arriba, derecha), hidratos de carbono (abajo, izquierda) y proteínas (abajo, derecha).

Recipe1M, no incluye otros datos relevantes, como las grasas saturadas, el azúcar o la sal. Además, como los ingredientes no están identificados con nombres estándar, no se pueden añadir estos datos de la misma manera que se hizo con los hidratos de carbono para Recipe1M. La distribución de cada uno de los datos nutricionales por 100 gramos en los platos del *dataset* se muestra en la Figura 5.9.

Al igual que Recipe1M, la utilización de este *dataset* presenta algunas ventajas e inconvenientes. Las principales ventajas son que incluye un gran número de imágenes y que el procedimiento seguido para calcular la información nutricional de los platos garantiza que los datos obtenidos representan con gran

fidelidad la composición real de los alimentos. También se garantiza que los vídeos del *dataset* realmente se corresponden con el plato al que están asociados, al contrario de lo que sucedía con Recipe1M.

El mayor inconveniente de este *dataset* es la escasez de su variedad, tanto desde el punto de vista de la composición de los platos, como desde el punto de vista del entorno en el que se grabaron los vídeos. Si se comparan las distribuciones de las Figuras 5.5 y 5.9, se observa fácilmente que las recetas de Recipe1M son más variadas nutricionalmente que los platos de Nutrition5k, a pesar de estar compuestas de un número menor de ingredientes distintos. Por otra parte, comparando los ejemplos de imágenes que se presentan en las Figuras 5.1 y 5.6, se observa que la mayoría de los vídeos de Nutrition5k se grabaron sobre un plato blanco y una mesa blanca, mientras que las imágenes de Recipe1M se tomaron en entornos más variados.

5.1.3 *Dataset* seleccionado

Teniendo en cuenta las ventaja e inconvenientes de ambos *datasets*, se ha decidido utilizar Nutrition5k para entrenar los modelos, puesto que, a pesar de ser menos variado, cuenta con muchos más ejemplos que el subconjunto de Recipe1M con información nutricional y aporta más garantías de que los datos nutricionales que incluye representen fielmente los reales de cada plato.

5.2 Entrenamiento

Para entrenar los modelos se ha utilizado el optimizador Adam [20] con los parámetros $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$ y $\epsilon = 10^{-7}$. Para *FCNutr* y *TFPort* se usó una tasa de aprendizaje de 10^{-4}, mientras que para *TFIng* se redujo a 10^{-5}. Todos los modelos se entrenaron durante un máximo de 80 épocas (*epochs*) y se utilizó *early stopping* con paciencia 10, de tal modo que se parase el entrenamiento después de 10 *epochs* sin mejoras en el error de validación, reteniendo los pesos de la mejor *epoch*. En el caso de *TFIng* se monitorizó la tasa de acierto en el conjunto de validación y se seleccionó la *epoch* que obtuvo el máximo valor para esta medida. En todos los casos se utilizó un tamaño de lote (*batch*) de 32, el máximo posible con el hardware usado.

Para los experimentos 1.b y 1.d se utilizó *dropout* con una tasa de 0,2 antes de la primera capa compartida para retener la mayor parte de las salidas de la CNN. Para el resto de las capas compartidas se utilizó *dropout* con una tasa de 0,5. Para el experimento 2 se utilizó *dropout* con una tasa de 0,1 del modo propuesto por Vaswani *et al.* [42].

En cuanto a los *transformers* del experimento 2, se utilizaron 3 bloques tanto para el *encoder* como para los *decoders*. Al final de cada uno de estos bloques se usó una capa densa con 1024 unidades. Para cada capa de atención se emplearon 8 cabezas. Se utilizó un tamaño de secuencia de 20 (incluyendo el *token* de inicio o de final). Para aquellos platos con 20 o más ingredientes, solo se tuvieron en cuenta los 19 con más peso durante el entrenamiento. Además, se usó un vocabulario de ingredientes de tamaño 558: los 555 ingredientes de Nutrition5k, además de los *tokens* de inicio, final y de relleno. Se utilizó está configuración

porque fue la que permitió obtener los mejores resultados en validación, sin sobrepasar la cantidad de memoria de video disponible.

También cabe destacar que las dos fases de *TFIngPort* se entrenaron independientemente para aprovechar al máximo los recursos hardware disponibles. Primero se entrenó *TFIng* y se guardaron los pesos aprendidos por la CNN. Después se entrenó *TFPort* cargando los pesos de la CNN y congelando todas sus capas. Para entrenar la segunda fase solo se utilizaron los ejemplos del conjunto de entrenamiento del *dataset* con más de un ingrediente, puesto que no es necesario estimar las proporciones en los platos con un único ingrediente. Además, se utilizó la técnica de *teacher forcing* [15], es decir, *TFPort* se entrenó con los ingredientes reales en vez de usar las predicciones de la primera fase. Durante la evaluación se empleó el modelo completo.

Para generar las entradas de los modelos se redujo el tamaño de las imágenes a 256 píxeles en el eje más pequeño, manteniendo su relación de aspecto. Además, se ha utilizado la técnica de aumento del *dataset* (*dataset augmentation*) [34]. Esta técnica consiste en realizar modificaciones aleatorias sobre las imágenes del *dataset* para obtener nuevas imágenes. En nuestro caso, se recortaron las imágenes aleatoriamente a un tamaño de 224×224 píxeles, se voltearon vertical y horizontalmente de forma aleatoria y se rotaron $0{,}2\pi$ radianes, también aleatoriamente. En validación y evaluación solamente se recortó la parte central de la imagen a un tamaño de 224×224 píxeles.

5.3 Entorno de ejecución

Para implementar los modelos se ha usado el lenguaje de programación Python y las bibliotecas TensorFlow [1] y Keras [11]. Se han empleado las siguientes versiones de estas herramientas:

- **Python:** 3.7.9.
- **TensorFlow:** 2.9.0.
- **Keras:** versión incluida con TensorFlow.
- **CUDA:** 11.2.

Todos los experimentos se han realizado en un servidor con la siguiente configuración:

- **Tarjeta gráfica (GPU):** NVIDIA GeForce RTX 2060.
- **Memoria de GPU dedicada:** 6 GB.
- **Procesador (CPU):** 2 x Intel Xeon E5-2630 v3 @ 2.40 GHz.
- **Memoria RAM:** 32 GB.
- **Sistema Operativo:** Windows Server 2019 Datacenter.

Capítulo 6

Resultados obtenidos

El entrenamiento se completó después de aproximadamente 1 día para cada versión de *FCNutr*, 12 horas para *TFIng* y 22 horas para *TFPort*. A continuación, se muestran los resultados generales obtenidos por todos los modelos, los resultados concretos de las dos fases de *TFIngPort* y algunos ejemplos de predicciones.

En la Tabla 6.1 se muestra, para cada modelo entrenado, el error absoluto medio al estimar la información nutricional por 100 g de los platos del conjunto de evaluación de Nutrition5k. También se muestra el error producido al estimar la información nutricional utilizando siempre la media de cada valor en el conjunto de entrenamiento del *dataset*. En la Tabla 6.2 se muestra este mismo error, pero expresado como el porcentaje de la media en el campo correspondiente.

