

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación.



*Estimación de composición calórica de
alimentos utilizando técnicas de
Aprendizaje Profundo y Automático.*

Tesis para obtener el grado de Maestría en:

Ciencias de la Computación

Presenta:

Jorge Alberto Cabrero Dávila

Director: Dr. Ivo Pineda

Asesor: Dra. Concepción Pérez

18 de noviembre de 2022

Resumen

En este trabajo se presenta una propuesta para la obtención de una estimación calórica de un platillo. La propuesta se basa en métodos para el procesamiento de las imágenes digitales de los platillos, como son la segmentación y detección de objetos a través de técnicas de Aprendizaje Profundo y algoritmos de Aprendizaje automático.

Índice

1. Agradecimientos.	5
2. Antecedentes del proyecto.	6
2.1. Un problema de Salud pública en México.	6
2.2. Origen y objetivo del tema de tesis.	7
3. Conceptos Tecnológicos y Estado del Arte.	8
3.1. Procesamiento de imágenes digitales.	8
3.1.1. ¿Qué es la visión por computadora ?	8
3.1.2. Clasificación de imágenes.	11
3.1.3. Detección de objetos en imágenes.	12
3.1.4. Segmentación de imágenes.	14
3.1.5. Tipos de Segmentación de imágenes.	15
3.2. Aprendizaje Automático y Aprendizaje Profundo.	18
3.2.1. Enfoques del aprendizaje automático.	19
3.2.2. Modelos del aprendizaje automático.	20
3.2.3. Aprendizaje Profundo.	26
3.2.4. Redes Neuronales Profundas.	27
3.3. Estado del Arte.	28
3.3.1. Estado del Arte Enfoques no móviles.	29
3.3.2. Estado del Arte Enfoques móviles.	31
3.3.3. Joutout y Yanai, 2009.	32
3.3.4. Puri et al., 2009.	32
3.3.5. Kong y Tan, 2012.	33
3.3.6. Rahman et al., 2012.	34
3.3.7. Kawano y Yanai, 2013.	35
3.3.8. Anthimopoulos et al., 2013.	36
3.3.9. Pouladzadeh, Shirmohammadi y Arici, 2013.	36
4. Propuesta de diseño.	37
4.1. Captura de imagen digital.	39
4.2. Detección de objeto.	39
4.3. Segmentación.	40
4.4. Cálculo de áreas segmentadas.	40
4.5. Cálculo de calorías.	41

5. Pruebas y resultados.	42
5.1. Prototipado - Clasificación de imágenes usando Aprendizaje Profundo.	42
5.2. Detección de platillos en una escena utilizando Redes Convolucionales Profundas.	50
5.2.1. Propuesta de diseño - Detección de platillo.	51
5.2.2. Resultados - Detección de platillos.	53
5.3. Segmentación de alimentos en un platillo usando un clasificador de Bosque Aleatorio.	59
5.3.1. Extracción de características de imágenes.	60
5.3.2. Desenfoque Gaussiano.	60
5.3.3. Operador de Sobel.	62
5.3.4. Matriz Hessiana.	63
5.3.5. Resultados - Segmentación con clasificador de Bosque Aleatorio.	64
5.4. Estimación calórica basada en área de imagen segmentada. .	67
6. Conclusiones.	73
6.1. Investigación futura.	73

1. Agradecimientos.

A Alhely, Sammy, Bengi, Chai y Miel que creen que puedo lograr lo que sea.

A mis profesores, amigos, compañeros y todos los que sin darse cuenta contribuyeron a la creación de este trabajo.

A mi nona, que no podrá ver lo que me dio.

2. Antecedentes del proyecto.

2.1. Un problema de Salud pública en México.

Según datos de la secretaría de Salud [1], en México siete de cada diez adultos, cuatro de cada diez adolescentes y uno de cada tres niños presentan sobrepeso u obesidad. Por estas cifras las enfermedades degenerativas relacionadas con la alimentación se consideran un grave problema de salud pública.

Se estima que 90 por ciento de los casos de diabetes mellitus tipo 2 son atribuibles al sobrepeso y la obesidad. Actualmente esta enfermedad se ubica como la **primera** causa de muerte entre la población mexicana si considera la relación que existe entre las enfermedades coronarias y la diabetes.

En México la mortalidad por diabetes ha tenido un incremento de 7.2 veces en el lapso entre 1980 a 2016, y en la actualidad tres de cada 20 fallecimientos son por esta enfermedad.

Se estima que de 2000 a 2016 se han acumulado más de un millón 200 mil decesos como consecuencia de la diabetes y sus complicaciones.

Además, es la principal causa de años de vida ajustados por discapacidad y la tercera de años de vida perdidos por muerte prematura.

El 8 de Enero de 2020 José Ángel Gurría, secretario general de la Organización para la Cooperación y el Desarrollo económico (OCDE) emitió un comunicado [2] en el cual menciona que en México se presenta el caso más preocupante en el tema de obesidad y diabetes de los países que pertenecen a la organización. Enlista los siguientes datos:

- El 73 % de la población mexicana sufre sobrepeso (comparado con 20 % en 1996).
- 34 % de las personas con sobrepeso son obesas mórbidas.
- Las enfermedades asociadas con el sobrepeso reducirán 4 años la esperanza de vida en México para 2050.

- El crecimiento en la obesidad infantil es el mayor para los países de la OCDE duplicándose de 7.5 % en 1996 a 15 % en 2016.
- La fuerza laboral en México se reducirá en 2.4 millones de trabajadores de tiempo completo por año debido a las enfermedades relacionadas con el sobrepeso.
- El tratamiento de las enfermedades relacionadas con el sobrepeso representarán el 8.9 % del gasto total en salud entre 2020 y 2050.
- Se reducirá el PIB de México en 5.3 % debido a las enfermedades relacionadas con el sobrepeso.

Todos los datos anteriores concretizan la sentencia de que la obesidad y en especial las enfermedades relacionadas con ésta; como la diabetes y los padecimientos coronarios, representan el reto más grande de salud pública en México. En consecuencia, el objetivo principal del proyecto de tesis es poder crear un mecanismo que pueda ayudar a revertir las tendencias estimadas por la OCDE.

2.2. Origen y objetivo del tema de tesis.

Actualmente la población en México no cuenta con herramientas que sean fáciles de utilizar para medir y cuantificar sus alimentos, el gobierno e instituciones privadas han creado diversos lineamientos de dietas que son difíciles de seguir, caen en la ambigüedad y no son estrictamente medibles a menos que se opte por pesar y cuantificar cada ingrediente de cada alimento que se consume; lo cual, resulta imposible para la mayoría de las personas que se encuentran en los estratos sociales medios y bajos que no cuentan ni con el tiempo ni los recursos para llevar a cabo dicha tarea. Además, los esfuerzos gubernamentales como el etiquetado de productos no han tenido los resultados esperados de disminución de consumo.

En suma hoy por hoy no existe una herramienta fácil de usar que sea capaz de estimar o medir de forma precisa la alimentación completa de un individuo. Es a partir de esta situación que nace la idea del proyecto de tesis;

con el objetivo final de que el mexicano promedio pueda medir, cuantificar y tomar decisiones informadas sobre lo que come y así poder revertir las tendencias más alarmantes del país.

3. Conceptos Tecnológicos y Estado del Arte.

La intención de las siguientes tres secciones es ofrecer una visión general de los conceptos técnicos más recientes sobre los cuales está basado el proyecto de tesis. Como ya se mencionó anteriormente el proyecto utiliza procesamiento de imágenes digitales. Debido a esto, es importante entender primero ¿Qué es la visión por computadora? Después es necesario conocer los conceptos de: clasificación de imágenes digitales, segmentación de imágenes digitales, tipos de segmentación, reconocimiento de objetos y detección de objetos.

Además del procesamiento de imágenes digitales el proyecto utiliza técnicas y conceptos de Aprendizaje Profundo y Aprendizaje Automático que son necesarios presentar también. Estos se presentarán en la sección de Aprendizaje Profundo y Aprendizaje Automático.

Para finalizar la revisión de conceptos; en la sección del Estado del Arte, se revisarán proyectos similares de todo el mundo que han tratado de resolver el mismo problema, se examinarán sus resultados y que retos presentan.

3.1. Procesamiento de imágenes digitales.

3.1.1. ¿Qué es la visión por computadora ?

Como seres humanos somos capaces de percibir objetos y estructuras tridimensionales de la realidad que nos rodea con aparente facilidad. Richard Szeliski en su libro Computer Vision: Algoritmos y Aplicaciones [3] nos describe el ejemplo de cuán vívido es observar un vaso lleno de agua o un florero sobre una mesa justo frente a nosotros. Podríamos distinguir la forma del vaso, del florero, de las flores, también podríamos distinguir la translucencia del líquido en el interior del vaso o de las hojas y los pétalos de las flores en el florero. También podríamos diferenciar los patrones de luz y sombra que

se forman sobre la mesa y sin mucho esfuerzo podríamos separar los objetos del fondo donde se encuentran.

También si observáramos un retrato con un grupo de personas en él, nos sería muy fácil determinar cuantas personas se encuentran en la escena, o cual es el estado de ánimo de cada una de las personas basados en las expresiones de sus rostros.

Todos estos procesos naturales representan el concepto de **visión humana** a la cual estamos muy acostumbrados. Sin embargo, replicar todos estos procesos en una computadora (los cuales un ser humano de dos años puede llevar a cabo sin mayor complicación) todavía nos elude.

En consecuencia es justamente la replicación de los procesos naturales de visión humana llevados a cabo por una máquina, la definición de visión por computadora que se ha deducido en este trabajo a partir de las observaciones de Szeliski.

Aunque la definición a la que se ha llegado es suficientemente general es ventajoso dar una definición más específica. Combinando los trabajos de Dana Ballard et. al. [4] y Huang T. [5] podemos construir la siguiente definición:

La visión por computadora es un campo científico interdisciplinario que se ocupa de cómo las computadoras pueden obtener una comprensión de alto nivel a partir de imágenes o videos digitales. Desde la perspectiva de la ingeniería, busca comprender y automatizar las tareas que puede realizar el sistema visual humano.

Las tareas de visión por computadora incluyen métodos para adquirir, procesar, analizar y comprender imágenes digitales, y la extracción de datos de alta dimensión del mundo real para producir información numérica o simbólica, por ejemplo, en forma de decisiones [6] [7] [8] [9].

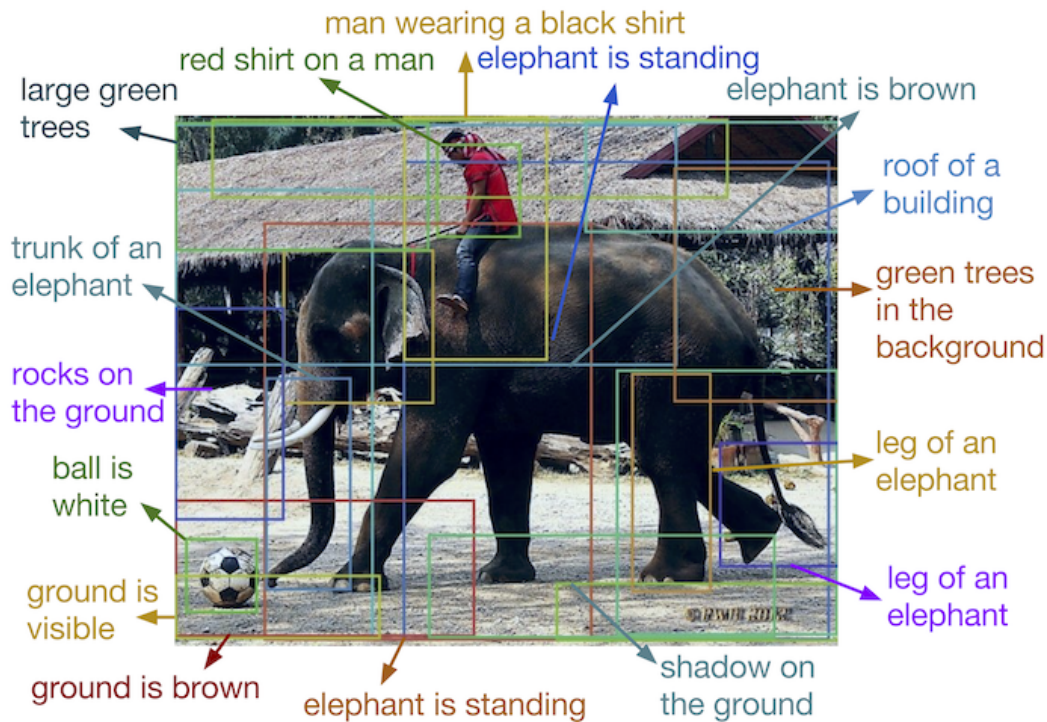


Figura 1: Ejemplo de resultado del software DenseCap analizando la fotografía de un hombre montando un elefante. [10]

La comprensión en este contexto significa la transformación de imágenes visuales (la entrada de la retina) en descripciones del mundo que tienen sentido para los procesos de pensamiento y pueden provocar la acción apropiada. Esta comprensión de la imagen puede verse como la separación de la información simbólica de los datos de la imagen utilizando modelos construidos con la ayuda de la geometría, la física, la estadística y la teoría del aprendizaje.

Los subdominios de la visión artificial (como también suele llamarse) incluyen: la reconstrucción de escenas, la detección de objetos, la detección de eventos, el seguimiento de video, el reconocimiento de objetos, la estimación de poses en 3D, el aprendizaje, la indexación, la estimación de movimiento, el control visual, el modelado de escenas en 3D, la restauración de imágenes y muchos otros más.

3.1.2. Clasificación de imágenes.

La clasificación de imágenes es la tarea de asignar una etiqueta o clase a una imagen completa. Se espera que las imágenes tengan solo una clase para cada imagen. Los modelos de clasificación de imágenes toman una imagen como entrada y devuelven una predicción sobre a qué clase pertenece la imagen [11].

La figura 2 muestra el proceso de la entrada y salida en un modelo de clasificación, la entrada en este caso es la imagen de un gato, la salida es la probabilidad de que la imagen represente un gato egipcio o un gato atigrado o un gato tigre.

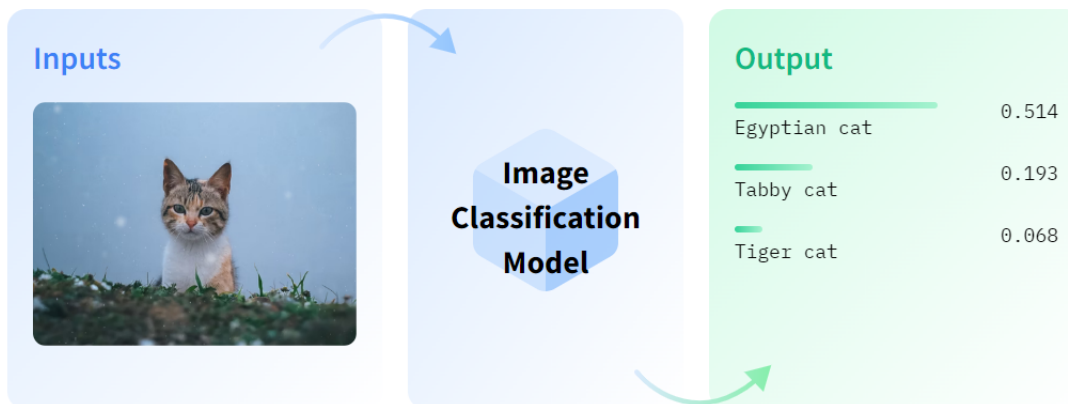


Figura 2: Ejemplo de entrada y salida de un modelo de clasificación de imágenes . [12]

Los modelos de clasificación de imágenes se pueden utilizar cuando no estamos interesados en instancias específicas de objetos con información de ubicación o su forma.

Los modelos de clasificación de imágenes se utilizan ampliamente en la fotografía de stock para asignar a cada imagen una palabra clave.

Los modelos entrenados en clasificación de imágenes pueden mejorar la experiencia del usuario organizando y categorizando galerías de fotos en el teléfono o en la nube, en múltiples palabras clave o etiquetas.

3.1.3. Detección de objetos en imágenes.

La detección de objetos es una tecnología informática relacionada con la visión artificial y el procesamiento de imágenes que se ocupa de detectar instancias de objetos semánticos de una determinada clase (como humanos, edificios o automóviles) en imágenes y videos digitales. [13]

Cada clase de objeto tiene sus propias características especiales que ayudan a clasificar la clase; por ejemplo, todos los círculos son redondos. La detección de clases de objetos utiliza estas características especiales. Por ejemplo, cuando se buscan círculos, se buscan objetos que están a una distancia particular de un punto (es decir, el centro). De manera similar, al buscar cuadrados, se necesitan objetos que sean perpendiculares en las esquinas y que tengan lados de igual longitud. Se utiliza un enfoque similar para la identificación de rostros donde se pueden encontrar los ojos, la nariz y los labios y se pueden encontrar características como el color de la piel y la distancia entre los ojos.

Los dominios bien investigados de detección de objetos incluyen la detección de rostros y la detección de peatones. La detección de objetos tiene aplicaciones en muchas áreas de la visión artificial, incluida la recuperación de imágenes y la videovigilancia.

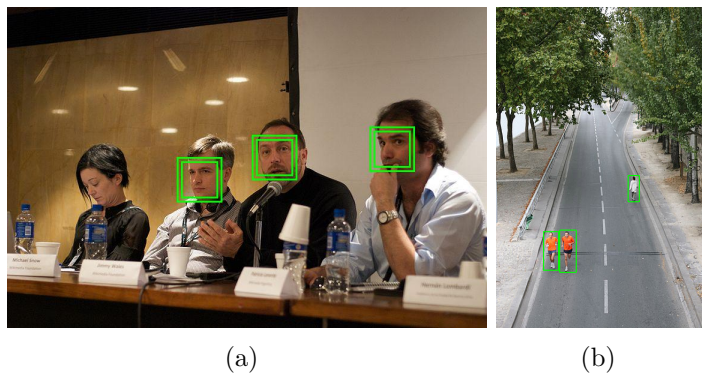


Figura 3: (a) Detección de rostros [14] (b) Detección de peatones [15]

neuronales pueden realizar la detección de objetos de un extremo a otro sin definir específicamente las características y, por lo general, se basan en redes neuronales convolucionales (CNN).