En la Tabla 6.3 aparece la tasa media de aciertos en la detección de ingredientes usando la fase *TFIng* del modelo *TFIngPort* en el conjunto de evaluación de Nutrition5k. La tasa de aciertos se ha calculado comparando cada posición de la lista de ingredientes real con la correspondiente de la estimada, ignorando el *token* de final de secuencia. Todos los ingredientes presentes en la lista real, pero no en la estimada, y aquellos que aparecían en la estimada, pero no en la real, se han contado como fallos. En la Tabla 6.3 también aparece el coeficiente de Jaccard medio entre el conjunto de ingredientes real de cada plato y el estimado. El coeficiente de Jaccard es siempre un valor entre 0 y 1. Además, se muestra la tasa de aciertos media y el coeficiente de Jaccard medio obtenidos al predecir la lista seleccionando un número aleatorio de ingredientes. Para seleccionar el ingrediente de cada posición de la lista se ha elegido uno aleatoriamente, teniendo en cuenta su frecuencia en esa posición en el conjunto de entrenamiento. Para elegir la longitud de la lista también se ha tenido en cuenta la distribución del número de ingredientes por plato en el conjunto de entrenamiento.

En la Figura 6.1 se muestra la variación de la tasa de aciertos media y del coeficiente de Jaccard medio según el número de ingredientes del plato. En la Figura 6.2 se muestra la variación de la tasa de aciertos media en función de la posición del ingrediente.

En la Tabla 6.4 se muestran el error absoluto y cuadrático medios entre

Tabla 6.1: Error absoluto medio al estimar cada valor nutricional por 100 g de los platos del conjunto de evaluación de Nutrition5k, utilizando cada uno de los modelos entrenados. La línea base es el error producido al estimar cada valor utilizando siempre su media en el conjunto de entrenamiento. El error en la estimación del valor energético está medido en kilocalorías, mientras que el resto de errores están medidos en gramos.

			Valor energético	Grasas	Proteínas	H. de carbono
	Línea base		79,19	6,69	6,40	6,14
	Capas compartidas	Dropout				
FCNutr	2	No	19,45	1,61	1,65	2,09
	2	Sí	**18,57**	1,63	1,68	2,10
	3	No	19,22	**1,55**	**1,64**	**2,00**
	3	Sí	19,34	1,73	1,73	2,12
	TFIngPort		21,41	1,78	1,81	2,05

Tabla 6.2: Error absoluto medio, expresado como porcentaje del valor medio del campo correspondiente, al estimar la información nutricional por 100 g de los platos del conjunto de evaluación de Nutrition5k, utilizando cada uno de los modelos entrenados. La línea base es el error producido al estimar cada valor utilizando siempre su media en el conjunto de entrenamiento.

			Valor energético	Grasas	Proteínas	H. de carbono
	Línea base		61,41 %	97,35 %	82,18 %	59,20 %
	Capas compartidas	Dropout				
FCNutr	2	No	15,08 %	23,40 %	21,20 %	20,20 %
	2	Sí	**14,40 %**	23,69 %	21,57 %	20,23 %
	3	No	14,90 %	**22,59 %**	**21,03 %**	**19,28 %**
	3	Sí	15,00 %	25,17 %	22,23 %	20,40 %
	TFIngPort		16,60 %	25,96 %	23,19 %	19,80 %

las estimaciones de proporciones producidas por la fase *TFPort* del modelo *TFIngPort* y los valores reales de cada plato con más de un ingrediente del conjunto de evaluación de Nutrition5k. También se muestra el error producido al estimar la proporción de cada ingrediente utilizando siempre la proporción media en el conjunto de entrenamiento de la posición en la que aparece.

En la Figura 6.3 se muestra la variación del error absoluto medio (expresado en porcentaje) obtenido por *TFIng* según el número de ingredientes del plato. En la Figura 6.4 se muestra la variación del error absoluto medio, expresado

Tabla 6.3: Resultados obtenidos por la fase *TFIng* del modelo *TFIngPort* en el conjunto de evaluación de Nutrition5k. Se muestra la tasa de acierto media de los ingredientes y sus posiciones, y el coeficiente de Jaccard medio entre los conjuntos de ingredientes reales y de los estimados. La línea base es la tasa de acierto y el coeficiente de Jaccard medios obtenidos al estimar la lista de ingredientes seleccionándolos aleatoriamente, teniendo en cuenta su distribución en el conjunto de entrenamiento.

	Tasa de acierto	Coeficiente de Jaccard
Línea base	0,85 %	0,02
TFIng	64,52 %	0,72

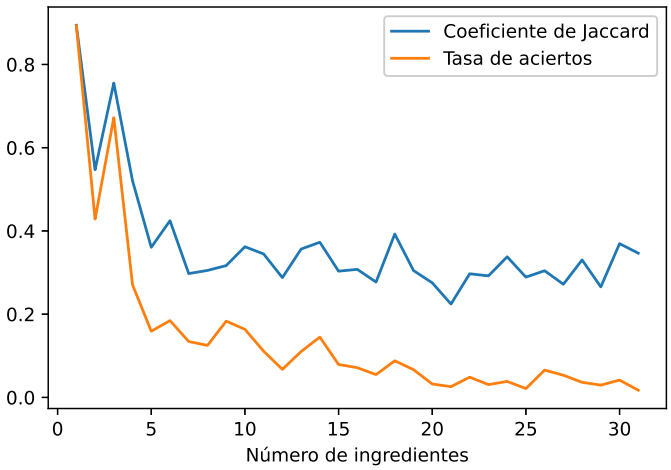

Figura 6.1: Variación de la tasa de aciertos media y del coeficiente de Jaccard medio obtenidos por la detección de ingredientes de *TFIng* según el número de ingredientes del plato, calculados en el conjunto de evaluación de Nutrition5k. La tasa de aciertos está expresada sobre 1.

también en porcentaje, obtenido por *TFIng* según la posición del ingrediente. Todos estos resultados se han calculado a partir de los ingredientes reales de los platos, en vez de usar las estimaciones generadas por *TFIng*. Además, antes de calcular los resultados se han escalado las predicciones (tanto las del modelo como las de la línea base) para que sumen 100.

También se han seleccionado 3 imágenes del conjunto de evaluación de Nutrition5k para mostrar ejemplos de estimaciones realizadas por los modelos. En la Figura 6.5 aparecen las estimaciones de información nutricional usando la versión de *FCNutr* que mejores resultados obtiene (3 capas compartidas, sin *dropout*). En la Figura 6.6 se muestran los resultados de estimar los ingredientes de los platos que aparecen en estas imágenes usando la fase *TFIng* del modelo *TFIngPort*. Asimismo, En la Figura 6.7 se muestran las estimaciones de las proporciones de los ingredientes reales de estos platos usando la fase *TFPort* del

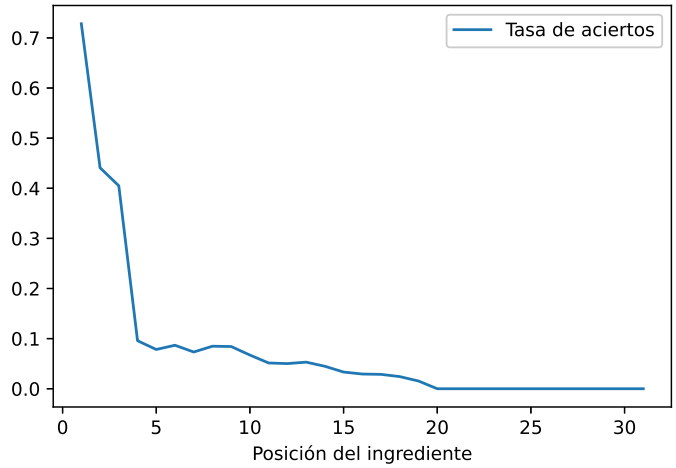

Figura 6.2: Variación de la tasa de aciertos media obtenido por la detección de ingredientes de *TFIng* según la posición del ingrediente, calculada en el conjunto de evaluación de Nutrition5k. La tasa de aciertos está expresada sobre 1.