Enfoques No-neuronales:

- Marco de detección de objetos de Viola-Jones basado en características de Haar. [21]
- Transformación de características de escala invariable (SIFT). [22]
- Histogram of oriented gradients (HOG) features. [23]

Enfoques Neuronales:

- Propuestas de Región (R-CNN [24], Fast R-CNN [25], Faster R-CNN [26], cascade R-CNN [27])
- Single Shot MultiBox Detector (SSD) [28]
- You Only Look Once (YOLO) [29] [30] [31] [32]
- RetinaNet [33] [27]
- Redes Convolucionales Deformables [34] [35]

3.1.4. Segmentación de imágenes.

En el procesamiento de imágenes digitales y la visión por computadora, la segmentación de imágenes es el proceso de dividir una imagen digital en varios segmentos de imagen, también conocidos como regiones de imagen u objetos de imagen (conjuntos de píxeles). El objetivo de la segmentación es simplificar y/o cambiar la representación de una imagen en algo que sea más significativo y más fácil de analizar. [36] [37]

La segmentación de imágenes se utiliza normalmente para localizar objetos y límites (líneas, curvas, etc.) en las imágenes. Con mayor precisión, la segmentación de imágenes es el proceso de asignar una etiqueta a cada píxel de una imagen de modo que los píxeles con la misma etiqueta compartan ciertas características.

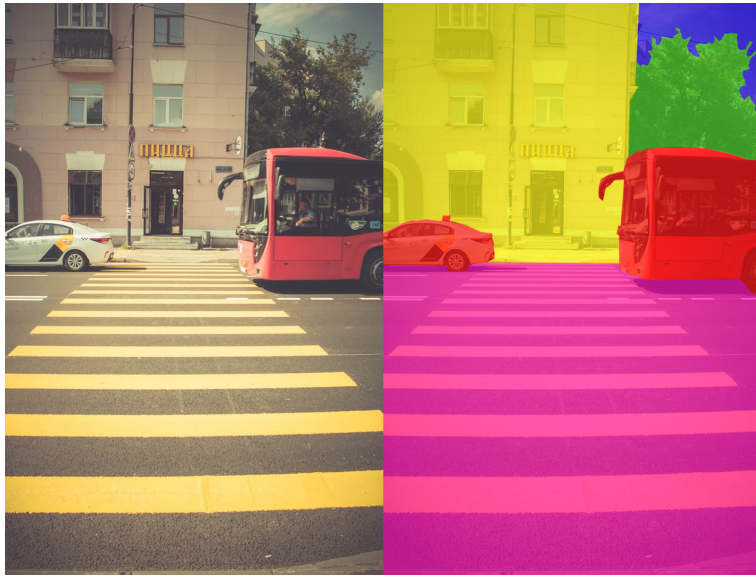


Figura 5: Ejemplo de segmentación de imagen (semántica). Cada color en la imagen segmentada tiene un significado distinto. [41]

El resultado de la segmentación de imágenes es un conjunto de segmentos que en conjunto cubren toda la imagen, o un conjunto de contornos extraídos de la imagen (detección de bordes). Cada uno de los píxeles de una región es similar con respecto a alguna característica o propiedad calculada [38], como el color, la intensidad o la textura. Las regiones adyacentes tienen un color significativamente diferente con respecto a las mismas características. [39] Cuando se aplica a un conjunto de imágenes, típico en imágenes médicas, los contornos resultantes después de la segmentación de la imagen se pueden usar para crear reconstrucciones 3D con la ayuda de algoritmos de interpolación como *marching cubes* [40].

3.1.5. Tipos de Segmentación de imágenes.

Segmentación Semántica.

La segmentación semántica consiste en organizar los píxeles de una imagen en función de las clases semánticas. En este modelo, cada píxel pertenece a una clase específica y el modelo de segmentación no hace referencia a ningún

otro contexto o información. [42]

Por ejemplo, la segmentación semántica realizada en una imagen con varios árboles y vehículos proporcionará una máscara que clasifica todos los tipos de árboles en una clase **árbol** y todos los tipos de vehículos, ya sean autobuses, automóviles o bicicletas, en una clase, **vehículos**.

Este enfoque da como resultado que la declaración del problema a menudo esté mal definida, especialmente si hay varias instancias agrupadas en la misma clase. Por ejemplo, una imagen de una calle llena de gente podría segmentar toda el área de la multitud como la clase de **personas**. La segmentación semántica no proporciona detalles profundos en imágenes complejas como esta.

Segmentación de Instancias.

La segmentación de instancias implica la clasificación de píxeles en función de las instancias de un objeto (a diferencia de las clases de objetos). Los algoritmos de segmentación de instancias no saben a qué clase pertenece cada región; más bien, separan regiones similares o superpuestas en función de los límites de los objetos. [43]

Por ejemplo, suponga que un modelo de segmentación de instancias procesa una imagen de una calle llena de gente. Idealmente, debería ubicar objetos individuales dentro de la multitud, identificando el número de instancias en la imagen. Sin embargo, no puede predecir la región o el objeto para cada instancia.

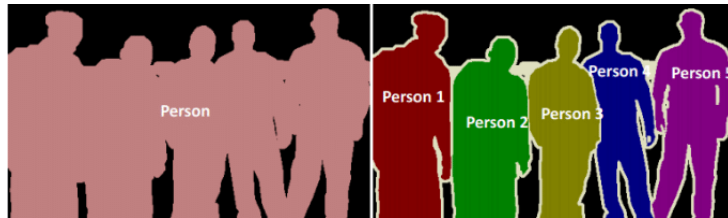


Figura 6: Ejemplo de segmentación semántica(izquierda) vs segmentación por instancias (derecha). [45]

Segmentación panóptica.

La segmentación panóptica es un tipo más nuevo de segmentación, a menudo expresada como una combinación de segmentación semántica y de instancias. Predice la identidad de cada objeto, segregando cada instancia de cada objeto en la imagen. [44]

La segmentación panóptica es útil para muchos productos que requieren grandes cantidades de información para realizar sus tareas. Por ejemplo, los automóviles autónomos deben poder capturar y comprender su entorno de manera rápida y precisa. Pueden lograr esto alimentando un flujo de imágenes en vivo a un algoritmo de segmentación panóptica.



Figura 7: Ejemplo de segmentación panóptica [46]

Las técnicas para realizar cualquier tipo de segmentación se engloban en dos clases; los algoritmos basados en técnicas de visión por computadora clásicos como: Thresholding, Clustering, Histogram-Based, Edge-Detection, Region-Growing, Watershed, etc. Y los algoritmos basados en algoritmos de Inteligencia Artificial ya sean de aprendizaje automático como los Random Forests o puramente de aprendizaje profundo como las Redes Neuronales Convolucionales.

3.2. Aprendizaje Automático y Aprendizaje Profundo.

El aprendizaje automático (Machine Learning) es un campo de investigación dedicado a comprender y crear métodos que **aprendan**, es decir, métodos que aprovechan los datos para mejorar el rendimiento en algún conjunto de tareas [47]. El aprendizaje automático se considera un subcampo de la inteligencia artificial. Los algoritmos de aprendizaje automático construyen un modelo basado en datos de muestra, conocidos como datos de entrenamiento, para hacer predicciones o tomar decisiones sin estar programados explícitamente para hacerlo [48]. Los algoritmos de aprendizaje automático se utilizan en una amplia variedad de aplicaciones, como: en medicina, filtrado de correo electrónico, reconocimiento de voz y visión artificial; donde es difícil o inviable desarrollar algoritmos convencionales para realizar las tareas necesarias [49].

Como esfuerzo científico, el aprendizaje automático surgió de la búsqueda de la inteligencia artificial. En los primeros días de la IA como disciplina académica, algunos investigadores estaban interesados en que las máquinas aprendieran de los datos. Intentaron abordar el problema con varios métodos simbólicos, así como lo que entonces se denominó **redes neuronales**; estos eran en su mayoría perceptrones y otros modelos que luego se descubrió que eran reinventaciones de los modelos lineales generalizados en estadística. [50] También se empleó el razonamiento probabilístico, especialmente en el diagnóstico médico automatizado. [51].

El aprendizaje automático, reorganizado como un campo separado, comenzó a florecer en la década de 1990. El campo cambió su objetivo de lograr inteligencia artificial a abordar problemas solucionables de naturaleza práctica. Cambió el enfoque lejos de los enfoques simbólicos que había heredado de la IA y hacia métodos y modelos tomados de estadística, la lógica difusa y la teoría de la probabilidad. [52].

La diferencia entre aprendizaje automático e Inteligencia Artificial a menudo se malinterpreta. el aprendizaje automático, aprende y predice en base a observaciones pasivas, mientras que la inteligencia artificial implica un agente que interactúa con el entorno para aprender y tomar acciones que maximicen sus posibilidades de lograr sus objetivos con éxito. [53]

A partir de 2020, muchas fuentes continúan afirmando que el aprendizaje automático sigue siendo un subcampo de Inteligencia artificial. [54] [55] [52]. Otros tienen la opinión de que no todo el aprendizaje automático es parte de la IA, sino que solo un **subconjunto inteligente** del aprendizaje automático debe considerarse IA.

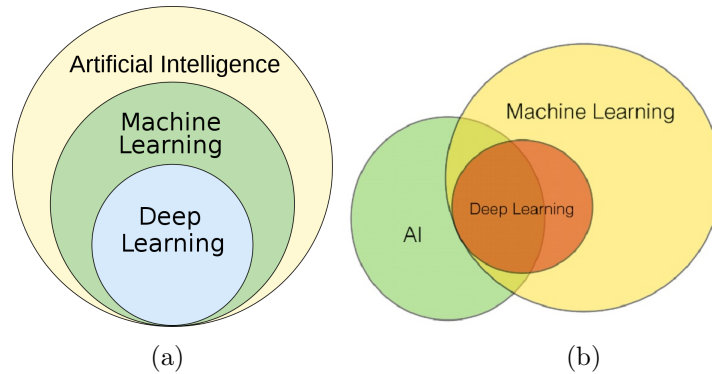


Figura 8: (a) Jerarquía tradicional [56] (b) y Jerarquía moderna de IA [57]

3.2.1. Enfoques del aprendizaje automático.

Los enfoques de aprendizaje automático se dividen tradicionalmente en tres categorías amplias, que corresponden a paradigmas de aprendizaje, según la naturaleza de la **señal** o **retroalimentación** disponible para el sistema de aprendizaje:

- **Aprendizaje supervisado:** a la computadora se le presentan entradas de ejemplo y sus salidas deseadas, dadas por un **maestro**, y el objetivo es aprender una regla general que asigna entradas a salidas.
- **Aprendizaje no supervisado:** no se asignan etiquetas al algoritmo de aprendizaje, dejándolo solo para encontrar una estructura en su entrada. El aprendizaje no supervisado puede ser un objetivo en sí mismo (descubrir patrones ocultos en los datos) o un medio para lograr un fin (aprendizaje de características).

- **Aprendizaje por refuerzo:** un programa de computadora interactúa con un entorno dinámico en el que debe realizar un objetivo determinado (como conducir un vehículo o jugar un juego contra un oponente). A medida que navega por su espacio problemático, el programa recibe retroalimentación que es análoga a las recompensas, que trata de maximizar. [58]

3.2.2. Modelos del aprendizaje automático.

Realizar el aprendizaje automático implica crear un **modelo**, que se entrena en algunos datos de entrenamiento y luego puede procesar datos adicionales para hacer predicciones. Se han utilizado e investigado varios tipos de modelos para los sistemas de aprendizaje automático:

- **Redes Neuronales Artificiales.**

Las redes neuronales artificiales (Artificial Neural Networks - ANN), o sistemas conexionistas, son sistemas informáticos vagamente inspirados en las redes neuronales biológicas que constituyen los cerebros de los animales. Dichos sistemas **aprenden** a realizar tareas considerando ejemplos, generalmente sin estar programados con ninguna regla específica de tareas.

Una ANN es un modelo basado en una colección de unidades o nodos conectados llamados **neuronas artificiales**, que modelan vagamente las neuronas en un cerebro biológico. Cada conexión, como las sinapsis en un cerebro biológico, puede transmitir información, una **señal**, de una neurona artificial a otra. Una neurona artificial que recibe una señal puede procesarla y luego enviar señales a otras neuronas artificiales conectadas a ella. En las implementaciones comunes de ANN, la señal en una conexión entre neuronas artificiales es un número real, y la salida de cada neurona artificial se calcula mediante alguna función no lineal de la suma de sus entradas. Las conexiones entre las neuronas artificiales se denominan **bordes**. Las neuronas artificiales y los bordes suelen tener un peso que se ajusta a medida que avanza el aprendizaje. El peso aumenta o disminuye la fuerza de la señal en una conexión. Las neuronas artificiales pueden tener un umbral tal que la señal solo

se envíe si la señal agregada cruza ese umbral. Por lo general, las neuronas artificiales se agregan en capas. Diferentes capas pueden realizar diferentes tipos de transformaciones en sus entradas. Las señales viajan desde la primera capa (la capa de entrada) hasta la última capa (la capa de salida), posiblemente después de atravesar las capas varias veces.

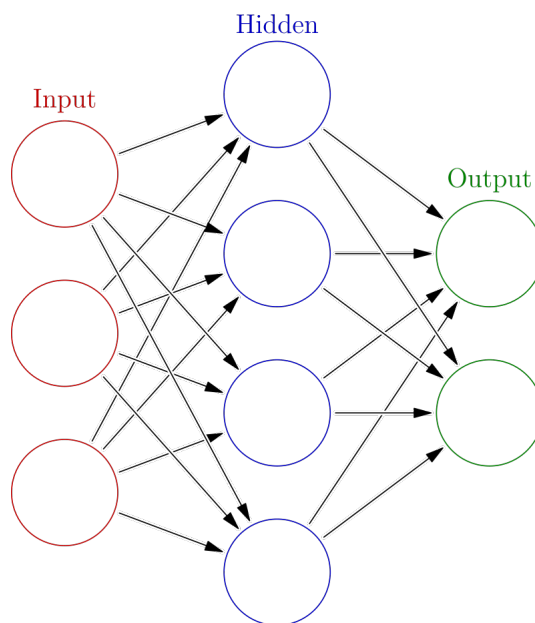


Figura 9: Cada nodo circular representa una neurona artificial y una flecha representa una conexión desde la salida de una neurona artificial a la entrada de otra. [59]

El objetivo original del enfoque ANN era resolver problemas de la misma manera que lo haría un cerebro humano. Sin embargo, con el tiempo, la atención se desplazó hacia la realización de tareas específicas, lo que provocó desviaciones de la biología. Las redes neuronales artificiales se han utilizado en una variedad de tareas, que incluyen visión por computadora, reconocimiento de voz, traducción automática, filtrado de redes sociales, juegos de mesa y videojuegos y diagnóstico médico.

El **aprendizaje profundo** consiste en múltiples capas ocultas en una red neuronal artificial. Este enfoque trata de modelar la forma en que el cerebro humano procesa la luz y el sonido en visión y audición. Algunas de las aplicaciones más exitosas del aprendizaje profundo son en los campos de la visión por computadora y el reconocimiento de voz. [60]

■ Árboles de decisión.

El aprendizaje del árbol de decisiones utiliza un árbol de decisiones como modelo predictivo para pasar de las observaciones sobre un elemento (representado en las ramas) a las conclusiones sobre el valor objetivo del elemento (representado en las hojas). Es uno de los enfoques de modelado predictivo utilizados en estadística, minería de datos y aprendizaje automático.

Los modelos de árbol en los que la variable de destino puede tomar un conjunto discreto de valores se denominan árboles de clasificación; en estas estructuras de árbol, las hojas representan etiquetas de clase y las ramas representan conjunciones de características que conducen a esas etiquetas de clase. Los árboles de decisión en los que la variable objetivo puede tomar valores continuos (normalmente números reales) se denominan árboles de regresión.

En el análisis de decisiones, se puede usar un árbol de decisiones para representar visual y explícitamente las decisiones y la toma de decisiones. En la minería de datos, un árbol de decisión describe los datos, pero el árbol de clasificación resultante puede ser una entrada para la toma de decisiones.

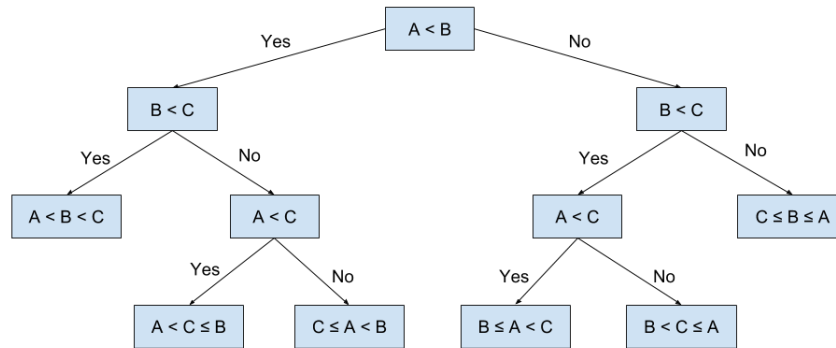


Figura 10: Ejemplo de árbol de decision. [61]

■ Máquinas de vectores de soporte.

Las máquinas de vectores de soporte (SVM), también conocidas como redes de vectores de soporte, son un conjunto de métodos de aprendizaje supervisado relacionados que se utilizan para la clasificación y la regresión.

Dado un conjunto de ejemplos de entrenamiento, cada uno marcado como perteneciente a una de dos categorías, un algoritmo de entrenamiento SVM construye un modelo que predice si un nuevo ejemplo cae en una categoría o en la otra. Un algoritmo de entrenamiento de SVM es un clasificador lineal, binario y no probabilístico, aunque existen métodos como el escalado de Platt para usar SVM en una configuración de clasificación probabilística. Además de realizar una clasificación lineal, las SVM pueden realizar de manera eficiente una clasificación no lineal utilizando lo que se denomina el truco del kernel, mapeando implícitamente sus entradas en espacios de características de alta dimensión.

clear

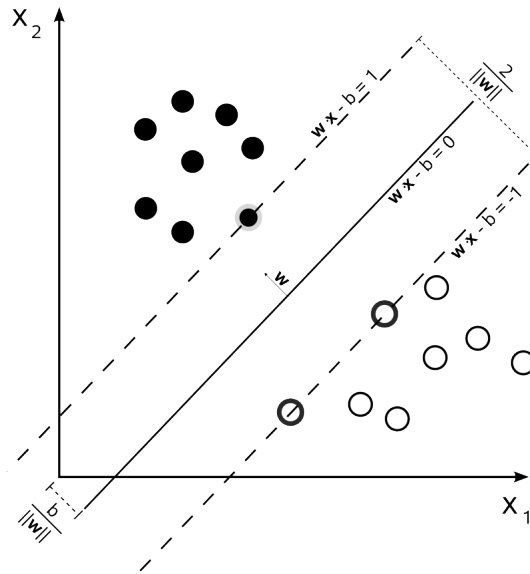


Figura 11: Una máquina de vectores de soporte es un modelo de aprendizaje supervisado que divide los datos en regiones separadas por un límite lineal. Aquí, el límite lineal divide los círculos negros de los blancos. [62]

■ Análisis de regresión.

El análisis de regresión abarca una gran variedad de métodos estadísticos para estimar la relación entre las variables de entrada y sus características asociadas. Su forma más común es la regresión lineal, en la que se dibuja una sola línea para que se ajuste mejor a los datos dados de acuerdo con un criterio matemático, como los mínimos cuadrados ordinarios. Este último a menudo se amplía mediante métodos de regularización para mitigar el sobreajuste y el sesgo, como en la regresión de cresta. Cuando se trata de problemas no lineales, los modelos de referencia incluyen la regresión polinomial, la regresión logística (a menudo utilizada en la clasificación estadística) o incluso la regresión kernel, que introduce la no linealidad. aprovechando el truco del kernel para mapear implícitamente las variables de entrada a un espacio de mayor dimensión.

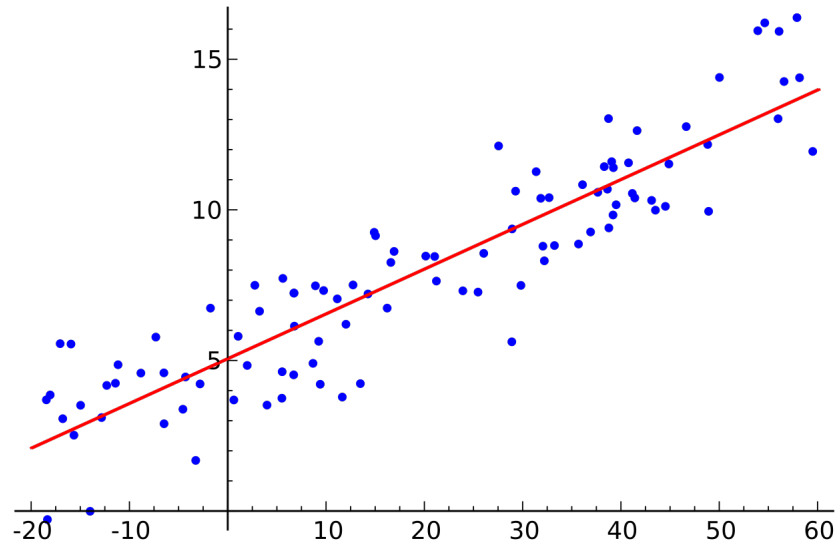


Figura 12: Ejemplo de una regresión lineal en ajustada a un conjunto de datos. [63]

■ Redes Bayesianas.

Una red bayesiana, red de creencias o modelo gráfico acíclico dirigido es un modelo gráfico probabilístico que representa un conjunto de variables aleatorias y su independencia condicional con un gráfico acíclico dirigido (DAG). Por ejemplo, una red bayesiana podría representar las relaciones probabilísticas entre enfermedades y síntomas. Dados los síntomas, la red se puede utilizar para calcular las probabilidades de la presencia de diversas enfermedades. Existen algoritmos eficientes que realizan inferencia y aprendizaje. Las redes bayesianas que modelan secuencias de variables, como señales de voz o secuencias de proteínas, se denominan redes bayesianas dinámicas. Las generalizaciones de las redes bayesianas que pueden representar y resolver problemas de decisión bajo incertidumbre se denominan diagramas de influencia.

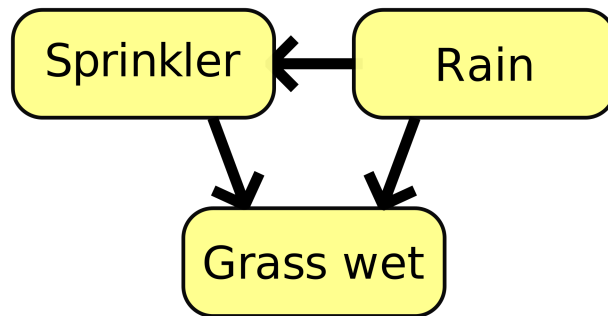


Figura 13: Una red Bayesiana simple. La lluvia (Rain) influye en la activación del aspersor (Sprinkler), y ambos factores; lluvia y aspersor influyen en el estado del pasto, es decir, si se encuentra mojado o no (Grass Wet). [64]

3.2.3. Aprendizaje Profundo.

El aprendizaje profundo (también conocido como aprendizaje estructurado profundo) es parte de una familia más amplia de métodos de aprendizaje automático basados en redes neuronales artificiales con aprendizaje de representación. El aprendizaje puede ser supervisado, semi-supervisado o no supervisado [65].

Las arquitecturas de aprendizaje profundo, como las redes neuronales profundas, las redes de creencias profundas, el aprendizaje de refuerzo profundo, las redes neuronales recurrentes, las redes neuronales convolucionales y los transformadores se han aplicado a campos que incluyen visión por computadora, reconocimiento de voz, procesamiento de lenguaje natural, traducción automática, bioinformática, diseño de fármacos, análisis de imágenes médicas, climatología, inspección de materiales y programas de juegos de mesa, en los que han producido resultados comparables y, en algunos casos, superiores al rendimiento de los expertos humanos. [66] [67] [68]

El adjetivo **profundo** en el aprendizaje profundo se refiere al uso de múltiples capas en la red. Los primeros trabajos mostraron que un perceptrón lineal no puede ser un clasificador universal, pero que una red con una función de activación no polinomial con una capa oculta de ancho ilimitado sí puede. El aprendizaje profundo es una variación moderna que se ocupa de un número ilimitado de capas de tamaño limitado, lo que permite una aplicación práctica

y una implementación optimizada, al tiempo que conserva la universalidad teórica en condiciones moderadas. En el aprendizaje profundo, también se permite que las capas sean heterogéneas y se desvíen ampliamente de los modelos conexionistas informados biológicamente, en aras de la eficiencia, la capacidad de entrenamiento y la comprensión, de ahí la parte **estructurada**.

3.2.4. Redes Neuronales Profundas.

Una red neuronal profunda (DNN) es una red neuronal artificial (ANN) con múltiples capas entre las capas de entrada y salida. [69] [70] Existen diferentes tipos de redes neuronales, pero siempre constan de los mismos componentes: neuronas, sinapsis, pesos, sesgos y funciones. [71]

Las DNN pueden modelar relaciones no lineales complejas. Las arquitecturas DNN generan modelos de composición en los que el objeto se expresa como una composición en capas de primitivas. [72] Las capas adicionales permiten la composición de características de las capas inferiores, lo que potencialmente modela datos complejos con menos unidades que una red superficial de rendimiento similar. [73][9] Por ejemplo, se demostró que los polinomios multivariantes dispersos son exponencialmente más fáciles de aproximar con DNN que con redes poco profundas. [74]

Las arquitecturas profundas incluyen muchas variantes de algunos enfoques básicos. Cada arquitectura ha tenido éxito en dominios específicos. No siempre es posible comparar el rendimiento de varias arquitecturas, a menos que se hayan evaluado en los mismos conjuntos de datos.

Las DNN suelen ser redes de retroalimentación en las que los datos fluyen desde la capa de entrada a la capa de salida sin retroceder. Al principio, la DNN crea un mapa de neuronas virtuales y asigna valores numéricos aleatorios, o **pesos**, a las conexiones entre ellas. Los pesos y las entradas se multiplican y devuelven una salida entre 0 y 1. Si la red no reconociera con precisión un patrón en particular, un algoritmo ajustaría los pesos. [75] De esa manera, el algoritmo puede hacer que ciertos parámetros sean más influyentes, hasta que determine la manipulación matemática correcta para procesar completamente los datos.

Las redes neuronales recurrentes (RNN), en las que los datos pueden fluir en cualquier dirección, se utilizan para aplicaciones como el modelado de lenguaje [76] [77] [78]. La memoria a largo plazo es particularmente eficaz para este uso. [79] [80]

Las redes neuronales profundas convolucionales (CNN) se utilizan principalmente en el área de visión por computadora y en el reconocimiento de voz. Grandes avances se han logrado en tareas de clasificación de imágenes, segmentación, detección de objetos y visión stereo. [81] [82]

3.3. Estado del Arte.

El conteo de calorías es uno de los dos componentes principales en casi todas las pautas de salud emitidas. Las investigaciones han demostrado que restringir las calorías ayuda a reducir el peso y aumentar los indicadores de salud, como la presión arterial, la resistencia a la insulina, los niveles de colesterol, la salud del hígado, etc. [83]. El otro componente importante es la densidad de nutrientes de los alimentos, es bien sabido que el cuerpo necesita ciertos nutrientes (vitaminas, minerales y fibra) y macronutrientes (proteínas, carbohidratos y grasas) para estar saludable y que estos nutrientes y macronutrientes solo están disponibles en los alimentos y no se pueden obtener en ningún otro lugar [84].

Para seguir los componentes principales de las pautas, es decir una dieta; está compuesta por un número recomendado de calorías en función de la edad, sexo, peso actual y actividad. A su vez esta compuesta por varios niveles de micro y macro nutrientes, medidos en gramos, miligramos o microgramos según el nutriente. Por ejemplo, un hombre de 25 años con una actividad promedio de 30 minutos al día podría necesitar una dieta de 2500 calorías, 1 gramo de proteína por kilo de peso corporal, 30 gramos de carbohidratos, 10 miligramos de zinc, 30 miligramos de manganeso, etc.

Con el ejemplo anterior como objetivo en mente, hay innumerables formas de alcanzar los niveles descritos y es muy fácil sobrepasar esos límites en cuanto a calorías y estar en el lado bajo en cuanto a nutrientes. Para cumplir con lo recomendado, la persona primero debe conocer la composición de muchos alimentos y, en segundo lugar, debe saber exactamente la cantidad

que está comiendo, ambos aspectos son difíciles y poco prácticos de medir en un estilo de vida moderno.

Si hubiera una manera de reconocer, acceder y medir fácilmente la composición de los alimentos sin tener que pesarlos, medirlos ó clasificarlos, significaría un gran paso para lograr una mejor salud pública en todo el mundo. En este trabajo, creemos que la visión por computadora combinada con algoritmos de aprendizaje automático y aprendizaje profundo se puede lograr el objetivo simplemente tomando una foto (con un teléfono celular o cualquier dispositivo similar) de la comida que está a punto de ser ingerida.

3.3.1. Estado del Arte Enfoques no móviles.

Actualmente no hay muchos trabajos sobre el tema en cuestión, sin embargo, el gigante tecnológico Google ya ha comenzado a investigar y desarrollar en este tema con su científico líder Kevin Murphy. Ya han desarrollado el software **Im2calories** una aplicación que puede contar calorías a partir de una foto de baja resolución tomada con un celular de un plato [85], solo funciona en el 30 por ciento de los casos pero lo están mejorando con aportes de personas desde 2015. El modelo no está abierto al público, pero se sabe que se basa en una red neuronal convolucional.

Yanchao Liang y Jianhua Li de la Universidad de Ciencia y Tecnología de China Oriental en 2017 [86], crearon la base de datos **ECUSTD** compuesta por 2987 imágenes de 20 categorías de alimentos, también introdujeron una arquitectura CNN para clasificación de imágenes y segmentación de alimentos. Esta arquitectura podría servir como base para la implementación de todo el proceso de estimación calórica, sin embargo, los autores no lo han desarrollado hasta el final.

Simon Mezgec y Barbara Seljak del departamento de Tecnologías de la Información y la Comunicación de la Universidad de Eslovenia desarrollaron en 2017 la **NutriNet**, una arquitectura de CNN que afirman tener un 94 % de precisión en algunos conjuntos de datos de alimentos, dicha precisión sólo es para la tarea de clasificar los alimentos no para estimar su composición calórica. [87]

Xioweng Wu et. al. del Living Analytics Research Center de la Universidad de Singapur, compilaron una base de datos; la **FoodSeg103** conformada de 9490 imágenes con 154 clases y máscaras de segmentación [88]. Además, propusieron un enfoque de preentrenamiento multimodal llamado ReLeM que dota al modelo de conocimiento semántico. Realizaron tres experimentos de segmentación: basado en la convolución dilatada, basado en la pirámide de características y basado en la transformación de la visión. La base y los modelos que propusieron están destinados a funcionar como un punto de referencia para trabajos futuros.

Se puede concluir a partir de los trabajos citados anteriormente y algunos otros no mencionados que el problema del reconocimiento de alimentos es un desafío debido a la naturaleza de los alimentos. Los alimentos son objetos deformables, lo que dificulta el proceso de definición de su estructura. Además, algunos tipos de alimentos pueden tener una variación intraclase alta y una variación baja entre clases. Todos los trabajos modernos de investigación de potencias tecnológicas como Estados Unidos, China, Singapur y Europa parecen indicar que un enfoque de aprendizaje profundo puede ser muy prometedor en el campo del reconocimiento de imágenes de alimentos.

Menos aún ninguno de los trabajos revisados hasta ahora ha tratado de estimar el valor nutricional además de la estimación de calorías, por lo que parece que hay mucho margen de mejora y el problema no puede considerarse resuelto todavía.

3.3.2. Estado del Arte Enfoques móviles.

El primer algoritmo de detección de alimentos, que se implementó para uso práctico en un dispositivo móvil, se describió en el artículo de Joutou en el año 2009 [89]. Le siguieron nuevos enfoques para reducir la cantidad de conocimiento y hardware necesarios para el análisis de la alimentación de los usuarios.

Todas las aplicaciones revisadas a continuación usan teléfonos móviles para registrar actividades de alimentación a través de imágenes. Para detectar automáticamente qué objetos de comida se comieron, se necesita un algoritmo de reconocimiento de objetos de comida. Los algoritmos de reconocimiento de comida se componen principalmente de los siguientes pasos o etapas: adquisición de imágenes, procesamiento de imágenes, segmentación de imágenes, extracción de características de imágenes y clasificación de imágenes.

La adquisición de imágenes trata con diferentes técnicas de captura de datos (algunas aplicaciones usan una imagen, algunas usan múltiples imágenes, algunas usan OCR o datos de voz), mientras que la etapa de procesamiento trata con diferentes técnicas de normalización de píxeles para la reducción del ruido presente. Estas dos etapas son necesarias para una mejor segmentación de imágenes, donde se extraen ROIs - regiones de interés. Las regiones de interés (generalmente el fondo, la sustracción del primer plano) permiten acortar el tiempo de procesamiento de la etapa posterior de extracción de características, ya que el área de análisis es más pequeña y específica del objeto. La siguiente etapa es extraer diferentes características de las imágenes segmentadas. Se pueden utilizar muchas técnicas de características de imagen diferentes. Estas características se utilizan posteriormente para la clasificación. Una vez que se realiza la clasificación, el algoritmo genera la clase a la que pertenece el conjunto de características ingresado. Después de la salida, está en manos del desarrollador front-end mostrar los resultados en la forma o forma deseada (análisis calórico, recomendación para otras actividades de alimentación, sugerencias de recetas, etc.). Debido a la diferencia de disponibilidad de tecnología y algoritmos de detección de objetos, las aplicaciones móviles de detección de alimentos difieren a lo largo de los años en la mayoría de las etapas. Si bien algunos aspectos de una etapa en particular coexisten, uno más nuevo y mejor reemplaza a los antiguos o se agrega a la mezcla.

3.3.3. Joutout y Yanai, 2009.

Según los autores, este es el primer sistema de detección de alimentos, que demostró implicaciones prácticas en un teléfono móvil. El futuro principal del algoritmo de detección de alimentos implementado es la clasificación de aprendizaje de múltiples núcleos (MKL). En términos simples, el algoritmo MKL clasifica los objetos en función de las características y sus pesos correspondientes. Esto significa que la importancia de una característica particular de la imagen cambia dinámicamente, según la categoría de alimentos. Esto va muy bien con los objetos de comida, especialmente cuando pasamos de la simple detección de frutas o verduras a la detección de comidas complejas. En el año siguiente (2010) este sistema se amplió para reconocer 85 categorías de alimentos diferentes [90]. El objetivo de los autores era mantener la tasa de éxito de clasificación del sistema anterior. Para esto, se agregaron ocho características adicionales de HoG (histograma de gradientes orientados).

3.3.4. Puri et al., 2009.

En el artículo de Puri et al. [91], los autores describieron y propusieron un sistema de estimación de volumen y artículos alimentarios. En el artículo, los autores reconocen que la evaluación de los alimentos es un procedimiento muy difícil incluso para los expertos en nutrición. La dificultad radica especialmente en platos donde se ocultan algunos nutrientes. Por lo tanto, un sistema basado en algoritmos de detección de alimentos y volumen necesita tanta información como sea posible. Un método basado en un reconocedor visual y de voz de ingesta de alimentos. Se propuso el sistema FIVR. Luego, el sistema se implementa utilizando un dispositivo móvil y, por lo tanto, utiliza imágenes y datos de voz para el reconocimiento de alimentos y la estimación del volumen.

El sistema está creado para reconocer múltiples artículos en un plato. Todos los algoritmos de segmentación y clasificación de imágenes se ejecutan en un servidor. El teléfono móvil se utiliza únicamente para la grabación de datos de voz e imagen.

La arquitectura del sistema: La característica principal de la aplicación es que permite reconocer alimentos y estimar su volumen. La arquitectu-

ra del sistema es el siguiente: el usuario toma inicialmente tres fotografías de una comida. Luego, los usuarios acompañan estas imágenes con datos de voz para crear etiquetas de alimentos para los artículos en el plato. Estos datos luego se envían al servidor. La comida se identifica en base a los datos del habla. Para cada objeto de comida identificado, sigue una reconstrucción 3D para estimar su volumen. Después de la estimación del volumen, los datos de los alimentos y sus volúmenes correspondientes se almacenan en el servidor y se envían al usuario a través de un mensaje de texto. El mensaje incluye datos nutricionales de cada artículo.