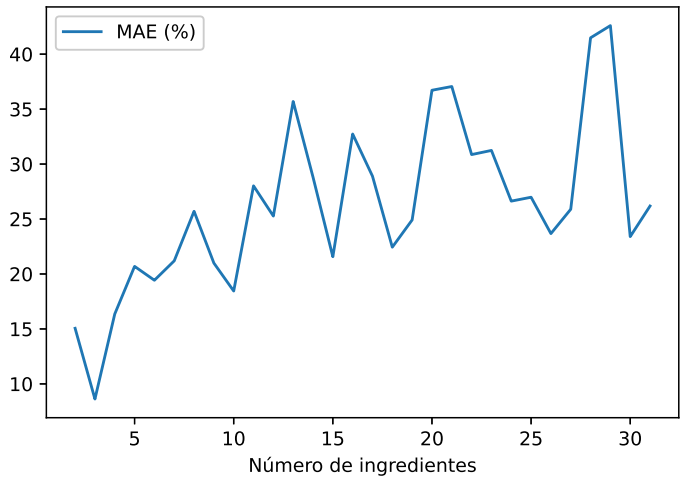

Figura 6.3: Variación del error absoluto medio (expresado en porcentaje) obtenido por *TFIng* según el número de ingredientes del plato, calculado en el conjunto de evaluación de Nutrition5k.

modelo *TFIngPort*. En el Apéndice B se muestran más ejemplos completos de las predicciones realizadas por los modelos.

En el resto de esta sección se analizarán y discutirán los resultados obtenidos. Además, se expondrán sus principales limitaciones.

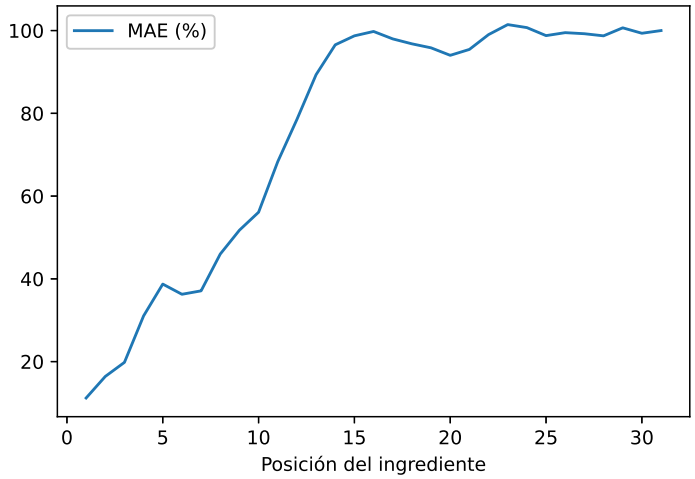

Figura 6.4: Variación del error absoluto medio (expresado en porcentaje) obtenido por *TFIng* según la posición del ingrediente, calculado en el conjunto de evaluación de Nutrition5k.

Tabla 6.4: Resultados obtenidos por la fase *TFPort* del modelo *TFIngPort* en los platos con más de un ingrediente del conjunto de evaluación de Nutrition5k. Se muestra el error absoluto medio (MAE) y el error cuadrático medio (MSE) entre las proporciones reales de los ingredientes por 100 g y las estimaciones. La línea base es el error producido al estimar la proporción de cada ingrediente utilizando la proporción media en el conjunto de entrenamiento de la posición en la que aparece. El MAE está medido en gramos y el MSE, en gramos al cuadrado.

	MAE	MSE
Línea base	6,45	78,84
TFPort	3,11	34,40

6.1 Discusión de los resultados

Los resultados generales de los modelos que aparecen en las Tablas 6.1 y 6.2 son peores que los publicados por Thames *et al.* [40], aunque estos no pudieron ser replicados. Sin embargo, mejoran significativamente la línea base, obteniendo errores entre tres y cuatro veces menores que los producidos al estimar utilizando la media. Además, estos resultados muestran que *FCNutr* tiene un rendimiento ligeramente mejor que *TFIngPort*. No obstante, la capacidad añadida de detectar ingredientes y proporciones puede compensar esta pequeña diferencia en el error, dependiendo de la aplicación a la que se destine el modelo. De las diferentes versiones de *FCNutr*, la que mejor resultados obtiene es la de mayor capacidad, con 3 capas compartidas y sin dropout, por lo que es posible que se obtengan mejores resultados si se aumenta más la capacidad del modelo, por

		Energía	Grasas	Proteínas	Hidratos de carbono
	Real	99,94	4,52	3,98	11,22
	Estimación	136,48	8,94	7,32	6,65
	Real	38,39	1,24	2,60	5,47
	Estimación	50,22	1,99	3,49	6,44
	Real	135,96	4,75	20,14	2,44
	Estimación	139,30	8,74	10,47	5,49

Figura 6.5: Ejemplos de resultados de aplicar el modelo *FCNutr* (3 capas compartidas, sin *dropout*) a imágenes del conjunto de evaluación de Nutrition5k. Para cada imagen se muestran los datos nutricionales reales (fila superior) y los estimados por el modelo (fila inferior).

ejemplo, añadiendo más capas compartidas.

Respecto a los resultados de *TFIng* (ver Tabla 6.3), tanto la tasa de acierto como el coeficiente de Jaccard superan con una gran diferencia los de la línea base. Sin embargo, no existen otros trabajos de detección de ingredientes realizados sobre este *dataset*, por lo que no es posible establecer una referencia con la que comparar los resultados.

Por otro lado, el coeficiente de Jaccard medio es mayor que la tasa de aciertos media, lo que indica que, en las ocasiones en las que el modelo no acierta el orden de los ingredientes, sigue siendo capaz de detectar los ingredientes correctos, aunque desordenados. Los resultados mostrados en la Figura 6.1 también corroboran esta hipótesis, ya que, mientras que la tasa de aciertos desciende de manera continua hasta los 30 ingredientes, el coeficiente de Jaccard se estabiliza en torno al 0,3 a partir de los 5 ingredientes. El hecho de que la tasa de aciertos y el coeficiente de Jaccard decrezcan a medida que el número de ingredientes aumenta era de esperar, ya que parece razonable que cuanto menor peso tenga un ingrediente en un plato, más difícil será detectarlo, y cuantos más ingredientes contenga un plato, mayor número de ingredientes con poco peso habrá en ese plato. Esta tendencia se muestra en la Figura 6.2, donde se observa que la tasa de aciertos disminuye según la posición del ingrediente. La tasa de aciertos para ingredientes más allá de la posición 19 siempre es 0 porque el tamaño de secuencia que usa el *transformer decoder* de *TFIng* es de 20, incluyendo el *token* de final de secuencia.