3.3.5. Kong y Tan, 2012.

En el artículo de Kong y Tan [92] se presentó la aplicación para teléfonos móviles DietCam. Similar al sistema de Puri et al. [91] este sistema se utiliza para detección y estimación de volumen de alimentos. La principal diferencia en el caso de DietCam es que ejecuta ambas funciones solo en base a imágenes. Además, la aplicación móvil utiliza el acelerómetro para calcular el ángulo del dispositivo, lo que resulta útil para el enfoque de múltiples vistas de la grabación de imágenes. Además de imágenes regulares de alimentos, técnica de reconocimiento óptico de caracteres (OCR) se utiliza para leer las etiquetas de los alimentos.

La arquitectura del sistema: Se puede describir de la siguiente manera: primero, los usuarios capturan 3 imágenes de diferentes poses de elementos o un video alrededor el artículo de comida. Luego la información es enviada en formato XML al servidor, donde se realizan todos los procesos de procesamiento de imágenes, extracción de características, clasificación y un 3D se ejecuta la reconstrucción para la estimación del volumen de alimentos. Primero, el administrador de imágenes extrae las características importantes, que se utilizan para la clasificación.

La clasificación se ejecuta primero utilizando una base de datos local, donde se almacenan las instancias de las imágenes de los usuarios. Si ninguna de las instancias almacenadas se puede utilizar para clasificar el alimento registrado, entonces se utiliza la base de datos global de todas las imágenes de alimentos. Este proceso de dos pasos permite una clasificación más rápida y eficaz de los alimentos. Después de la clasificación, el volumen de

los alimentos se estima a través de la reconstrucción 3D. El sistema también tiene la posibilidad de extraer los residuos de la comida, que luego se utiliza para el total de calorías consumidas. Después de esta etapa, las calorías y los alimentos consumidos se envían al teléfono móvil y se almacenan en la base de datos.

3.3.6. Rahman et al., 2012.

En el artículo de Rahman, Pickering y Kerr [93] se propuso una nueva función de textura que aumentaría la precisión del reconocimiento de alimentos en una plataforma móvil. La característica de textura propuesta se basaría en los bancos de filtros Gabor, que producirían características de textura globales invariantes a escala y rotación. La principal diferencia con otras aplicaciones de consultoría dietética es que utiliza múltiples escalas e imágenes de orientación para los alimentos. Esto permite una mejor detección de alimentos en poses diferentes. Para la implementación de esta nueva característica de textura, se utilizó la Evaluación Dietética Asistida por Tecnología (TADA). El sistema ejecuta la detección y estimación de volumen de comida.

La arquitectura del sistema: TADA se basa en una interacción entre un dispositivo móvil para la adquisición de imágenes y un servidor para el procesamiento de imágenes, segmentación, extracción de características, clasificación y estimación de volumen. El usuario comienza tomando una foto de un alimento, que se acompaña con un marcador de calibración de color. Además de la foto, el usuario también envía algunos datos adicionales, como la fecha, la hora y la geolocalización. Luego, esta información se envía al servidor donde se ejecuta la identificación de alimentos y la estimación del volumen (según los procesos subyacentes). Los resultados de la identificación de alimentos y estimación de volumen son enviados posteriormente al usuario, para su aprobación y posibles ajustes. Después de la confirmación, la actividad de alimentación se almacena en el servidor, según la base de datos nutricional (que también se encuentra en el servidor). Finalmente, los resultados de los cálculos de nutrientes se envían a un investigador para su posterior análisis, que en trabajos futuros incorporaría los comentarios de los usuarios para el análisis de grupos de alimentos y recomendaciones dietéticas.

3.3.7. Kawano y Yanai, 2013.

Kawano y Yanai han propuesto un sistema ligero para el reconocimiento de alimentos, en el que todo el procesamiento se ejecuta en el dispositivo móvil [94]. Esta es la primera aplicación que lo hace. El propósito de ejecutar todas las etapas de detección de alimentos localmente en el teléfono móvil fue, para eliminar los retrasos y costes entre el cliente y el servidor, como consecuencia de la transmisión de datos. Para acelerar aún más, el usuario puede dibujar cuadros delimitadores alrededor de los alimentos, lo que permite una rápida segmentación de la imagen. Debido a que la detección de alimentos se ejecuta en tiempo real, el usuario puede ajustar los ángulos de la cámara si el algoritmo de detección de alimentos no detecta el alimento correcto. Esto se hace a través de una característica adicional, que propone el ángulo de la cámara, donde más información sobre la comida debe ser visible. El sistema no estima automáticamente el volumen de los artículos, pero permite al usuario indicar el volumen con un control deslizante en la pantalla. En el mismo año (2013) se optimizó el sistema reduciendo el tiempo de detección de alimentos dentro de los cuadros delimitadores [95]. Esto se ha logrado con éxito utilizando vectores de Fisher [96] y diferentes características de imagen. La versión optimizada tuvo un mejor tiempo de procesamiento de reconocimiento de alimentos (de 0,26 segundos a 0,065 segundos) y una tasa de éxito ligeramente mejor al duplicar el número de categorías (de 50 a 100). En 2015 se implantaron estos dos sistemas de forma en la plataforma android para uso público [97]

La arquitectura del sistema en esta aplicación es la siguiente. Primero, el usuario apunta con su cámara al alimento deseado. Debido a que la detección de alimentos se ejecuta en tiempo real, la aplicación recopila marcos continuamente. A continuación, el usuario dibuja cuadros delimitadores alrededor de los alimentos, lo que permite una segmentación rápida. Después de que el usuario haya creado ROI, se ejecuta la detección de alimentos en cada uno de ellos. Los resultados del algoritmo de detección de alimentos es una lista de los 5 mejores candidatos y la dirección de la región de los alimentos. La dirección se muestra como una flecha en la pantalla. Después de la lista de candidatos devuelta, el usuario tiene dos opciones. Si el alimento correcto está en la lista, entonces el usuario puede ajustar el volumen estimado del alimento y seleccionar el alimento (tocando el elemento de la lista). Si la lista de los 5 candidatos principales no incluye el alimento correcto, el

usuario mueve el teléfono móvil en la dirección propuesta indicada por la flecha. Al hacerlo, se vuelve a ejecutar el algoritmo de detección de alimentos. Este ajuste continúa hasta que se detecta el alimento deseado. Después de la selección del alimento deseado, el alimento y la información nutricional se muestran en la pantalla. Además, el usuario puede guardar los registros de comidas en un servidor. Posteriormente puede acceder a esta información en la Web.

3.3.8. Anthimopoulos et al., 2013.

Se propuso un sistema para el conteo de carbohidratos [98] y estaba dirigido especialmente para ayudar a los pacientes con diabetes tipo 1. La clasificación de las imágenes se basa en seis categorías diferentes: carne, empanados, arroz, pasta, papas y verduras, por lo que no es específica de un alimento. El objetivo es separar e identificar regiones de los grupos de alimentos antes mencionados en un plato. La propiedad especial de este sistema es la complejidad de la segmentación de alimentos que se logra con diferentes algoritmos.

La arquitectura del sistema comienza con una foto tomada por la cámara del teléfono móvil. Más tarde, la imagen grabada se segmenta y es reconocida como una de las seis categorías de alimentos. En función de la segmentación y la categoría de alimentos clasificados, el sistema puede devolver el valor estimado de carbohidratos y la dosis de insulina correspondiente, según los datos del paciente.

3.3.9. Pouladzadeh, Shirmohammadi y Arici, 2013.

En el artículo de Pouladzadeh [99] se proponía un sistema de medición automática de alimentos. Además de la detección de alimentos, el objetivo del sistema es estimar la calorías consumidas, utilizando imágenes de antes y después de la comida. El sistema está basado en el cliente, es decir, todas las etapas de detección de alimentos se ejecutan en el dispositivo móvil.

La arquitectura del sistema es la siguiente: el usuario registra dos imágenes de la comida; uno de arriba y otro de lado. Al tomar la foto desde arriba, el usuario debe acompañar la imagen con su pulgar, que luego se usará para estimar el volumen. Una de las imágenes es procesada y segmentada.

Después de esto, se extraen diferentes características para cada alimento segmentado. Las características extraídas se utilizan luego para una clasificación correcta. La aplicación envía al usuario la información sobre la clasificación para la aprobación. A continuación, sigue la estimación del área de alimentos. Estimación de área se produce utilizando ambas imágenes del alimento, con la ayuda del pulgar. Con la estimación del área y las etiquetas nutricionales, el consumo calórico total es calculado. Si el usuario no termina toda la comida, luego puede grabar una imagen de las sobras. El volumen de las sobras se resta de la volumen de la comida inicial

4. Propuesta de diseño.

Como se ha visto la dificultad más grande que presentan la mayoría de los modelos modernos de estimación calórica, es el tratamiento de la generalización de las clases de alimentos; en otras palabras, el modelo será capaz de hacer una estimación de un alimento en específico si este último fue incluido en el conjunto de imágenes de entrenamiento, de lo contrario la estimación será totalmente aleatoria. Otra desventaja que presentan los modelos que colectan más de una imagen son los errores que se acarrearán al tratar de corregir la disparidad de las imágenes para crear un modelo de visión stereo sólido. Además justamente para estos modelos se necesita una arquitectura de visión stereo física (dos cámaras de las mismas características, calibradas y a una distancia suficiente), lo que los hace más difíciles de migrar al mundo de los teléfonos móviles.

La forma más directa de eliminar el problema de generalización de clases, es eliminar la clasificación del proceso de estimación calórica y sustituirlo por retroalimentación humana, en el caso del presente trabajo se optó por el texto aunque bien podría ser voz.

La eliminación de la clasificación, además, acarrea otro beneficio; la estimación se puede personalizar a distintos tipos de dietas tan variadas como sean posibles, con el bajo costo de crear una base de conocimiento.

El problema de las múltiples fotografías se puede sustituir con estimaciones basadas en áreas y para poder realizar esta sustitución se requiere otra

retroalimentación humana muy sencilla; proporcionar el diámetro del plato donde estan contenidos los alimentos. Con esta sencilla retroalimentación se pueden realizar cálculos de áreas y se puede estimar calorías con dichas áreas.

En este trabajo se propone sustituir los dos problemas fundamentales del proceso de estimación de calorías por las soluciones expuestas en los párrafos anteriores, de tal forma que el proceso quedaría de la siguiente forma:

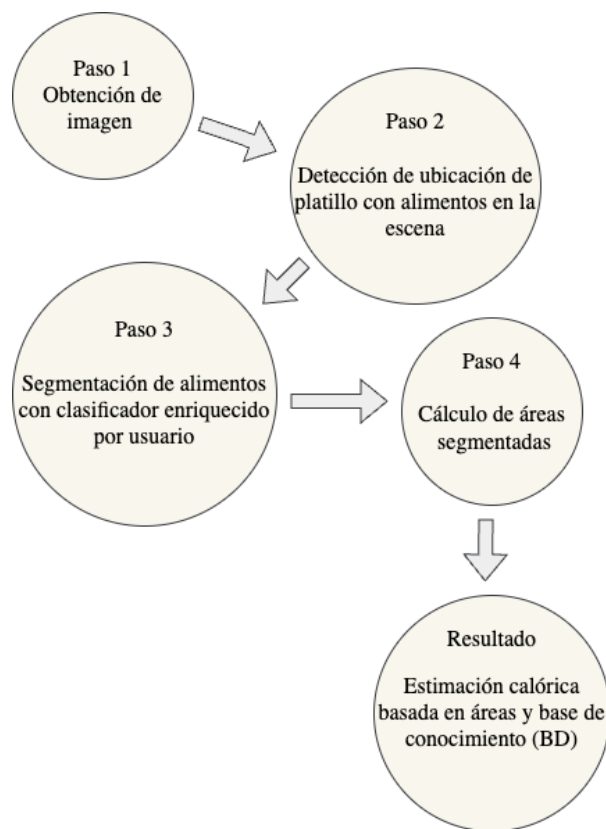


Figura 14: Propuesta de Pipeline para proceso de estimación calórica basada en imagen.

En la figura 14 se muestra el proceso propuesto en este trabajo para resolver el problema de estimación calórica de alimentos basados en su imagen.

4.1. Captura de imagen digital.

El **primer paso** consiste en obtener una imagen digital a través de cualquier dispositivo que capture imágenes digitales. La imagen debe contener en la escena un platillo con alimentos, el plato contenedor deberá ser un plato circular y la foto deberá tomarse desde una perspectiva superior.



Figura 15: Paso 1: Captura de imagen Digital.

4.2. Detección de objeto.

El **segundo paso** es un paso completamente automático, un algoritmo de aprendizaje profundo detecta en que parte de la escena se encuentra el plato que contiene los alimentos y dibuja un **cuadro delimitador** en la ubicación del platillo.

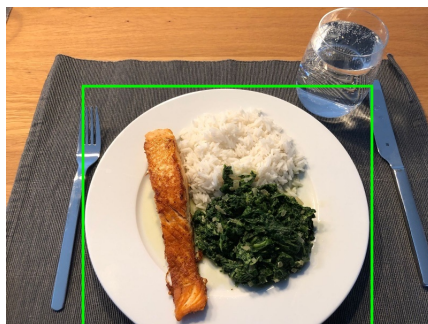


Figura 16: Paso 2: Detección del platillo con alimentos en la escena.

4.3. Segmentación.

En el **tercer paso** se segmentan cada uno de los alimentos, el plato contenedor y el fondo de la subimagen dentro del cuadro delimitador que se obtiene en el paso anterior. La segmentación utiliza un algoritmo de aprendizaje automático basado en un bosque aleatorio (random forest), cuyas variables las conforman características de textura e intensidad de píxeles.



Figura 17: Paso 3: Segmentación de la región de interés.

4.4. Cálculo de áreas segmentadas.

En el **cuarto paso**, con las áreas segmentadas del paso anterior se pueden obtener áreas en centímetros cuadrados, gracias al conocimiento del perímetro del plato contenedor, dato que es proporcionado por el usuario. El usuario también proporciona la clase a la que pertenece cada área. El área del plato se podría obtener de forma automática si se tuviera otro objeto en la imagen como referencia, de dicho objeto se tendrían que saber sus dimensiones y además el modelo de detección también debería detectar exactamente donde se encuentra. Es por estas razones que se rechazó la idea.

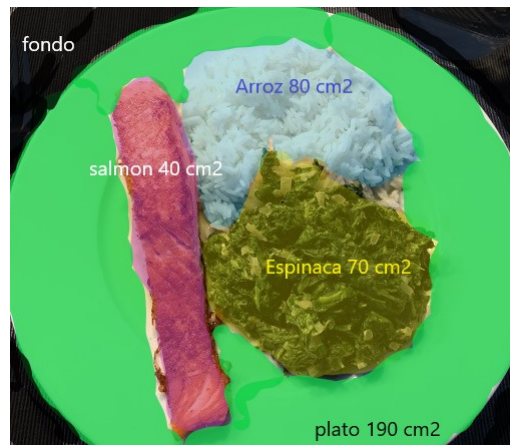


Figura 18: Paso 4: Cálculo de áreas de regiones segmentadas y clasificación.

4.5. Cálculo de calorías.

El último y **quinto paso**, consiste en realizar el cálculo de las calorías basadas en el área ocupada en centímetros cuadrados y la clase de alimento que se trata, este cálculo es obtenido de una base de datos que contiene una base de conocimiento previo, con clases de alimentos, con sus respectivas calorías por centímetro cuadrado. En el ejemplo anterior, si como resultado se obtuvo 40 centímetros cuadrados de salmón, en la base de datos estaría guardado que en promedio 40 centímetros cuadrados de salmón representa 100 gramos de salmón y aporta 208 calorías. El mismo ejercicio se realiza para el arroz y las espinacas. La calorías de cada clase de alimento se suman y se obtiene el total de calorías por platillo.

5. Pruebas y resultados.

5.1. Prototipado - Clasificación de imágenes usando Aprendizaje Profundo.

Dados los prometedores resultados de otros trabajos y el hecho de que las redes neuronales convolucionales se usan para resolver problemas complejos [100], se decidió probar las conocidas arquitecturas de CNN en conjuntos de datos de alimentos para comprender la clasificación de imágenes de alimentos.

Se utilizaron principalmente dos conjuntos de datos: el conjunto de datos Food 101 formado por 101,000 imágenes de 101 clases de alimentos recopiladas por Bossard et al. por su trabajo en Minería de componentes Discriminativos con bosques aleatorios. Los platos presentados en este conjunto de datos pueden ser tan simples como un pastel de manzana o complejos como una ensalada César. El otro conjunto de datos que se utilizó es el conjunto de datos ECUSTD recopilado por Yanchao Liang y Jianhua Li de la Universidad de Ciencia y Tecnología de China Oriental. Este conjunto de datos está formado por 2987 imágenes de 20 clases de alimentos, las clases son simples como manzana, plátano, uvas, mango, rosquilla, etc. Solo hay una clase etiquetada como mezcla que incluye dos alimentos de dos clases diferentes. Las imágenes en ambos conjuntos de datos se trataron para ajustarse al tamaño de 224 x 224 píxeles, que es el tamaño utilizado en las arquitecturas CNN más populares creadas para ImageNet Challenge.

El tratamiento utilizado para cambiar el tamaño de las imágenes fue una interpolación bicúbica sobre una vecindad de 4x4 píxeles. El lenguaje utilizado fue Python con la librería cv2. [?]

Cuadro 1: Conjuntos de datos utilizados.

	Imágenes	Clases	Tamaño Original	Tamaño Tratado
Food 101	101,000	101	512 x 384	224 x 224
ECUSTD	2,987	20	816 x 551	224 x 224

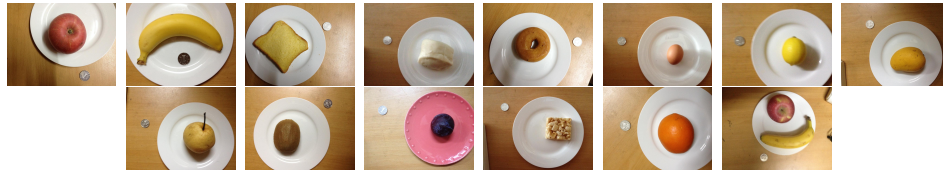


Figura 19: Ejemplos de elementos de ECUSTD. De izquierda a derecha de arriba a abajo: manzana, plátano, pan, bollo, rosquilla, huevo, limón, mango, pera, qivi, ciruela, sachiman, naranja, mezcla.