También se observa una gran diferencia entre la tasa de aciertos de los tres

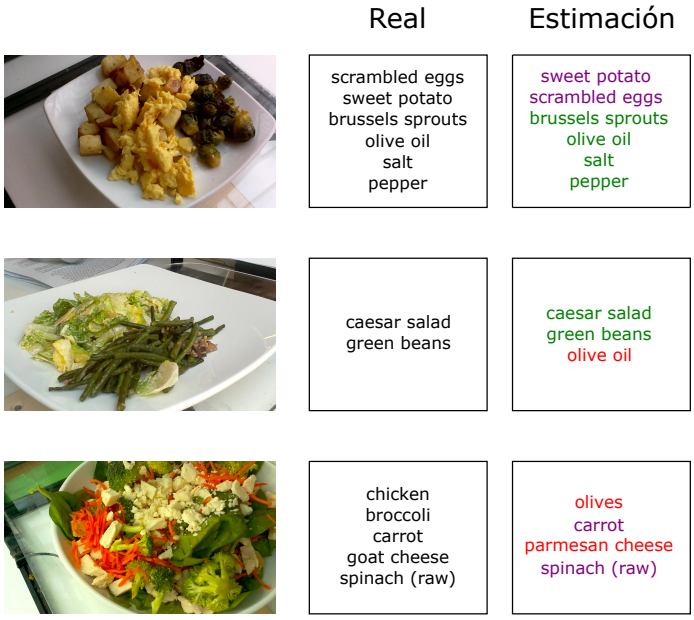

Figura 6.6: Ejemplos de resultados de aplicar la fase *TFIng* del modelo *TFIng-Port* a imágenes del conjunto de evaluación de Nutrition5k. Los ingredientes reales se muestran en la columna central, mientras que los estimados aparecen en la de la derechas. Los ingredientes estimados que aparecen en verde son aquellos que están en el plato real y se ha acertado su posición en la lista de ingredientes. Los que aparecen en morado son los que están en el plato real pero no se ha acertado su posición. Los que aparecen en rojo son los que están en el plato real.

primeros ingredientes y el resto. Esto probablemente se deba a que más de la mitad de los platos del *dataset* tiene menos de cuatro ingredientes, como se puede ver en la Figura 5.7.

Por su parte, los resultados de *TFPort* (ver Tabla 6.4) también mejoran significativamente la línea base, obteniendo menos de la mitad del error producido al estimar usando siempre la proporción media de cada posición. Sin embargo, tampoco existe ninguna referencia con la que compararlos. En la Figura 6.3 se observa que el error crece al aumentar el número de ingredientes del plato, del mismo modo que la tasa de aciertos de *TFIng* disminuía, aunque no lo hace de una manera tan pronunciada. Al igual que con *TFIng*, el error aumenta según la posición del ingrediente en el plato (ver Figura 6.4).

En cuanto a los ejemplos de *TFIng* (ver Figura 6.6), se puede observar que los errores del modelo son razonables. En el primer ejemplo acierta todos los ingredientes, pero cambia de orden "sweet potato" y "scrambled eggs". Además, aparecen ingredientes como "olive oil", "salt" y "pepper", lo que demuestra que es capaz de detectar ingredientes que no se pueden observar a simple vista, sino que los deduce a partir del resto de ingredientes del plato. Algo similar ocurre en el segundo ejemplo, aunque en este caso el modelo no acierta al estimar que el plato lleva aceite de oliva. En el tercer ejemplo no acierta el orden de ninguno

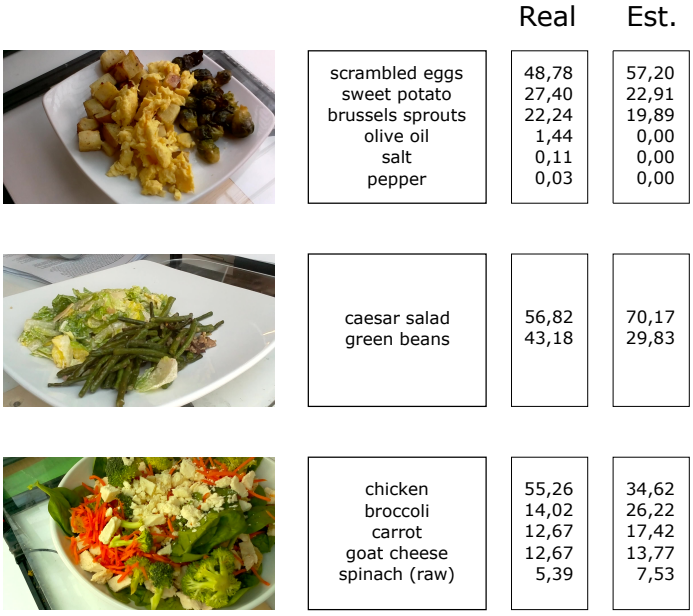

Figura 6.7: Ejemplos de resultados de aplicar la fase *TFPort* del modelo *TFIngPort* a imágenes del conjunto de evaluación de Nutrition5k. Empezando por la izquierda, en la segunda columna se muestran los ingredientes reales. En la tercera columna aparecen las proporciones reales de los ingredientes. En la cuarta columna se muestran las estimaciones realizadas por el modelo. Estas estimaciones se han obtenido a partir de los ingredientes reales del plato, sin aplicar la fase *TFIng*.

de los ingredientes, por lo que se consideraría que tiene una tasa de acierto de 0. Sin embargo, algunos de los errores son comprensibles, como confundir dos tipos de queso: "goat cheese" y "parmesan cheese".

Respecto a los ejemplos de *TFPort* (ver Figura 6.7), la mayoría de los errores son aceptables, aunque parece que el modelo tiende a estimar una proporción de 0 para aquellos ingredientes que tienen muy poco peso en el plato.

A continuación, vamos a analizar los platos con los peores resultados obtenidos por los modelos. En la Figura 6.8 se muestran imágenes de los 9 platos del conjunto de evaluación de Nutrition5k para los que la mejor versión de *FCNutr* (3 capas compartidas, sin *dropout*) obtiene los peores resultados. Los dos primeros platos están incorrectamente etiquetados en el *dataset*. El resto son platos con pocos ingredientes claramente separados entre sí, por lo que, en principio, parece fácil que el modelo produzca una buena estimación para estos ejemplos. Sin embargo, es complicado explicar por qué se producen estos errores.