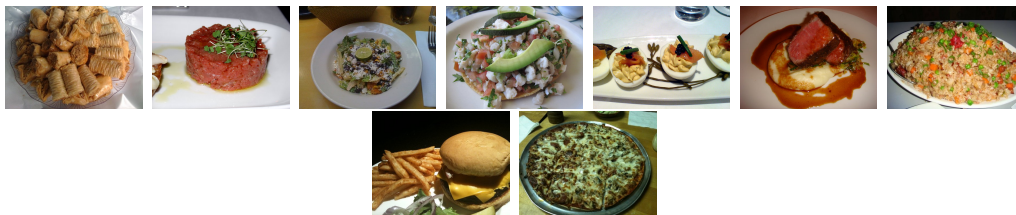


Figura 20: Ejemplos de elementos de Food 101. De izquierda a derecha y de arriba a abajo: baklava, tartar de ternera, ensalada César, ceviche, huevos rellenos, filet mignon, arroz frito, hamburguesa, pizza.

Se eligieron 3 arquitecturas para probar la clasificación de alimentos: MobileNet, GoogLeNet y VGG16. La decisión detrás de probar GoogLeNet parece obvia; el modelo tiene el recuento de parámetros más bajo de las arquitecturas modernas junto con la gran precisión obtenida en el desafío ImageNet. Por otro lado elegimos el modelo VGG16 por la sencillez y elegancia de la arquitectura combinada con su bajo error.

AlexNet y ZFNet se dejaron de lado debido a la gran cantidad de parámetros en el modelo junto con una mayor tasa de error. El modelo ResNet también se dejó de lado, esta vez la decisión se debió al hecho de que el modelo tiene 152 capas y el tamaño de las imágenes es difícil de ajustar al modelo, por lo que se debería usar el modelo ResNet18, que tiene una precisión menor que GoogLeNet. y VGG16.

GoogLeNet y VGG16 ocuparon el primer y segundo lugar respectivamente en el Desafío de reconocimiento visual a gran escala de ImageNet en 2014.

GoogLeNet tiene 22 capas y VGG 16 tiene 19 capas. Ambas arquitecturas son grandes y lentas de entrenar, pero rápidas de usar en predicciones una vez que finaliza el entrenamiento. La Cuadro 4 muestra una comparación de varios campeones del desafío ImageNet y sus características.

Cuadro 2: Arquitectura GoogLeNet [101].

Operator	Input
conv1	3 x 224 x 224
pool1	64 x 112 x 112
conv2	64 x 56 x 56
pool2	192 x 56 x 56
inception3a	192 x 28 x 28
inception3b	256 x 28 x 28
pool3	480 x 28 x 28
inception4a	480 x 14 x 14
inception4b	512 x 14 x 14
inception4c	512 x 14 x 14
inception4d	512 x 14 x 14
inception4e	512 x 14 x 14
pool4	832 x 14 x 14
inception5a	832 x 7 x 7
inception5b	832 x 7 x 7
pool5	1024 x 7 x 7
fc	1024 x 1 x 1
prob	1000

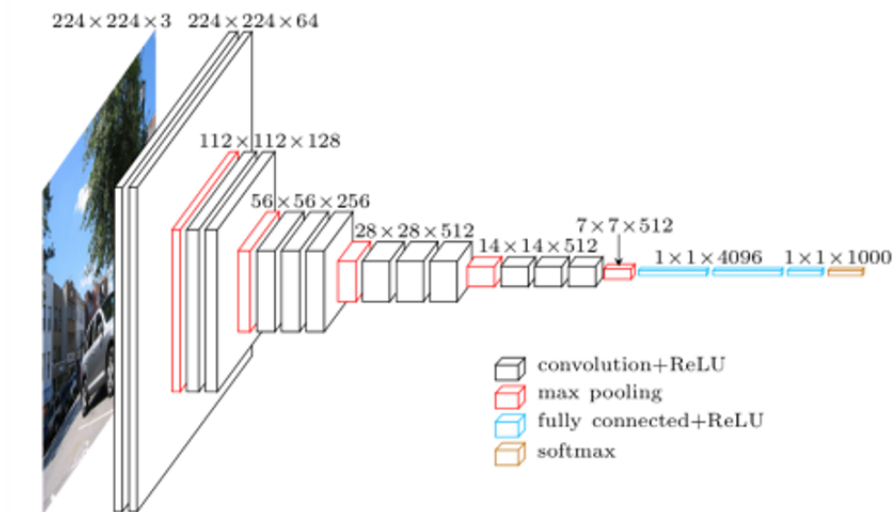


Figura 21: Arquitectura - VGG16 [102].

MobileNet también fue elegido para probar la precisión de la tarea de clasificación. MobileNet es parte de una familia de MobileNets diseñada también por Google para maximizar efectivamente la precisión mientras se tiene en cuenta los recursos restringidos para una aplicación integrada o en el dispositivo.

MobileNets son modelos pequeños, de baja latencia y bajo consumo de energía parametrizados para cumplir con las limitaciones de recursos de una variedad de casos de uso. Se pueden utilizar para clasificación, detección, incrustaciones y segmentación.

Cuadro 3: Arquitectura MobileNet [103].

Operator/Stride	Input
Conv / s2	3 x 224 x 224
Conv dw / s1	32 x 112 x 112
Conv / s1	32 x 112 x 112
Conv dw / s2	64 x 112 x 112
Conv / s1	64 x 56 x 56
Conv dw / s1	128 x 56 x 56
Conv / s1	128 x 56 x 56
Conv dw / s2	128 x 56 x 56
Conv / s1	128 x 28 x 28
Conv dw / s1	256 x 28 x 28
Conv / s1	256 x 28 x 28
Conv dw / s2	256 x 28 x 28
Conv / s1	256 x 14 x 14
5 X (Conv dw / s1 , Conv / s1)	512 x 14 x 14
Conv dw / s2	512 x 14 x 14
Conv / s1	512 x 7 x 7
Conv dw / s2	1024 x 7 x 7
Conv / s1	1024 x 7 x 7
Avg Pool / s1	1024 x 7 x 7
FC / s1	1024 x 1 x 1
Softmax / s1	1000 x 1 x 1

Cuadro 4: Comparación de los campeones del ImageNet Challenge.

año	CNN	Autor	Lugar	Error	Params.
1998	LeNet	Yann LeCun et.al.	1ero		60k
2012	AlexNet	Alex krizhenvsky et.al.	1ero	15.3 %	60M
2013	ZFNet	Zeiler y Fergus	1ero	14.8 %	60M
2014	GoogLeNet	Google	1ero	6.67 %	4M
2014	VGG16	Simonyan y Zisserman	2do	7.3 %	138M
2015	ResNet(152)	Kaiming He	1ero	3.6 %	60M

Fuente: Elaboración propia basado en los resultados del ImageNet Challenge.

Se hicieron algunos cambios en las capas completamente conectadas de las tres arquitecturas antes de entrenarlas con los conjuntos de datos de imágenes de alimentos. Para VGG16, se cambió la cantidad de neuronas en las capas completamente conectadas, originalmente la arquitectura tiene 3 capas completamente conectadas de 4096 neuronas cada una y una salida final de 1000. Se cambió la cantidad de neuronas a 512 para las 3 capas finales y la salida final se cambió a 20 para la capacitación de elementos de ECUSTD y a 101 para la capacitación de elementos de Food 101.

Para GoogLeNet y MobileNet, se cambió la cantidad de neuronas en la capa de salida final para que coincida con la cantidad de categorías en ambas bases de datos.

Las 3 arquitecturas fueron entrenadas en ambos conjuntos de datos, para el conjunto de datos ECUSTD se usaron dos técnicas para el entrenamiento: **transferencia de aprendizaje** y **aprendizaje completo**.

La transferencia de aprendizaje se utiliza cuando en las capas convolucionales de una arquitectura ya se tienen buenos extractores de información, pero se desea enfocar el resto de la red en una tarea en específico. Este tipo de entrenamiento permite obtener buenos resultados en tareas específicas sin tener que obtener una muestra tan grande de ejemplos para reentrenar la parte no convolucional de la red. En cambio en el entrenamiento de aprendizaje completo se tiene que entrenar la red completa, por lo que, sino se tiene una muestra bastante grande de ejemplos los resultados no serán prometedores. Además, en el aprendizaje completo tarda varios órdenes de magnitud más en tiempo, para ser entrenado, dependiendo de la arquitectura ya que

las convoluciones son operaciones costosas.

Dado que el conjunto de datos de Food 101 es considerablemente más grande, solo se probó con transferencia de aprendizaje (ya que el aprendizaje completo tardaría mucho tiempo en entrenar). La técnica de transferencia de aprendizaje aplicada en este trabajo fue congelar las capas convolucionales de la red con los pesos entrenados con el conjunto de datos de ImageNet y solo entrenar las capas completamente conectadas durante 100 épocas. La técnica de entrenamiento completo entrena todas las capas, incluidas las capas de convolución, con el conjunto de datos seleccionado durante 100 épocas.

La función de pérdida utilizada en todos los experimentos fue la función de pérdida de entropía cruzada, el optimizador Adam Step se utilizó en la mayoría de los experimentos, excepto en uno en el que se utilizó el descenso de gradiente estocástico. El método de validación utilizado fue una validación cruzada de 5 pliegues.

Los experimentos se realizaron utilizando la biblioteca Pytorch de Python y se ejecutaron en el servidor colaborativo de Google y el servidor BUAP. [?]

Los cuadros 5 y 6 se muestran los resultados de los diferentes experimentos realizados:

Cuadro 5: Experimentos - ECUSTD (Plataforma Colab)

Modelo	Trans. Aprend.	Épocas	Opt.	Val.	Prec.	Tiempo
GoogleNet	V	100	Adam	5 folds	99.56	1 h
GoogleNet	V	100	SGD	5 folds	99.43	45m
GoogleNet	F	100	Adam	5 folds	99.6	1h 42m
VGG16	V	100	Adam	5 folds	10.4	1h 15m
VGG16	F	100	Adam	5 folds	5.01	2h
MobileNet	V	100	Adam	5 folds	99.76	40m
MobileNet	F	100	Adam	5 folds	99.63	1h 30m

Cuadro 6: Experimentos - Food 101 (Plataforma Servidor BUAP)

Modelo	Trans. Aprend.	Épocas	Opt.	Val.	Prec.	Tiempo
MobileNet	V	100	Adam	5 folds	63.08	12 h
VGG16	V	100	Adam	5 folds	18.14	24 h
GoogLeNet	V	100	Adam	5 folds	66.3	14 h

De los resultados de los cuadros 5 y 6 se puede observar que para el conjunto de datos ECUSTD se puede obtener una precisión muy buena con el aprendizaje por transferencia en dos de las tres arquitecturas con tiempos de entrenamiento relativamente bajos. VGG16 tuvo la precisión más baja tanto con aprendizaje de transferencia como sin él, es importante señalar que VGG16 es el modelo con más parámetros para optimizar y es por eso que los tiempos de entrenamiento son los más grandes en general. MobileNet y GoogLeNet no mejoraron mucho la precisión al realizar un entrenamiento completo sin transferencia de aprendizaje.

Para el conjunto de datos Food 101, la mayor precisión alcanzada se obtuvo con GoogLeNet, el segundo y muy cercano fue MobileNet. VGG16, al igual que con el otro conjunto de datos, no logró realizar una clasificación decente.

Con los experimentos anteriores se pudo llegar a la conclusión que la clasificación de alimentos mediante el aprendizaje por transferencia y las conocidas arquitecturas CNN se puede lograr satisfactoriamente en los casos en que las clases de alimentos son simples y están aisladas en una imagen, por ejemplo, imágenes de frutas. Por otro lado, cuando se tienen en cuenta platos complejos el reconocimiento cae drásticamente, sin embargo, un 60 % de precisión para platos complejos sin ningún tipo de segmentación previa o detección de objetos refleja un resultado positivo, porque significa que con una segmentación y detección adecuada, la precisión puede mejorar mucho.

5.2. Detección de platillos en una escena utilizando Redes Convolucionales Profundas.

El primer paso del pipeline general del proceso es precisamente determinar en que parte de la escena (fotografía) se encuentra el platillo con los alimentos. La tarea de encontrar donde se encuentra un objeto en una escena, ha sido resuelta en la literatura con Redes Neuronales Convolucionales Recurrentes y No Recurrentes. Ejemplos de la perspectiva recurrente son: **RCNN** (Recurrent Convolutional Neural Network), **Fast-RCNN** y **Faster-RCNN**, también se desarrolló el modelo **Mask R-CNN** que además de ubicar los objetos les asigna una máscara, es decir, en lugar de proponer un cuadro delimitador donde se encuentra el objeto, segmenta los píxeles del objeto, diferenciándolos de los demás que componen la escena.

Los modelos no basados en redes recurrentes son: **SSD (Single Shot Detectors)** y **YOLO(You only look once)**, estos modelos hacen uso de una única red neuronal convolucional para detectar objetos en imágenes. Para su funcionamiento, la red neuronal divide la imagen en regiones, prediciendo cuadros de identificación y probabilidades por cada región; las cajas son ponderadas a partir de las probabilidades predichas [104]. El algoritmo aprende representaciones generalizables de los objetos, permitiendo un bajo error de detección para entradas nuevas, diferentes al conjunto de datos de entrenamiento.

Las **Redes Residuales (ResNet)**, también se han utilizado ampliamente para la detección de objetos. Una red neuronal residual es una red neuronal artificial (ANN). Es una variante de una **HighwayNet** [105]; la primera red neuronal de retroalimentación muy profunda que funciona con cientos de capas, mucho más profundas que las redes neuronales anteriores. Las conexiones de salto o los accesos directos se utilizan para saltar sobre algunas capas (Las HighwayNets también puede aprender los pesos de salto por sí mismos a través de una matriz de peso adicional para sus puertas) [106]. Los modelos típicos de ResNet se implementan con saltos de doble o triple capa que contienen no linealidades (ReLU) y normalización por lotes en el medio. Los modelos con varios saltos paralelos se denominan **DenseNets** [107]. En el contexto de las redes neuronales residuales, una red no residual puede describirse como una red simple.

5.2.1. Propuesta de diseño - Detección de platillo.

Existen muchas arquitecturas diferentes para resolver el problema de detección de objetos, decidir que arquitectura utilizar para desarrollar la solución de encontrar un platillo en una escena es una labor de investigación en sí misma. Desarrollar una arquitectura propia desde cero, en el contexto anterior sería equivalente a tratar de reinventar la rueda. Es por eso que en este trabajo se eligió trabajar con una arquitectura ya existente; en realidad, probar varias y más aún, utilizar transferencia de aprendizaje en cada una de ellas para acelerar el entrenamiento y poder comparar los resultados de precisión.

Como el lenguaje de programación elegido es Python con la biblioteca TensorFlow (debido a que Pytorch es menos flexible en ciertos aspectos).

Los modelos de arquitecturas a los que se acotó para probar fueron los siguientes:

Cuadro 7: Arquitecturas elegidas disponibles en TensorFlow preentrenadas con el conjunto de datos COCO (Common Objects in Context).

Modelo	Velocidad (ms)	COCO maP	Output
CenterNet HG 512x512	70	41.9	Boxes/Keypoints
SSD MobileNet 320x320	22	22.2	Boxes
SSD MobileNet 640x640	39	28.2	Boxes
Faster R-CNN 640x640	53	29.3	Boxes
ResNet50 640x640	46	34.3	Boxes

Fuente: Elaboración propia con información del TensorFlow Zoo [108]

Como se puede observar la red CenterNet tiene la mayor precisión (mean average precision) en el dataset COCO, sin embargo, es la red más lenta en realizar detecciones con 70 milisegundos por detección. Las redes SSD parecen tener una menor precisión pero significativamente mayor velocidad. La red Faster-RCNN y la red ResNet tienen velocidades promedio y precisiones promedio. Existen muchas arquitecturas más en la biblioteca de TensorFlow, sin embargo, se decidió probar sólo con las que aparecen en el cuadro 7. La arquitectura YOLO se descartó porque tiene un desempeño similar a la arquitectura SSD y no se encuentra disponible en TensorFlow.

Para aplicar la transferencia de aprendizaje en las arquitecturas anteriores fue necesario obtener una muestra de fotografías que contengan un platillo con alimentos en la escena y etiquetar dichas fotografías, creando el cuadro delimitador y un archivo aparte (para cada fotografía) con las coordenadas de los vértices del cuadro delimitador y las dimensiones de la imagen (se desarrolló un script de python [109]).

Se recopiló una base de 1084 fotografías (tomadas por el autor) con sus respectivas etiquetas, la cual se ha denominado **DishLoc** (Todos los derechos reservados).

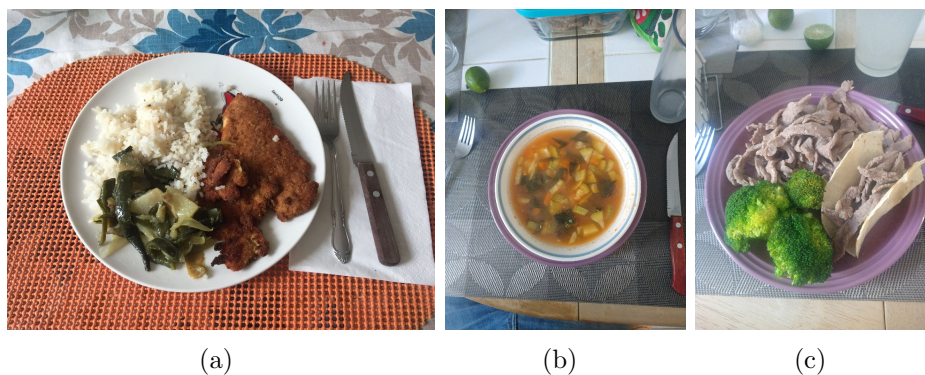


Figura 22: Ejemplos de la base DishLoc.