Por otro lado, en la Figura 6.9 se muestran imágenes de los 9 platos del conjunto de evaluación de Nutrition5k para los que *TFIngPort* obtiene los peores resultados en la estimación de información nutricional. El plato de la primera imagen está etiquetado en el *dataset* con un solo ingrediente: "plate only". Este "ingrediente" solo aparece en dos platos del *dataset*, uno en el conjunto de validación y otro en el de evaluación, por lo que es lógico que el modelo no lo

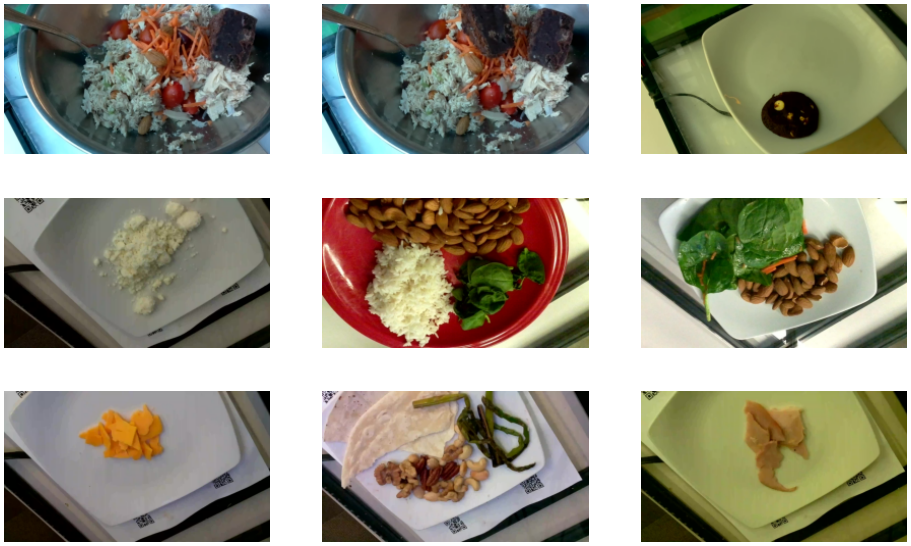

Figura 6.8: Imágenes de los 9 platos del conjunto de evaluación de Nutrition5k para los que la mejor versión de *FCNutr* (3 capas compartidas, sin *dropout*) obtiene los peores resultados. Se muestran ordenados de mayor a menor error, de arriba abajo y de izquierda a derecha.

Figura 6.9: Imágenes de los 9 platos del conjunto de evaluación de Nutrition5k para los que *TFIngPort* obtiene los peores resultados. Se muestran ordenados de mayor a menor error, de arriba abajo y de izquierda a derecha.

detecte, pues nunca lo ha visto durante el entrenamiento. El resto de los platos coinciden con aquellos para los que *FCNutr* producía las peores estimaciones. En este caso podemos analizar con más detalle el porqué de estos errores. Por ejemplo, para el plato en el que aparece queso en polvo, el modelo detecta

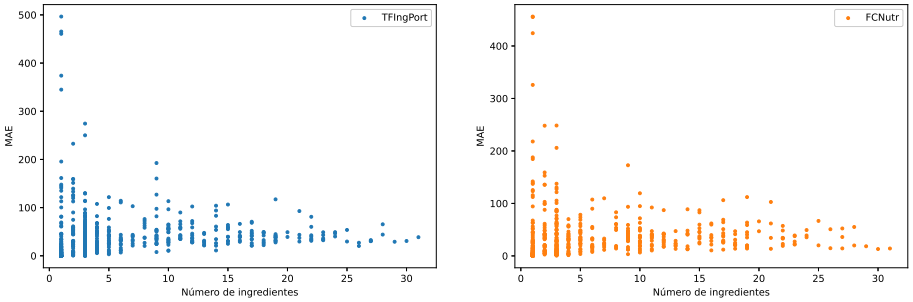

Figura 6.10: Diagramas de dispersión del error producido (MAE) por *TFIngPort* (izquierda) y la mejor versión de *FCNutr* (derecha) frente al número de ingredientes en el conjunto de evaluación de Nutrition5k.

arroz blanco o avena. Para el plato en el que aparece queso cheddar, detecta zanahorias y melón cantalupo (una variedad de melón de color naranja). Todos los errores tienen en común que los ingredientes reales se confunden con otros visualmente similares, pero con una composición nutricional muy distinta. En el caso del queso, el ingrediente real tiene muchas calorías y grasas, pero los ingredientes detectados (arroz, avena, zanahoria...) no. El efecto de estos fallos se nota especialmente en platos con pocos ingredientes, donde el ingrediente confundido tiene un peso importante.

A la vista de estos resultados, parece razonable pensar que ambos modelos siguen un razonamiento similar para producir sus predicciones, es decir, que *FCNutr* también trata de detectar los ingredientes y estimar sus proporciones para predecir la información nutricional del pato. Si observamos el diagrama de dispersión del error producido por ambos modelos frente al número de ingredientes (ver Figura 6.10), podemos observar que son extremadamente similares. Además, se observa que la mayoría de los errores crecen según aumenta el número de ingredientes, mientras que los errores más extremos se producen en platos con pocos ingredientes, lo que concuerda con lo observado en las Figuras 6.8 y 6.9.

Para evaluar la hipótesis de que ambos modelos siguen un razonamiento similar, vamos a estudiar la correlación entre las distribuciones de los errores producidos por ambos modelos. En la Figura 6.11 se muestran diagramas de dispersión representando la relación entre los errores producidos por los dos modelos para cada ejemplo del conjunto de evaluación de Nutrition5k. Se observa que hay una clara correlación positiva entre ambas variables. En la Tabla 6.5 se muestra el coeficiente de correlación de Pearson entre los errores de los modelos para cada valor nutricional. Todos los coeficientes son positivos y más cercanos a 1 que a 0, lo que indica que existe una correlación positiva entre los errores.

Estos datos solo demuestran la existencia de una correlación lineal positiva entre los errores de los modelos, es decir, que los errores varían de la misma manera y en el mismo sentido. Sin embargo, una explicación razonable para estas observaciones es que ambos modelos sigan un proceso similar para realizar las estimaciones, y por eso obtengan unos errores similares en cada ejemplo.

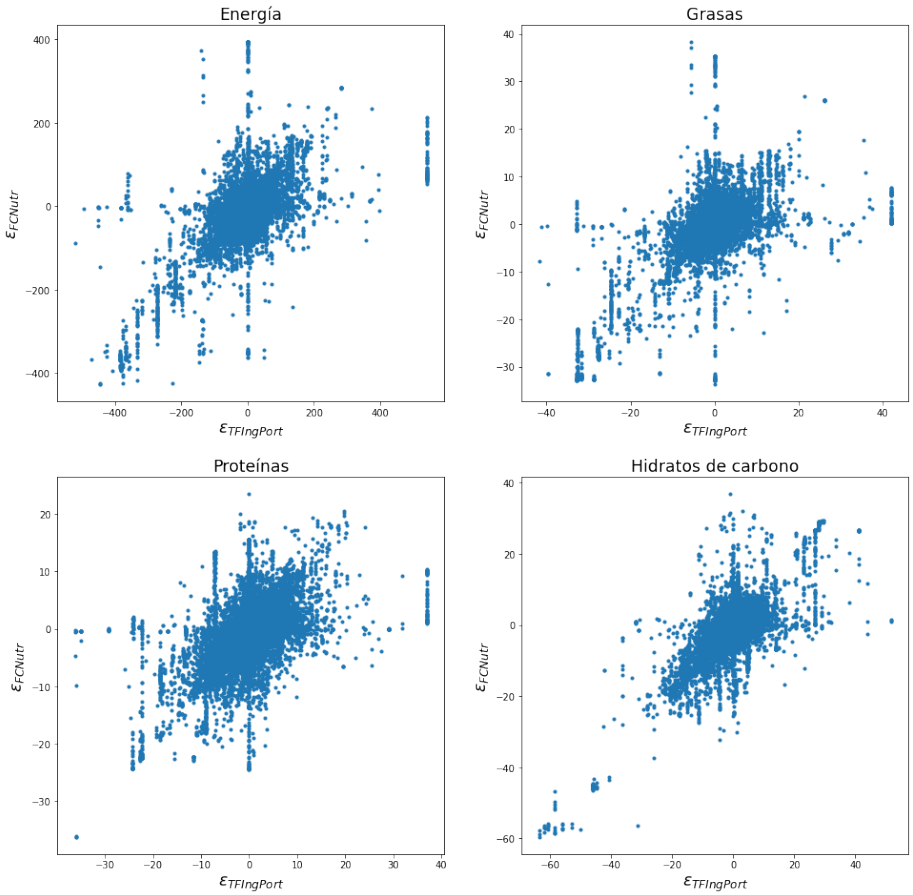

Figura 6.11: Diagramas de dispersión de los errores obtenidos por la mejor versión de *FCNutr* (3 capas compartidas, sin *dropout*) y *TFIngPort* en el conjunto de evaluación de Nutrition5k. El error al estimar la energía está expresado en kilocalorías, mientras que el resto está expresado en gramos.

Tabla 6.5: Coeficientes de correlación de Pearson entre los errores obtenidos por la mejor versión de *FCNutr* (3 capas compartidas, sin *dropout*) y *TFIngPort* en el conjunto de evaluación de Nutrition5k.

X	Y	$\rho_{X,Y}$
$\epsilon_{TFIngPort_{Energ.}}$	$\epsilon_{FCNutr_{Energ.}}$	0,61
$\epsilon_{TFIngPort_{Grasas}}$	$\epsilon_{FCNutr_{Grasas}}$	0,60
$\epsilon_{TFIngPort_{Prot.}}$	$\epsilon_{FCNutr_{Prot.}}$	0,53
$\epsilon_{TFIngPort_{H.Carb.}}$	$\epsilon_{FCNutr_{H.Carb.}}$	0,71

6.2 Limitaciones

Los resultados generales obtenidos por los modelos difieren de los presentados por Thames *et al.* [40]. Sin embargo, no se conoce la configuración exacta utili-

zada en sus experimentos y sus resultados no pudieron replicarse utilizando una configuración lo más parecida posible.

Por otro lado, aunque las observaciones realizadas acerca de la correlación entre los errores de los modelos son indicios de una posible similitud en su funcionamiento interno, ambos modelos no dejan de ser cajas negras. Es cierto que, al estar dividido en fases, *TFIngPort* permite interpretar sus predicciones más fácilmente. Sin embargo, las fases que lo componen no son cajas blancas. Existe todo un campo de estudio, llamado Inteligencia Artificial Explicable [13], acerca del desarrollo de modelos de aprendizaje automático o aprendizaje profundo cuyo funcionamiento interno pueda ser comprendido por humanos.

Por otra parte, estos resultados tienen una importante limitación, y es que todos los experimentos se han realizado usando un *dataset* relativamente pequeño y poco variado. Los resultados observados podrían no repetirse usando fotografías que contengan platos muy diferentes a los de Nutrition5k o que estén tomadas en entornos muy distintos.

Otra limitación es la escasez de trabajos similares realizados utilizando esta *dataset*, lo que hace complicado evaluar algunos resultados, puesto que no existen referencias con las que compararlos.

Capítulo 7

Conclusiones y trabajo futuro

Hemos diseñado dos arquitecturas de modelos para lograr el principal objetivo del trabajo: predecir la información nutricional de un plato a partir de una imagen utilizando aprendizaje profundo. Una de ellas está basada en una arquitectura existente y la otra es nueva, y se basa en la detección de ingredientes y la estimación de sus proporciones. A partir de estas arquitecturas, hemos entrenado varios modelos con el *dataset* Nutrition5k y hemos medido su rendimiento. Los resultados obtenidos demuestran que es posible cumplir esta tarea con un error aceptable. También concluimos que la predicción de los valores nutricionales basada en la detección de ingredientes y la estimación de sus proporciones no ayuda a reducir el error, pero aporta más datos acerca de la composición del plato y permite conocer con más detalle el porqué de las predicciones realizadas por el modelo.

También hemos detectado que existen indicios de que los modelos entrenados para predecir la información nutricional del plato directamente a partir de una imagen son capaces de detectar internamente sus ingredientes y estimar las proporciones de esos ingredientes en el plato, puesto que sus errores presentan una correlación fuerte con los producidos por el modelo entrenado para detectar ingredientes y estimar porciones.

7.1 Trabajo futuro

La principal limitación de los resultados obtenidos es que se ha utilizado un *dataset* relativamente pequeño y poco variado. Si bien hemos partido de modelos preentrenados, los principales avances en aprendizaje profundo se han producido con *dataset*s con un gran número de ejemplos. Para avanzar en el campo de la comprensión automática de los alimentos y su composición se necesita crear un *dataset* a gran escala que contenga datos de calidad. La creación de un *dataset* con estas características es complicada, puesto que anotar correctamente la información nutricional y la composición de un plato es un proceso que requiere tiempo e intervención humana.

Si se llegase a crear un *dataset* así, futuras investigaciones podrían utilizarlo para evaluar el rendimiento de las arquitecturas presentadas en este trabajo. También se podrían evaluar utilizando el *dataset* Recipe1M+ [24], siempre que se dispusiese del hardware necesario, aunque la calidad de los datos de este *dataset* no es la ideal para el objetivo del presente trabajo.

7.2 Difusión de los resultados

El código de los modelos y los enlaces para descargar sus pesos entrenados están disponibles en `https://github.com/davidaf3/ReverseNutrition`.

Bibliografía

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. `www.tensorflow.org`, 2015.

[2] R. A. Adan, E. M. van der Beek, J. K. Buitelaar, J. F. Cryan, J. Hebebrand, S. Higgs, H. Schellekens, and S. L. Dickson. Nutritional psychiatry: Towards improving mental health by what you eat. *European Neuropsychopharmacology*, 29(12):1321–1332, 2019.

[3] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[4] M. Bolaños, A. Ferrà, and P. Radeva. Food ingredients recognition through multi-label learning. In *New Trends in Image Analysis and Processing – ICIAP 2017*, pages 394–402, Cham, 2017. Springer International Publishing.

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[6] M. J. Butler and R. M. Barrientos. The impact of nutrition on covid-19 susceptibility and long-term consequences. *Brain, Behavior, and Immunity*, 87:53–54, 2020.

[7] M. Carvalho, R. Cadène, D. Picard, L. Soulier, N. Thome, and M. Cord. Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings. In *The 41st International ACM SIGIR Conference on Re-*

search & Development in Information Retrieval, SIGIR '18, page 35–44, New York, NY, USA, 2018. Association for Computing Machinery.

[8] J. Chen and C.-w. Ngo. Deep-based ingredient recognition for cooking recipe retrieval. In *Proceedings of the 24th ACM International Conference on Multimedia*, MM '16, page 32–41, New York, NY, USA, 2016. Association for Computing Machinery.

[9] J.-j. Chen, C.-W. Ngo, and T.-S. Chua. Cross-modal recipe retrieval with rich food attributes. In *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, page 1771–1779, New York, NY, USA, 2017. Association for Computing Machinery.

[10] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[11] F. Chollet et al. Keras. `keras.io`, 2015.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, jun 2019. Association for Computational Linguistics.

[13] F. K. Došilović, M. Brčić, and N. Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0210–0215, 2018.

[14] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[15] A. Goyal, A. Lamb, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4608–4616. Curran Associates Inc., 2016.