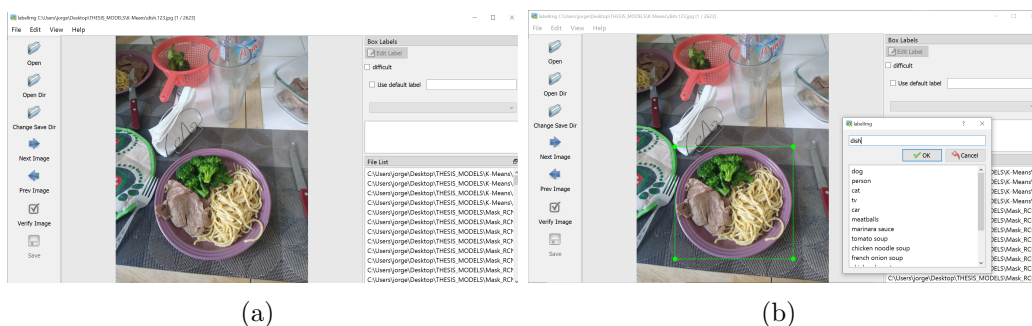


Figura 23: Etiquetador de imágenes desarrollado por Nicholas Renotte [110]
a) Antes de etiquetar b) Después de etiquetar.


```

<?xml version="1.0"?>
- <annotation>
  <folder>K-Means</folder>
  <filename>dish.123.jpg</filename>
  <path>C:\Users\jorge\Desktop\THESIS_MODELS\K-Means\dish.123.jpg</path>
  - <source>
    <database>Unknown</database>
  </source>
  - <size>
    <width>600</width>
    <height>800</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  - <object>
    <name>dish</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    - <bndbox>
      <xmin>119</xmin>
      <ymin>350</ymin>
      <xmax>488</xmax>
      <ymax>700</ymax>
    </bndbox>
  </object>
</annotation>

```

Figura 24: Ejemplo de archivo XML con la información de la etiqueta.

5.2.2. Resultados - Detección de platillos.

Para el entrenamiento se desarrolló un script en Python [111], donde se dividen las imágenes con sus respectivas etiquetas en dos conjuntos; el conjunto de entrenamiento y el conjunto de prueba ó evaluación. El conjunto de entrenamiento consta de una muestra aleatoria del 80 % de las imágenes, esta muestra es la que se utiliza para entrenar los modelos. El entrenamiento utilizado es transferencia de aprendizaje, es decir, las capas convolucionales de las arquitecturas se quedan con los pesos pre-entrenados con la base de datos COCO, mientras que las capas no convolucionales son las capas que son entrenadas con la base de datos DishLoc.

El conjunto de prueba que constituye el 20 % restante se utiliza para medir la precisión de los modelos. La medida de precisión usualmente utilizada en los modelos de detección de objetos es el **Mean Average Precision (mAP)**, es decir, los casos verdaderos positivos sobre los verdaderos positivos más los falsos positivos, para de un límite de la **intersección sobre la unión (IoU)** de los cuadros delimitadores reales contra los predichos. Para este trabajo se propuso un límite de IoU de 0.75.

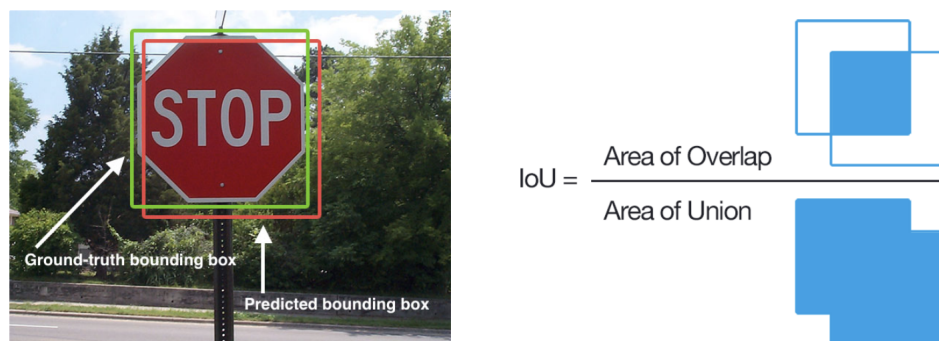


Figura 25: Intersección sobre Unión. [112]

If IoU threshold = 0.5

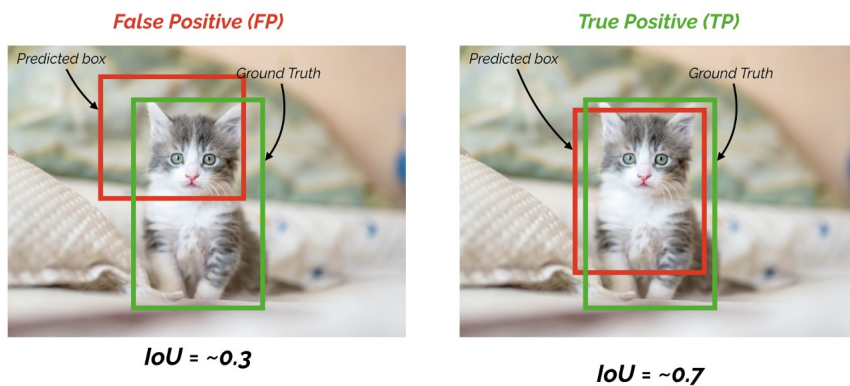


Figura 26: Ejemplo de Falsos Positivos y Verdaderos Positivos para un límite de IoU de 0.5 [112]

Todos los entrenamientos en esta sección se realizaron en un CPU Intel(R) Core(TM) i7-8565 @ 1.8 GHz 1.99 GHz, 16.0GB RAM.

Los primeros entrenamientos mostrados a continuación se realizaron con un conjunto de entrenamiento de 492 imágenes y se evaluaron con 129 imágenes:

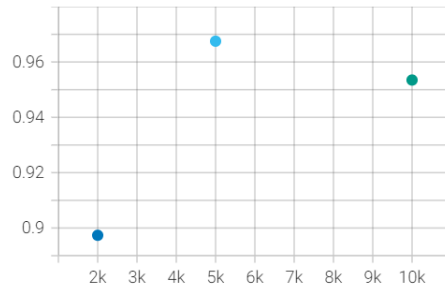
Cuadro 8: Resultados Entrenamiento Modelos de Detección.

Modelo	Entrenamiento	Épocas	mAP @ IoU = 0.75
SSD MobileNet 320x320	40m 37s	2000	89.7 %
SSD MobileNet 320x320	2h 17m	5000	96.8 %
SSD MobileNet 320x320	3h 1m	10000	95.3 %
SSD MobileNet 640x640	3h 8m	2000	91.8 %
SSD MobileNet 640x640	6h 18m	5000	92.5 %
SSD MobileNet 640x640	10h 50 m	10000	95.7 %
CenterNet HG 512x512	2d 7h 29m	2000	96.8 %
Faster R-CNN 640x640	9h 1m	2000	99.9 %
RestNet50 640x640	10h 2m	2000	90.3 %

Fuente: Elaboración propia [113]

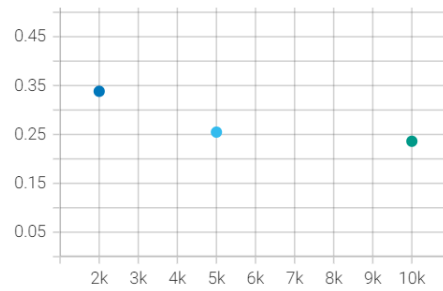
Los resultados obtenidos fueron sorprendentemente altos en precisión, siendo el más bajo 89.7 % para la red SSD con inputs de 320x320, sin embargo, ese resultado incrementó con un mayor número de épocas (cuando se incrementó a 5000), y luego se estanca cuando se incrementa a 10000 épocas esto se debe al fenómeno de **Sobreajuste**. El mismo comportamiento se puede observar para la red SSD 640x640. Para el resto de las arquitecturas, el tiempo de entrenamiento fue sustancialmente más alto debido a que el número de capas y neuronas es mucho mayor, es decir, el número de parámetros a optimizar es más grande. Estas arquitecturas alcanzaron niveles de precisión más alto con tan solo 2000 épocas de entrenamiento (Faster R-CNN obtuvo 99.9 %), se debe recordar también que las evaluaciones son más lentas en estas arquitecturas.

DetectionBoxes_Precision/mAP@.75IOU
tag: DetectionBoxes_Precision/mAP@.75IOU



(a)

Loss/total_loss
tag: Loss/total_loss



(b)

Figura 27: a) Precisión y b) Pérdida Total (conjunto de evaluación) del modelo SSD 320x320 con 2000, 4000 y 10000 épocas.

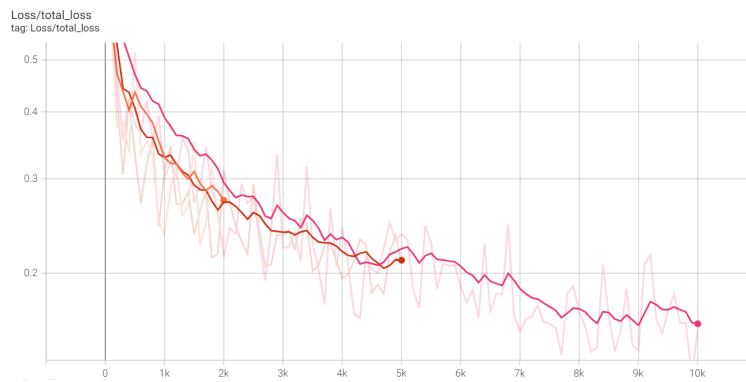


Figura 28: Pérdida Total (entrenamiento) para el modelo SSD320x320.

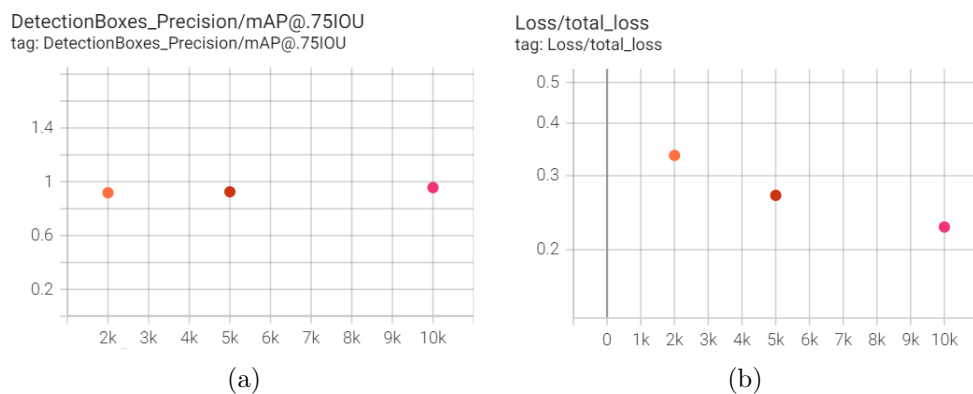


Figura 29: a) Precisión y b) Pérdida Total (conjunto de evaluación) del modelo SSD 640x640 con 2000, 4000 y 10000 épocas.

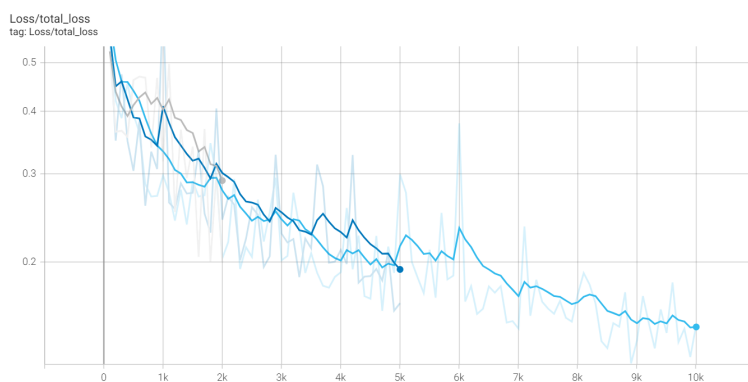


Figura 30: Pérdida Total (entrenamiento) para el modelo SSD640x640.

Se realizó otro lote de entrenamiento esta vez para tratar de refinar los resultados de la redes SSD, la muestra de entrenamiento se aumentó a 820 imágenes y se evaluó con 264 imágenes. Los resultados se muestran en el cuadro 9.

Cuadro 9: Resultados Entrenamiento Modelos de Detección.

Modelo	Entrenamiento	Épocas	mAP @ IoU = 0.75
SSD MobileNet 320x320	33m 26s	2000	91.8 %
SSD MobileNet 320x320	1h 25m	5000	91.2 %
SSD MobileNet 640x640	2h 20m	2000	88.9 %

Fuente: Elaboración propia [114]

La precisión se mejoró para el modelo SSD 320x320 convirtiéndolo en un modelo interesante por su velocidad de evaluación. El modelo SSD de entrada de 640x640 conservó una precisión similar a los primeros entrenamientos.

Se creó un script de Python [115] combinado con la herramienta de Front-End Mercury para poder realizar detecciones, es decir, se introduce una imagen, y se obtienen: una imagen con el cuadro delimitador donde se encuentra el platillo y una imagen recortada que contiene solo el interior del cuadro delimitador. Esta última imagen servirá como entrada para el siguiente paso que es la segmentación.

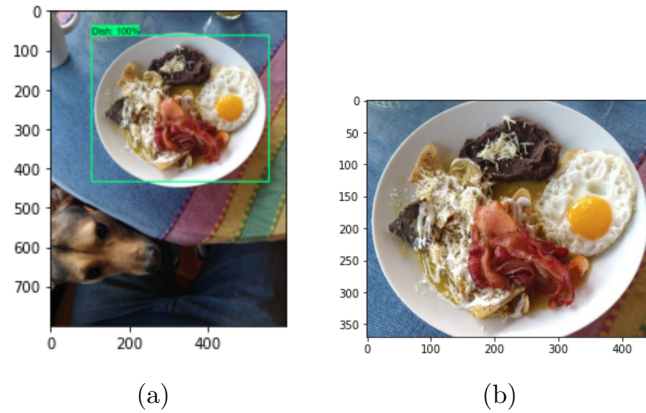


Figura 31: a) Detección de platillo usando red Faster RCNN entrenada y b) contenido del cuadro delimitador.

En conclusión los modelos obtenidos en esta tarea de detección de objetos funcionan satisfactoriamente en la mayoría de los casos y con un tiempo de evaluación adecuado, además, se pueden seguir refinando ampliando la base de datos de imágenes y de etiquetas para obtener modelos más robustos que produzcan cada vez una tarea de identificación de platillos más precisa.

5.3. Segmentación de alimentos en un platillo usando un clasificador de Bosque Aleatorio.

El siguiente paso en el proceso es la segmentación de los alimentos que se encuentran dentro del platillo detectado en la fotografía en el paso anterior. El tipo de segmentación necesaria es una segmentación semántica ya que es necesario dividir los alimentos que sean de distintas clases pero considerar cada clase como una sola instancia, así cada píxel de la imagen suma para el cálculo de área de la clase a la que pertenece sin importar donde se localiza en el platillo.

El método que se eligió para llevar a cabo la segmentación es un algoritmo de aprendizaje automático basado en un ensamble de árboles de decisión llamado **Bosque Aleatorio**.

No se eligió una técnica de aprendizaje profundo como la arquitectura Mask R-CNN porque se tendría la restricción de sólo hacer segmentaciones basadas en las clases con las que fue entrenada la red, en cambio con el algoritmo de Bosque Aleatorio se crea un clasificador entrenado sobre la marcha cada vez que se va a segmentar una nueva imagen, independientemente de los tipos de comida que aparezcan en la imagen. Otro ejemplo más drástico sería utilizar el algoritmo KNN (K-Nearest Neighbors), este algoritmo de aprendizaje automático no se basa en ningún entrenamiento sino que se basa en similitudes de datos observables y métricas de distancia sofisticadas para generar predicciones precisas [116], tanto el algoritmo de Bosque Aleatorio al vuelo y KNN se basan en el paradigma de **aprendizaje perezoso**, en donde la generalización de los datos de entrenamiento se retrasa hasta que se realiza una consulta al sistema, a diferencia del **aprendizaje entusiasta**, en el que el sistema intenta generalizar los datos de entrenamiento antes de recibir consultas [117] como es el caso con la red Mask R-CNN.

Se desea entrenar un clasificador ya que la segmentación no es más que asignar una clase, a cada píxel que conforman la imagen, para poder clasificar cada píxel se necesita extraer características de cada uno de ellos como su color y/o propiedades de cada uno de sus vecindarios(bordes, textura).

5.3.1. Extracción de características de imágenes.

En la visión por computadora y el procesamiento de imágenes, una característica es una información sobre el contenido de una imagen; típicamente sobre si una cierta región de la imagen tiene ciertas propiedades. Las características pueden ser estructuras específicas en la imagen, como puntos, bordes u objetos. Las características también pueden ser el resultado de una operación general de vecindad o detección de características aplicada a la imagen. Otros ejemplos de características están relacionados con el movimiento en secuencias de imágenes o con formas definidas en términos de curvas o límites entre diferentes regiones de imágenes. [118]

En términos más generales, una característica es cualquier información que sea relevante para resolver la tarea computacional relacionada con una determinada aplicación. Este es el mismo sentido que la característica en el aprendizaje automático y el reconocimiento de patrones en general, aunque el procesamiento de imágenes tiene una colección de características muy sofisticada. El concepto de característica es muy general y la elección de características en un sistema de visión por computadora en particular puede depender en gran medida del problema específico en cuestión [119].

5.3.2. Desenfoque Gaussiano.

En el procesamiento de imágenes, un desenfoque gaussiano (también conocido como suavizado gaussiano) es el resultado de desenfocar una imagen mediante una función gaussiana (llamada así por el matemático y científico Carl Friedrich Gauss).

Es un efecto ampliamente utilizado en el software de gráficos, normalmente para reducir el ruido de la imagen y reducir los detalles. El efecto visual de esta técnica de desenfoque es un desenfoque suave parecido al de ver la

imagen a través de una pantalla translúcida, claramente diferente del efecto bokeh producido por una lente desenfocada o la sombra de un objeto bajo iluminación habitual [120].

El suavizado gaussiano también se utiliza como una etapa de preprocesamiento en los algoritmos de visión por computadora para mejorar las estructuras de la imagen en diferentes escalas.



Figura 32: Diferencia entre un desenfoco gaussiano pequeño y grande [121]

Matemáticamente, aplicar un desenfoco gaussiano a una imagen es lo mismo que convolucionar la imagen con una función gaussiana. Esto también se conoce como transformada bidimensional de Weierstrass.

Dado que la transformada de Fourier de una gaussiana es otra gaussiana, aplicar un desenfoco gaussiano tiene el efecto de reducir los componentes de alta frecuencia de la imagen; un desenfoco gaussiano es, por lo tanto, un filtro de paso bajo [122] atenuando la señales de alta frecuencia.

5.3.3. Operador de Sobel.

El operador de Sobel, a veces llamado operador de Sobel-Feldman o filtro de Sobel, se usa en el procesamiento de imágenes y la visión por computadora, particularmente dentro de los algoritmos de detección de bordes donde crea una imagen que enfatiza los bordes. Lleva el nombre de Irwin Sobel y Gary Feldman, colegas del Laboratorio de Inteligencia Artificial de Stanford (SAIL). Sobel y Feldman presentaron la idea de un **Operador de gradiente de imagen isotrópico 3 x 3** en una charla en SAIL en 1968 [123]. Técnicamente, es un operador de diferenciación discreta que calcula una aproximación del gradiente de la función de intensidad de la imagen. En cada punto de la imagen, el resultado del operador Sobel-Feldman es el vector de gradiente correspondiente o la norma de este vector. El operador Sobel-Feldman se basa en convolucionar la imagen con un filtro pequeño, separable y de valor entero en las direcciones horizontal y vertical y, por lo tanto, es relativamente económico en términos de cálculos. Por otro lado, la aproximación de gradiente que produce es relativamente tosca, en particular para variaciones de alta frecuencia en la imagen.

El operador utiliza dos núcleos de 3x3 que se convolucionan con la imagen original para calcular aproximaciones de las derivadas: uno para cambios horizontales y otro para cambios verticales. Si se definen A como la imagen fuente, y G_x y G_y son dos imágenes que en cada punto contienen las aproximaciones de derivadas horizontal y vertical respectivamente, los cálculos son los siguientes [124]:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A$$

y

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

donde $*$ denota la operación de convolución de procesamiento de señal bidimensional.

El resultado del operador Sobel-Feldman es un mapa bidimensional del gradiente en cada punto. Se puede procesar y ver como si fuera una imagen en sí misma, con las áreas de alto gradiente (los bordes probables) visibles como líneas blancas. Las siguientes imágenes ilustran esto, mostrando el cálculo del operador Sobel-Feldman en una imagen simple.

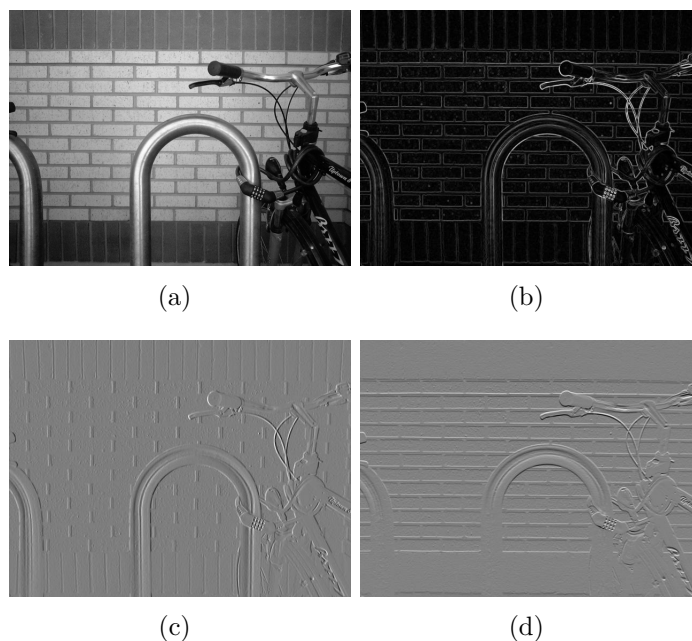


Figura 33: a) Fotografía original y operador de Sobel en: (b) Magnitud de gradiente denotado I_{xy} (c) Magnitud de gradiente x denotado I_x , (d) Magnitud de gradiente y denotado I_y . [125] [126] [127] [128]

5.3.4. Matriz Hessiana.

En matemáticas, la matriz hessiana ó simplemente hessiana es una matriz cuadrada de derivadas parciales de segundo orden de una función con valores escalares, o campo escalar [129]. Describe la curvatura local de una función de muchas variables. La matriz hessiana fue desarrollada en el siglo XIX por el matemático alemán Ludwig Otto Hesse y más tarde recibió su nombre. Hesse usó originalmente el término **determinantes funcionales**.

Si se supone que $f : \mathbb{R}^n \rightarrow \mathbb{R}$ es una función que toma como entrada un vector $x \in \mathbb{R}^n$ y genera un escalar $f(x) \in \mathbb{R}$. Si todas las segundas derivadas parciales de f existen entonces la matriz hessiana H de f es una matriz cuadrada de $n \times n$ usualmente definida como:

$$(\text{Hess } f)_{ij} \equiv \frac{\partial^2 f}{\partial x_i \partial x_j}$$

La operación Hessiana para una imagen I en 2D esta definida como :

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \times I = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

donde I_{xx} es la aplicación del filtro de sobel G_x dos veces en la imagen, así mismo I_{yy} es la aplicación del filtro de sobel G_y , I_{xy} denota aplicar G_x y G_y consecutivamente o viceversa.

La operación Hessiana en una imagen 2D se puede escribir en términos de los eigenvalores λ_i de la matriz resultante:

$$\begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

De estos eigenvalores se pueden obtener características de la imagen, dependiendo del problema que se está tratando. Por ejemplo, en medicina, los eigenvalores aportan información sobre la estructura de los vasos sanguíneos en fotografías médicas. [130]

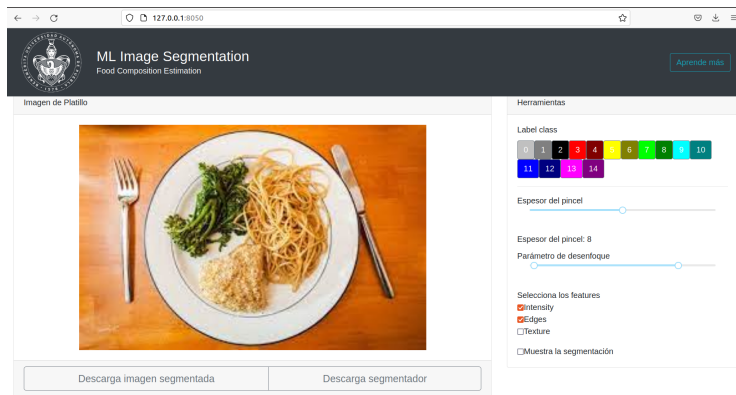
5.3.5. Resultados - Segmentación con clasificador de Bosque Aleatorio.

Se creó una aplicación en Python en combinación con la biblioteca Dash para darle forma al Front-End [131]. En dicha aplicación se calcula una segmentación basada en píxeles utilizando características locales basadas en la intensidad, los bordes y las texturas locales a diferentes escalas. Se utiliza una máscara proporcionada por el usuario para identificar diferentes regiones. Los píxeles de la máscara se utilizan para entrenar un clasificador de bosque aleatorio [132] de scikit-learn. A continuación, los píxeles sin etiquetar se etiquetan a partir de la predicción del clasificador.

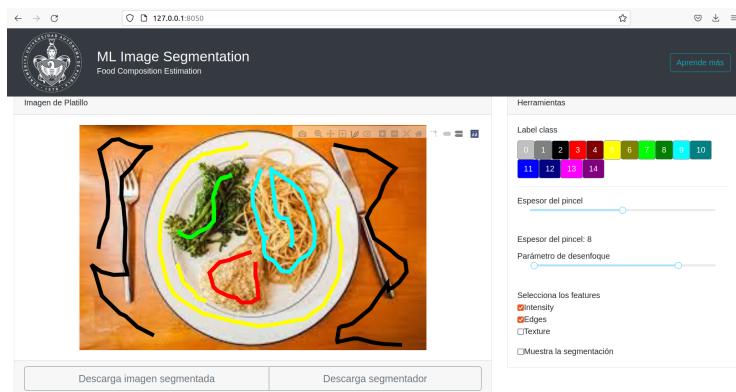
Todas las características extraídas en este modelo están sujetas a los dos parámetros por default $\sigma_{min} = 1$ y $\sigma_{max} = 16$. Las características de intensidad local, por ejemplo, son el promedio de intensidad en vecindarios que van de 1 píxel hasta 16 píxeles, con incrementos de 2^{n-1} con $n = 1, 2, 3, 4, 5$ en los tres canales RGB de la imagen. Las características de bordes se calculan con el promedio de la intensidad de los gradientes en los distintos vecindarios antes mencionados, es decir, se aplican los operadores de Sobel después de un desenfoque gaussiano para los diferentes valores de sigma. Y para las características de textura se toman los eigenvalores de la matriz Hessiana después de aplicar un desenfoque gaussiano para los distintos valores de sigma [133]. La extracción de características de alguna categoría es desactivable en cualquier momento en la aplicación de segmentación.

El número de árboles de decisión que componen al bosque aleatorio se fija en 50, con una profundidad máxima de 10 niveles, se utilizó el **índice de gini** para medir la calidad de la divisiones binarias al construir los árboles de decisión. [134]

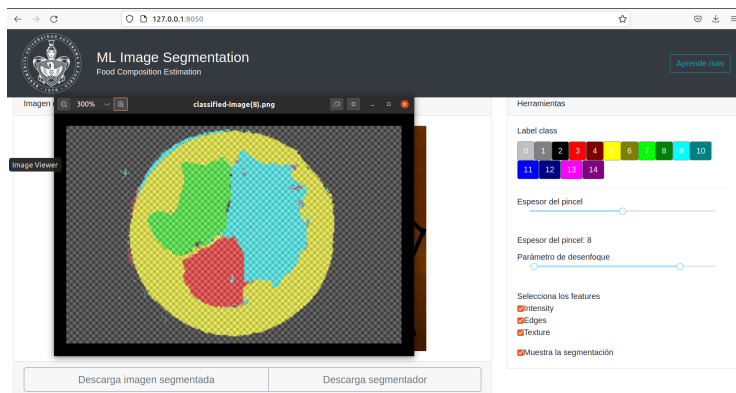
La figura 34 muestra el proceso y resultado de la segmentación de una imagen de un platillo, que debe ser la salida del paso de detección de objetos. La máscara resultante de la segmentación será el punto de partida del siguiente paso para el cálculo de áreas, clasificación y finalmente cálculo de calorías.



(a)



(b)



(c)

Figura 34: a) Foto original de un platillo, b) Píxeles segmentados por usuario para entrenar el algoritmo c) Máscara de imagen segmentada con el algoritmo de Bosque Aleatorio.

5.4. Estimación calórica basada en área de imagen segmentada.

El último paso del proceso, es la estimación calórica; basada en las máscaras de segmentos obtenidas en el paso anterior de segmentación. Para la realización de este último paso, se creó un script de Python que se ejecuta en un Jupyter Notebook [135], el script utiliza como entradas para procesar la máscara de segmentos y el platillo detectado en los pasos anteriores.

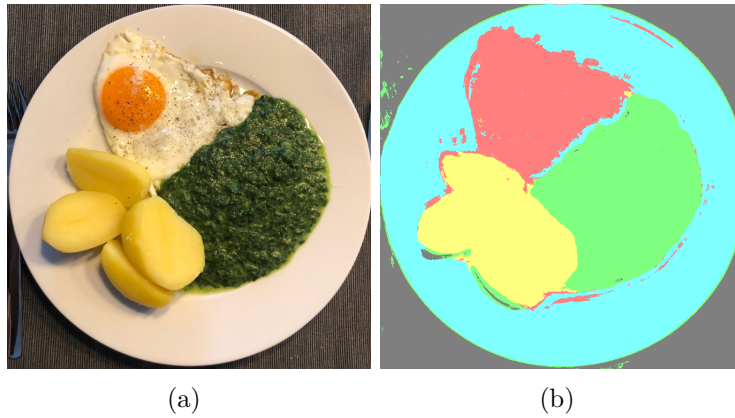
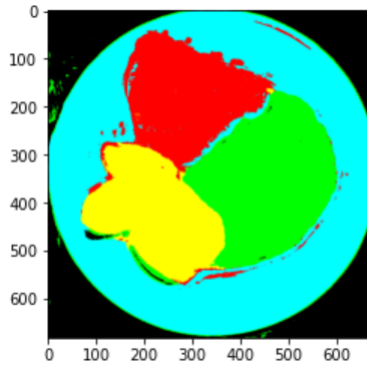


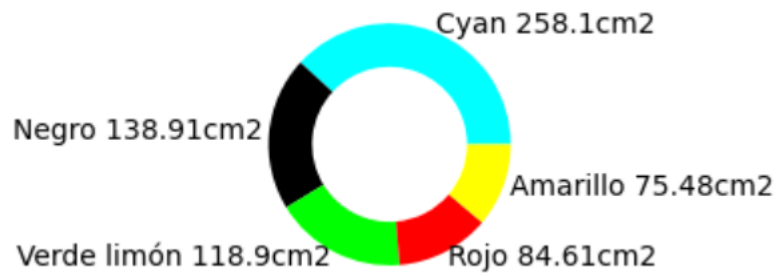
Figura 35: Entradas para el paso final de estimación de calorías : a) Platillo detectado de una escena completa, b) Máscara de segmentos obtenida en el paso de segmentación.

La primera tarea para realizar la estimación calórica es obtener el área que representa cada color en la máscara de segmentos. Para poder calcular las áreas necesarias se necesita alguna medida de referencia, ya que se desconoce la distancia a la que se encuentra el platillo del dispositivo que captura la fotografía. La referencia más fácil y práctica de obtener es el diámetro del plato.

En el ejemplo de la figura 35 el diámetro del plato es de 26cm (un plato estándar tiene entre 25 y 28 cm), lo cual quiere decir que la imagen, que resulta ser un cuadrado tiene de área $(26cm)^2 = 676cm^2$ y cuenta con 464,440 píxeles, por lo tanto cada píxel tiene un área de $\frac{676cm^2}{464,440} \approx 0,0014555cm^2$. Ya que se conoce el área unitaria por píxel se puede calcular el área por color.



(a)



(b)

Figura 36: (b) áreas obtenidas a partir de las clases de la máscara segmentada (a).

Una vez obtenidas las áreas por clases/colores, falta asociar cada color/clase al alimento que representa en la realidad. Para crear la asociación se utilizan inputs del usuario final, programaticamente el script de Python utiliza un ciclo que recorre cada clase y pregunta al usuario de que alimento se trata; el usuario escribe en un input box la clase del alimento, la figura 37 muestra la progresión del proceso.


```

In [123]: list_food = []
          for i in list_color:
            list_food.append(input("Que alimento representa el color " + i + ":"))

Que alimento representa el color Cyan: plato

[*]: In [123]: list_food = []
          for i in list_color:
            list_food.append(input("Que alimento representa el color " + i + ":"))

Que alimento representa el color Cyan:plato
Que alimento representa el color Negro:fondo
Que alimento representa el color Verde limón: espinaca

In [123]: list_food = []
          for i in list_color:
            list_food.append(input("Que alimento representa el color " + i + ":"))

Que alimento representa el color Cyan:plato
Que alimento representa el color Negro:fondo
Que alimento representa el color Verde limón:espinaca
Que alimento representa el color Rojo:huevo
Que alimento representa el color Amarillo:papa

```

Figura 37: Progresión de la captura del nombre de las clases por el usuario, es necesario capturar las clases plato y fondo, las cuales en la suma final de calorías aportan 0 calorías.

Hasta este momento ya se cuenta cuanto mide cada área segmentada en centímetros cuadrados y a que clase de alimento pertenece:

Cuadro 10: Segementos, áreas y clases.

Color	Área (cm^2)	Clase
Cyan	258.1	plato
Amarillo	75.48	papa
Rojo	84.61	huevo
Negro	138.91	fondo
Verde Limón	118.9	espinaca

Para poder estimar la cantidad de calorías en cada área de cada clase de alimento naturalmente se necesita el dato **calorías por centímetro cuadrado**, si se tuviera el dato entonces el cálculo de calorías sería bastante directo:

$$C_i = T_i \times A_i$$

donde C_i es el total de calorías en el segmento de clase i , T_i son las calorías por centímetro cuadrado de la clase i y A_i representa el área de la clase i en la imagen.

El dato de calorías por centímetro cuadrado no se encuentra disponible directamente en la web, sin embargo, los datos que si se encuentran ampliamente en la red son las calorías por gramaje (usualmente por 100g) de las distintas clases de alimentos.

Una búsqueda sencilla de “huevos calorías” produce el resultado: **155 calorías por 100 gramos**. Estos resultados que arroja google de forma directa provienen del departamento de agricultura de Estados Unidos (USDA)

espinacas calorías

× | 🔊 📷 🔍

🔍 Todo 🖼️ Imágenes 📺 Vídeos 📰 Noticias 🛒 Shopping ⋮ Más 🛠️ Herramientas

Cerca de 284,000 resultados (0.49 segundos)

Espinaca / Cantidad de calorías

23 calorías

Tipo Cantidad

Espinacas 100 gramos

Las fuentes incluyen: USDA Comentarios

Figura 38: Búsqueda de Espinaca calorías en Google.

La búsqueda de distintas clases de alimentos y sus calorías por gramaje constituyen la base de conocimiento que se agrupan en una base de datos, en el cuadro 11 se muestra un ejemplo. El mismo ejercicio se puede realizar para macronutrientes, vitaminas y minerales.

Cuadro 11: Ejemplo Base de conocimiento - Calorías por clase.

Clase	Calorías por 100g
Huevo	155
Frijol	91
Espinaca	23
Brocoli	34
Carne de res	250
Pollo	239
Arroz	130
Pasta	131
Manzana	52
Mango	60
...	...

Ahora que se cuenta con la información de calorías por peso surge la incógnita ¿Cómo se puede traducir a calorías por área?. Para traducir las calorías por peso a calorías por área se requiere el siguiente proceso, situar un alimento en un plato en el cual se conozca su diámetro, pesar el plato(vacío) y el plato con el alimento (para obtener el peso del alimento aislado), después se toma una fotografía del plato con el alimento (desde una perspectiva superior), dicha fotografía se pasa por el proceso de detección de objetos y segmentación y se obtiene el área del alimento.