[16] O. Guardamagna, F. Abello, P. Cagliero, and L. Lughetti. Impact of nutrition since early life on cardiovascular prevention. *Italian Journal of Pediatrics*, 38(1):73, Dec 2012.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[18] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[19] A. B. Jelodar and Y. Sun. Calorie aware automatic meal kit generation from an image. *CoRR*, abs/2112.09839, 2021.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[21] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[22] C. D. Lee, J. Chae, T. E. Schap, D. A. Kerr, E. J. Delp, D. S. Ebert, and C. J. Boushey. Comparison of known food weights with image-based portion-size automated estimation and adolescents' self-reported portion size. *Journal of Diabetes Science and Technology*, 6(2):428–434, 2012.

[23] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, and Y. Ma. Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment. In C. K. Chang, L. Chiari, Y. Cao, H. Jin, M. Mokhtari, and H. Aloulou, editors, *Inclusive Smart Cities and Digital Health*, pages 37–48, Cham, 2016. Springer International Publishing.

[24] J. Marin, A. Biswas, F. Ofli, N. Hynes, A. Salvador, Y. Aytar, I. Weber, and A. Torralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

[25] S. Mezgec and B. K. Seljak. Nutrinet: A deep learning food and drink image recognition system for dietary assessment. *Nutrients*, 9, 2017.

[26] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[27] A. Mithal, J.-P. Bonjour, S. Boonen, P. Burckhardt, H. Degens, G. El Hajj Fuleihan, R. Josse, P. Lips, J. Morales Torres, R. Rizzoli, N. Yoshimura, D. A. Wahl, C. Cooper, and B. Dawson-Hughes. Impact of nutrition on muscle mass, strength, and performance in older adults. *Osteoporosis International*, 24(5), May 2013.

[28] S. P. Mohanty, G. Singhal, E. A. Scuccimarra, D. Kebaili, H. Héritier, V. Boulanger, and M. Salathé. The food recognition benchmark: Using deep learning to recognize food in images. *Frontiers in Nutrition*, 9, 2022.

[29] M. Muscaritoli. The impact of nutrients on mental health and well-being: Insights from the literature. *Frontiers in Nutrition*, 8, 2021.

[30] A. Myers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. Murphy. Im2calories: Towards an automated mobile vision food diary. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1233–1241, 2015.

[31] A. Salvador, M. Drozdzal, X. Giro-i Nieto, and A. Romero. Inverse cooking: Recipe generation from food images. In *2019 IEEE/CVF Conference on*

Computer Vision and Pattern Recognition (CVPR), pages 10445–10454, 2019.

[32] A. Salvador, N. Hynes, Y. Aytar, J. Marin, F. Ofli, I. Weber, and A. Torralba. Learning cross-modal embeddings for cooking recipes and food images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[33] T. E. Schap, B. L. Six, E. J. Delp, D. S. Ebert, D. A. Kerr, and C. J. Boushey. Adolescents in the united states can identify familiar foods at the time of consumption and when prompted with an image 14 h postprandial, but poorly estimate portions. *Public health nutrition*, 14(7):1184–1191, Jul 2011.

[34] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, Jul 2019.

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[37] Stanford Vision Lab. ImageNet, 2021. `www.image-net.org`. Accedido: 20-05-2022.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[39] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[40] Q. Thames, A. Karpur, W. Norris, F. Xia, L. Panait, T. Weyand, and J. Sim. Nutrition5k: Towards automatic nutritional understanding of generic food. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8899–8907, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society.

[41] U.S. Department of Agriculture, Agricultural Research Service. FoodData Central, 2019. `fdc.nal.usda.gov`.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates Inc., 2017.

[43] X. Wang, D. Kumar, N. Thome, M. Cord, and F. Precioso. Recipe recognition with large multimodal food dataset. In *2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6, 2015.

[44] Z. Zhu and Y. Dai. Food ingredients identification from dish images by deep learning. *Journal of Computer and Communications*, 9(4):85–101, abril 2021.

[45] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Proceedings of the Institute of Radio Engineers*, 109(1):43–76, Jan. 2021.

Apéndice A

Planificación y presupuesto

A.1 Planificación

La planificación del proyecto incluye todas las tareas a realizar, desde la lectura de trabajos relacionados hasta la elaboración de esta memoria. En la Tabla A.1 aparecen las tareas principales del WBS del proyecto (*Work Breakdown Structure*, Estructura de Desglose de Trabajo), junto con su duración estimada y sus fechas planeadas de inicio y fin. Esta planificación supone una duración total de aproximadamente 3 meses. Se planificó que el proyecto comenzase el 8 de marzo de 2022 y se estimó que se completaría el 10 de junio de 2022, 2 días antes del final del plazo para la presentación de la memoria.

Todas las tareas fueron asignadas a una única persona con el rol de investigador, a excepción de las tareas de entrenamiento de los modelos, que se dejaron sin asignar. Se estableció un horario de cuatro horas diarias para el investigador, de modo que pudiera compatibilizar el desarrollo del proyecto con el resto de asignaturas. A las tareas de entrenamiento se les asignó un horario de 24 horas diarias, puesto que el servidor donde se ejecutaron puede trabajar ininterrumpidamente. En la Figura A.1 se muestra el diagrama de Gantt del proyecto con todas las tareas.

Tabla A.1: Tareas principales del WBS del proyecto.

Tarea	Inicio	Fin	Duración
Revisión de trabajo relacionado	08/03/22	18/03/22	36 h
Selección de dataset	21/03/22	24/03/22	16 h
Diseño de la arquitectura de los modelos	25/03/22	08/04/22	44 h
Implementación de los modelos	11/04/22	18/04/22	24 h
Entrenamiento de los modelos	18/04/22	24/04/22	130 h
Medición de los resultados	25/04/22	02/05/22	24 h
Análisis de los resultados	03/05/22	06/05/22	16 h
Redacción de la memoria	09/05/22	07/06/22	88 h
Revisión de la memoria	08/06/22	10/06/22	12 h

Id	EDT	Nombre de tarea
1	**1**	**Proyecto**
2	**1.1**	**Revisión de trabajo relacionado**
3	1.1.1	Visión artificial
4	1.1.2	Detección de ingredientes
5	1.1.3	Estimación de información nutricional
6	1.1.4	Modelos de secuencia a secuencia
7	**1.2**	**Selección de dataset**
8	1.2.1	Análisis de Recipe1M
9	1.2.2	Análisis de Nutrition5k
10	1.2.3	Comparación de datasets analizados
11	**1.3**	**Diseño de la arquitectura de los modelo**
12	1.3.1	Diseño de FCNutr
13	**1.3.2**	**Diseño de TFIngPort**
14	1.3.2.1	Diseño de TFIng
15	1.3.2.2	Diseño de TFPort
16	**1.4**	**Implementación de los modelos**
17	1.4.1	Implementación de FCNutr
18	**1.4.2**	**Implementación de TFIngPort**
19	1.4.2.1	Implementación de TFIng
20	1.4.2.2	Implementación de TFPort
21	**1.5**	**Entrenamiento de los modelos**
22	1.5.1	Entrenamiento de FCNutr
23	**1.5.2**	**Entrenamiento de TFIngPort**
24	1.5.2.1	Entrenamiento de TFIng
25	1.5.2.2	Entrenamiento de TFPort
26	**1.6**	**Medición de los resultados**
27	1.6.1	Medición de los resultados generales
28	1.6.2	Medición de los resultados de TFIng
29	1.6.3	Medición de los resultados de TFPort
30	1.7	Análisis de los resultados
31	**1.8**	**Redacción de la memoria**
32	1.8.1	Redacción de fijación de objetivos
33	1.8.2	Redacción de trabajo relacionado
34	1.8.3	Redacción de descripción del sistema
35	1.8.4	Redacción de metodología
36	1.8.5	Redacción de resultados obtenidos
37	1.8.6	Redacción de conclusiones
38	1.8.7	Redacción de introducción
39	1.8.8	Redacción de resumen
40	1.8.9	Redacción de apéndices
41	1.9	Revisión de la memoria

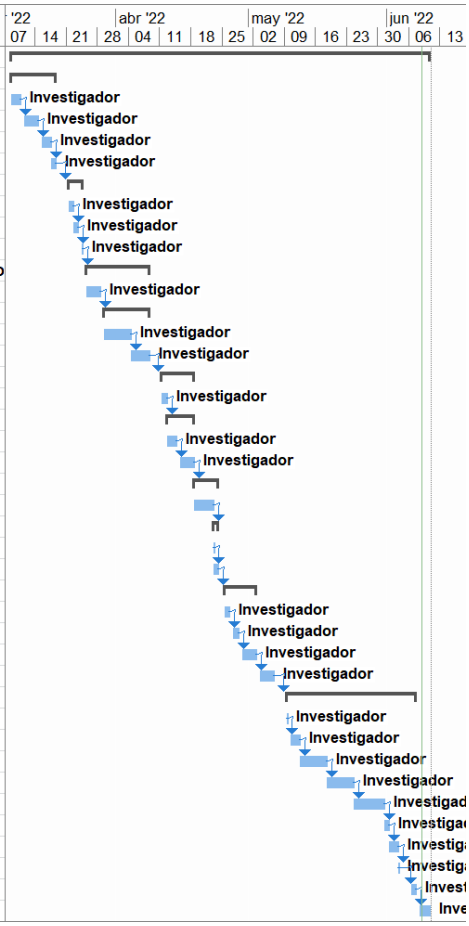

Figura A.1: Diagrama de Gantt del proyecto.

A.2 Presupuesto

En la Tabla A.2 se muestra el presupuesto del proyecto. El proyecto tiene un coste total de 8.222,50 €. Para calcular el coste del entrenamiento se utilizó el precio por hora de una máquina virtual del Google Compute Engine con unas características lo más similares posibles a las del servidor utilizado en el proyecto. Este precio es de aproximadamente 60 céntimos por hora.

Tabla A.2: Presupuesto del proyecto.

Partida	Ítem	Descripción	Importe	Total
01		Revisión de literatura y selección de dataset		1.628,90 €
	01	Revisión de trabajo relacionado	1.127,70 €	
	02	Selección de dataset	501,20 €	
02		Diseño e implementación de modelos		2.130,10 €
	01	Diseño de la arquitectura de los modelos	1.378,30 €	
	02	Implementación de los modelos	751,80 €	
03		Entrenamiento de los modelos		78,00 €
	01	Entrenamiento de FCNutr	57,60 €	
	02	Entrenamiento de TFIngPort	20,40 €	
04		Medición y análisis de resultados		1.253,00 €
	01	Medición de resultados	751,80 €	
	02	Análisis de resultados	501,20 €	
05		Elaboración de la memoria		3.132,50 €
	01	Redacción de la memoria	2.756,60 €	
	02	Revisión de la memoria	375,90 €	
Total				8.222,50 €

Apéndice B

Ejemplos de predicciones

En este apéndice se incluyen ejemplos de predicciones completas realizadas por los modelos entrenados a partir de imágenes del conjunto de evaluación de Nutrition5k. Para cada ejemplo se muestran los ingredientes y sus proporciones reales, los ingredientes y proporciones estimados por *TFIngPort*, la información nutricional real, los valores nutricionales estimados por la mejor versión de *FCNutr* (3 capas compartidas, sin *dropout*) y los calculados a partir de las predicciones de *TFIngPort*.

Real		Estimación	
Ingredientes	Proporciones	Ingredientes	Proporciones
scrambled eggs sweet potato brussels sprouts olive oil salt pepper	48,78 27,40 22,24 1,44 0,11 0,03	sweet potato scrambled eggs brussels sprouts olive oil salt pepper	58,89 24,31 16,79 0,00 0,00 0,00

	Información nutricional			
	Energía	Grasas	Proteínas	H. de carbono
Real	99,94	4,52	3,98	11,22
FCNutr	136,48	8,94	7,32	6,65
TFIngPort	112,86	6,53	6,70	6,83

Figura B.1: Predicciones realizadas por los modelos a partir de una imagen del plato 1562776497 tomada por la cámara D.

Real		Estimación	
Ingredientes	Proporciones	Ingredientes	Proporciones
caesar salad green beans	56,82 43,18	caesar salad green beans olive oil	70,26 29,38 0,36

	Información nutricional			
	Energía	Grasas	Proteínas	H. de carbono
Real	38,39	1,24	2,60	5,47
FCNutr	50,22	1,99	3,49	6,44
TFIngPort	43,25	1,87	2,78	5,08

Figura B.2: Predicciones realizadas por los modelos a partir de una imagen del plato 1561661347 tomada por la cámara B.

Real		Estimación	
Ingredientes	Proporciones	Ingredientes	Proporciones
chicken broccoli carrot goat cheese spinach (raw)	55,26 14,02 12,67 12,67 5,39	olives parmesan cheese carrot spinach (raw)	53,51 18,27 18,03 10,19

	Información nutricional			
	Energía	Grasas	Proteínas	H. de carbono
Real	135,96	4,75	20,14	2,44
FCNutr	139,30	8,74	10,47	5,49
TFIngPort	149,03	10,96	7,81	6,32

Figura B.3: Predicciones realizadas por los modelos a partir de una imagen del plato 1565811091 tomada por la cámara A.

Real		Estimación	
Ingredientes	Proporciones	Ingredientes	Proporciones
tuna salad fried rice squash carrot avocado olive oil salt	40,64 21,03 20,22 13,37 4,46 0,20 0,08	tuna cherry tomatoes carrot onions olives	45,34 18,20 15,41 12,29 8,76

	Información nutricional			
	Energía	Grasas	Proteínas	H. de carbono
Real	127,71	5,16	7,96	12,45
FCNutr	103,88	1,98	15,89	4,03
TFIngPort	83,53	1,29	13,66	3,91

Figura B.4: Predicciones realizadas por los modelos a partir de una imagen del plato 1568401302 tomada por la cámara A.

Real		Estimación	
Ingredientes	Proporciones	Ingredientes	Proporciones
chicken cauliflower cucumbers	46,75 27,55 26,32	chicken cauliflower cucumbers	46,80 27,26 25,94

	Información nutricional			
	Energía	Grasas	Proteínas	H. de carbono
Real	86,66	1,77	15,25	1,90
FCNutr	84,17	1,72	15,40	1,37
TFIngPort	86,63	1,77	15,26	1,88

Figura B.5: Predicciones realizadas por los modelos a partir de una imagen del plato 1550706705 tomada por la cámara C.