Contando con los datos de área y peso reales se puede realizar el cálculo:

$$\frac{W_i}{A_i} = S_i$$

Donde W_i representa el peso, A_i el área y S_i los gramos por centímetro cuadrado.

Para refinar el resultado, el proceso descrito en el párrafo anterior puede realizarse repetidas veces, con los alimentos en disitintas posiciones y sacar un promedio de gramos por centímetro cuadrado por clase de alimento.

Una vez obtenido los valores S_i , se puede calcular fácilmente las calorías por clases y las calorías totales en la escena.

Cuadro 12: Ejemplo de Cálculo de calorías para el platillo de la figura 36

Clase	Calorías por 100g	Área	S_i	Calorías
Huevo	155	84.60	1.7	$(84,60 \cdot 1,7) \cdot \left(\frac{155}{100}\right) = 222,92$
Espinaca	23	118.89	0.5	$(118,89 \cdot 0,5) \cdot \left(\frac{23}{100}\right) = 13,67$
Papa	86	75.48	2.3	$(75,48 \cdot 2,3) \cdot \left(\frac{86}{100}\right) = 149,29$
Plato	0	258.10	NA	0
Fondo	0	138.91	NA	0

El cuadro 12 muestra los cálculos para la estimación calórica del platillo. La clase plato y fondo por default tienen valor 0 en calorías y aunque se calcula el área que representan en la escena no se contabiliza ninguna caloría. Las calorías totales serán la suma de la columna calorías en este caso **Calorías totales** = $222.92 + 13.67 + 149.29 = \mathbf{385.88}$

Con este resultado se ha llegado a la conclusión que se deseaba: una estimación calórica de un platillo a partir del tratamiento de su imagen digital.

La estimación no es perfecta pero es un punto de partida; el cual depende mucho de la obtención de la base de conocimiento y de los valores de S_i (cuya exactitud puede variar dependiendo de las condiciones), de la exactitud de los modelos de aprendizaje profundo y automático y de los inputs humanos. Sin embargo, el diseño de todo el proceso se pensó justamente valorando la mejora continua a la cual pueden someterse cada una de sus etapas, y que su mejora dependa en el mejoramiento del entrenamiento ó la obtención de mejor conocimiento y no tanto de las arquitecturas de los modelos o las clases en los entrenamientos.

6. Conclusiones.

En conclusión en este trabajo se produjeron los siguientes resultados:

- Se creó una base de datos de 1084 imágenes de alimentos (Dishloc) etiquetada para entrenamiento de modelos de detección de objetos,
- Se produjeron 13 modelos entrenados con transferencia de aprendizaje para detectar los platillos en la imagen, crear un cuadro delimitador y recortar los contenidos dentro del cuadro delimitador.
- Se produjo un modelo de Bosque Aleatorio para segmentar los alimentos de las imágenes así como el plato que los contiene y el fondo.
- Se produjo un módulo que calcula las áreas de los segmentos del paso anterior y en base a estas áreas y una base de conocimiento en una base de datos calcula la estimación calórica.

Quedó fuera del alcance de este trabajo crear una implementación que una todos los procedimientos anteriores en una sola ejecución. La implementación puede ser móvil, es decir, desarrollada en software como android o iOS pero también se puede hacer una implementación para PC. Todos los módulos aceptan imágenes de cualquier resolución así que no importa de donde provenga la imagen digital.

6.1. Investigación futura.

Este trabajo abre las puertas para que la investigación en esta línea se profundice. Algunos temas que se pueden abordar son la ampliación de la base de conocimiento para incluir comidas de muchos países además incluir también macronutrientes, vitaminas y minerales, para los cuales el proceso de medición sería casi idéntico a las calorías. También se puede investigar como obtener valores S_i más exactos para mejorar los cálculos. Los modelos de detección de platillos y segmentación también son mejorables, ampliando el conjunto de entrenamiento e investigando mejores características para la segmentación. En conclusión, la investigación queda abierta.

Referencias

- [1] Secretaría de Gobernación (2018). Sobre peso y obesidad factores para desarrollar diabetes. Tomado el 17 de Septiembre de 2022 en: <https://www.gob.mx/salud/es/articulos/sobrepeso-y-obesidad-factores-de-riesgos-para-desarrollar-diabetes>
- [2] Gurriá, José Ángel (2021). The Heavy Burden of Obesity: The Economics of Prevention. Tomado el 17 de Septiembre de 2022 de: <https://www.oecd.org/about/secretary-general/heavy-burden-of-obesity-mexico-january-2020.htm>.
- [3] Szeliski, Richard . Computer Vision: Algorithms and Applications (2011). Nueva York. pp.3
- [4] Ballard, Dana H; Brown, Christopher M. (1982). Computer Vision. Prentice Hall.
- [5] Huang, T, Vandoni, Carlo, E. (1996) Computer Vision : Evolution And Promise. 19th CERN School of Computing. Geneva: CERN. pp. 21-25.
- [6] Klette, Reinhard. (2014). Concise Computer Vision. Springer
- [7] Shapiro, Linda G; Stockman, George C. (2001). Computer Vision. Prentice Hall.
- [8] Morris, Tim (2004). Computer Vision and Image Processing. Palgrave Macmillan.
- [9] Jähne Bernd ; Haußecker Horst (2000). Computer Vision and Applications, A Guide for Students and Practitioners. Academic Press.
- [10] Johnson, Justin; Karpathy, Andrej; Fei-Fei Li. DenseCap: Fully Convolutional Localization Networks for Dense Captioning. 2016. CVPR. Tomado el 17 de Septiembre de 2022 de: <https://github.com/jcjohnson/densecap>
- [11] What is image classification. Image Classification. (2022). En huggingface.co, <https://huggingface.co/tasks/image-classification>.
- [12] Tomado el 17 de Septiembre de: <https://huggingface.co/tasks/image-classification>

- [13] Dasiopoulou, Stamatia, et al. Knowledge-assisted semantic video object detection. *IEEE Transactions on Circuits and Systems for Video Technology* 15.10 (2005): 1210-1224
- [14] Wikimania2009, Beatrice Murch (2009). Jimmy answering questions. Tomado el 21 de Septiembre de 2022 de: https://commons.wikimedia.org/wiki/File:Jimmy_answering_questions.jpg
- [15] Vincent Desjardins (2010). Paris voie Georges Pompidou sans voitures. Tomado el 21 de Septiembre de 2022 de: https://en.wikipedia.org/wiki/File:D%C3%A9tection_de_personne_-_exemple_1.jpg
- [16] Ling Guan; Yifeng He; Sun-Yuan Kung (2012). *Multimedia Image and Video Processing*. CRC Press. pp. 331.
- [17] Alsanabani, Ala; Ahmed, Mohammed; AL Smadi, Ahmad (2020). Vehicle Counting Using Detecting-Tracking Combinations: A Comparative Analysis. 2020 the 4th International Conference on Video and Image Processing. pp. 48-54.
- [18] Wu, Jianxin, et al. (2007). A scalable approach to activity recognition based on object use. 2007 IEEE 11th international conference on computer vision. IEEE, 2007.
- [19] MTheiler (2019). Schreibtisch-mit-Objekten. Tomado el 21 de Septiembre de 2022 de: https://en.wikipedia.org/wiki/Object_detection#/media/File:Detected-with-YOLO--Schreibtisch-mit-Objekten.jpg
- [20] Comunidad de Software Libre Hackem (2020). HackemCON Series Sample of Simón Bolívar Avenue. Tomado el 21 de Septiembre de 2022 de: https://en.wikipedia.org/wiki/Object_detection#/media/File:Computer_vision_sample_in_Sim%C3%B3n_Bolívar_Avenue,_Quito.jpg
- [21] Viola, Paul; Jones, Michael J. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision*. 57 (2). pp. 137-154.
- [22] Lowe, David G. (1999). Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision*. Vol. 2. pp. 1150-1157.

- [23] Dalal, Navneet (2005). Histograms of oriented gradients for human detection. Computer Vision and Pattern Recognition.
- [24] Ross, Girshick (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE. pp. 580-587.
- [25] Girschick, Ross (2015). Fast R-CNN. Proceedings of the IEEE International Conference on Computer Vision. pp. 1440-1448.
- [26] Shaoqing, Ren (2015). Faster R-CNN. Advances in Neural Information Processing Systems.
- [27] Pang, Jiangmiao; Chen, Kai; Shi, Jianping; Feng, Huajun; Ouyang, Wanli; Lin, Dahua (2019). Libra R-CNN: Towards Balanced Learning for Object Detection.
- [28] Liu, Wei (2016). SSD: Single shot multibox detector. Computer Vision - ECCV 2016. European Conference on Computer Vision. Lecture Notes in Computer Science. Vol. 9905. pp. 21-37.
- [29] Redmon, Joseph (2016). You only look once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.
- [30] Redmon, Joseph (2017). YOLO9000: better, faster, stronger. Tomado el 24 de Septiembre de 2022 de: <https://arxiv.org/abs/1612.08242>
- [31] Redmon, Joseph (2018). Yolov3: An incremental improvement. Tomado el 24 de Septiembre de 2022 de: <https://arxiv.org/abs/1804.02767>
- [32] Wang, Chien-Yao (2021). Scaled-YOLOv4: Scaling Cross Stage Partial Network. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Tomado el 24 de Septiembre de 2022 de: <https://arxiv.org/abs/2011.08036>
- [33] Lin, Tsung-Yi (2020). Focal Loss for Dense Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 42 (2): pp. 318-327.

- [34] Zhu, Xizhou (2018). Deformable ConvNets v2: More Deformable, Better Results. Tomado el 24 de Septiembre de 2022 de: <https://arxiv.org/abs/1811.11168>
- [35] Dai, Jifeng (2017). Deformable Convolutional Networks. Tomado el 24 de Septiembre de 2022 de: <https://arxiv.org/abs/1703.06211>
- [36] Shapiro, Linda G; Stockman George C. (2001): Computer Vision, pp 279-325, Prentice-Hall.
- [37] Barghout, Lauren, and Lawrence W. Lee. Perceptual information processing system. Paravue Inc. Aplicación de patente 10/618,543, archivada el 11 de Julio de 2003.
- [38] Nielsen, Frank; Nock, Richard (2003). On region merging: The statistical soundness of fast sorting, with applications. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. Vol. 2. IEEE. pp.19-26.
- [39] Shapiro, Linda G; Stockman, George C. (2001). Computer Vision. pp 279-325, New Jersey, Prentice-Hall.
- [40] Zachow, Stefan; Zilske, Michael; Hans-Christian Hege. 3D reconstruction of individual anatomy from medical image data: Segmentation and geometry processing. (2007).
- [41] Extraída de WikiCommons el 2 de Octubre de 2022 de: https://upload.wikimedia.org/wikipedia/commons/d/d4/Image_segmentation.png
- [42] Guo, Dazhou; Pei, Yanting; Zheng, Kang; Yu, Hongkai; Lu, Yuhang; Wang, Song (2020). Degraded Image Semantic Segmentation With Dense-Gram Networks. IEEE Transactions on Image Processing. 29: 782-795.
- [43] Yi, Jingru; Wu, Pengxiang; Jiang, Menglin; Huang, Qiaoying; Hoeppner, Daniel J.; Metaxas, Dimitris N. (2019). Attentive neural cell instance segmentation. Medical Image Analysis. 55: 228-240.
- [44] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, Piotr Dollár (2018). Panoptic Segmentation. Tomado el 2 de Octubre de: <https://arxiv.org/abs/1801.00868>

- [45] Extraída de Research Gate el 2 de Octubre de 2022 de: <https://www.researchgate.net/profile/Vinorth-Varatharasan/publication/339328277/figure/fig1/AS:86455488820429101583137356688/Semantic-segmentation-left-and-Instance-segmentation-right-8.ppm>
- [46] Extraída el 2 de Octubre de 2022 de: https://mlv4xkdrf2yq.i.optimole.com/cb:011_~a759/w:1024/h:576/q:mauto/https://datagen.tech/app/uploads/2022/05/image2-1.png
- [47] Mitchell, Tom (1997). Machine Learning. New York: McGraw Hill.
- [48] Koza, John R.; Bennett, Forrest H.; Andre, David; Keane, Martin A. (1996). Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming. Artificial Intelligence in Design '96. Artificial Intelligence in Design '96. Springer, Dordrecht. pp. 151-170.
- [49] Hu, J.; Niu, H.; Carrasco, J.; Lennox, B.; Arvin, F., Voronoi-Based (2020). Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning. IEEE Transactions on Vehicular Technology.
- [50] Sarle, Warren (1994). Neural Networks and statistical models. CiteSeerX 10.1.1.27.699
- [51] Russell, Stuart; Norvig, Peter (2003) Artificial Intelligence: A Modern Approach (2da ed.). Prentice Hall.
- [52] Langley, Pat (2011). The changing science of machine learning. Machine Learning. 82 (3): 275-279.
- [53] Alpaydin, Ethem (2010). Introduction to Machine Learning. MIT Press. p. 9.
- [54] Cornell University Library (2001). Breiman: Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). Statistical Science. 16 (3).
- [55] Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani (2013). An Introduction to Statistical Learning. Springer. p. vii.

- [56] Extraída el 16 de Octubre de: https://en.wikipedia.org/wiki/Machine_learning#/media/File:AI_hierarchy.svg
- [57] Extraída el 16 de Octubre de: https://en.wikipedia.org/wiki/Machine_learning#/media/File:Fig-y_Part_of_ML_as_subfield_of_AI_or_AI_as_subfield_of_ML.jpg
- [58] Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer.
- [59] Extraída el 16 de Octubre de: https://en.wikipedia.org/wiki/Machine_learning#/media/File:Colored_neural_network.svg
- [60] Honglak Lee, Roger Grosse, Rajesh Ranganath, Andrew Y. Ng. (2009). Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations, Proceedings of the 26th Annual International Conference on Machine Learning.
- [61] Extraída el 18 de Octubre de: https://www.researchgate.net/figure/A-toy-example-for-evaluating-a-decision-tree_fig2_336849423
- [62] Extraída el 18 de Octubre de: https://en.wikipedia.org/wiki/Machine_learning#/media/File:Svm_max_sep_hyperplane_with_margin.png
- [63] Extraída el 18 de Octubre de: https://en.wikipedia.org/wiki/Machine_learning#/media/File:Linear_regression.svg
- [64] Extraída el 18 de Octubre de: https://en.wikipedia.org/wiki/Machine_learning#/media/File:SimpleBayesNetNodes.svg
- [65] LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). Deep Learning. Nature. 521 (7553): 436-444.
- [66] Ciresan, D.; Meier, U.; Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3642-3649.
- [67] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey (2012). ImageNet Classification with Deep Convolutional Neural Networks.

- [68] Google's AlphaGo AI wins three-match series against the world's best Go player. TechCrunch. 25 de Mayo 2017.
- [69] Bengio, Y.; Courville, A.; Vincent, P. (2013). Representation Learning: A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence. 35 (8): 1798-1828.
- [70] Schmidhuber, J. (2015). Deep Learning in Neural Networks: An Overview. Neural Networks. 61: 85-117.
- [71] A Guide to Deep Learning and Neural Networks, Extraído el 18 de Octubre de: <https://serokell.io/blog/deep-learning-and-neural-network-guide#components-of-neural-networks>
- [72] Szegedy, Christian; Toshev, Alexander; Erhan, Dumitru (2013). Deep neural networks for object detection. Advances in Neural Information Processing Systems: 2553-2561.
- [73] Bengio, Yoshua (2009). Learning Deep Architectures for AI". Foundations and Trends in Machine Learning. 2 (1): 1-127.
- [74] Rolnick, David; Tegmark, Max (2018). The power of deeper networks for expressing natural functions. International Conference on Learning Representations.
- [75] Hof, Robert D. "Is Artificial Intelligence Finally Coming into Its Own?". MIT Technology Review. Extraído el 18 de Octubre de 2022 de: <https://web.archive.org/web/20190331092832/https://www.technologyreview.com/s/513696/deep-learning/>
- [76] Gers, Felix A.; Schmidhuber, Jürgen (2001). LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages. IEEE Transactions on Neural Networks. 12 (6): 1333-1340.
- [77] ozefowicz, Rafal; Vinyals, Oriol; Schuster, Mike; Shazeer, Noam; Wu, Yonghui (2016). Exploring the Limits of Language Modeling.
- [78] Gillick, Dan; Brunk, Cliff; Vinyals, Oriol; Subramanya, Amarnag (2015). Multilingual Language Processing from Bytes.
- [79] Hochreiter, Sepp; Schmidhuber, Jürgen (1997). Long Short-Term Memory. Neural Computation. 9(8): 1735-1780.

- [80] Gers, Felix ; Schraudolph, Nicol (2002). Learning Precise Timing with LSTM Recurrent Networks. ResearchGate.
- [81] LeCun, Y.; et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 86 (11): 2278-2324.
- [82] Sainath, Tara N.; Mohamed, Abdel-Rahman; Kingsbury, Brian; Ramabhadran, Bhuvana (2013). Deep convolutional neural networks for LVCSR. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 8614-8618.
- [83] Varady, K. A. (2011). Intermittent versus daily calorie restriction: which diet regimen is more effective for weight loss?. *Obesity reviews*, 12(7), pp.593-601.
- [84] Sauberlich, H. E. (1985). Bioavailability of vitamins. *Progress in food and nutrition science*, 9(1-2), pp.1-33.
- [85] Meyers, A., Johnston, N., Rathod, V., Korattikara, A., Gorban, A., Silberman, N., Murphy, K. P. (2015). Im2Calories: towards an automated mobile vision food diary. In *Proceedings of the IEEE International Conference on Computer Vision* pp. 1233-1241.
- [86] Liang, Y., Li, J. (2017). Computer vision-based food calorie estimation: dataset, method, and experiment. *arXiv preprint arXiv:1705.07632*.
- [87] Mezgec, S., Korou I Seljak, B. (2017). NutriNet: a deep learning food and drink image recognition system for dietary assessment. *Nutrients*, 9(7), pp.657.
- [88] Wu, X., Fu, X., Liu, Y., Lim, E. P., Hoi, S. C., Sun, Q. (2021). A Large-Scale Benchmark for Food Image Segmentation. *arXiv preprint arXiv:2105.05409*.
- [89] T. Joutou; K. Yanai (2009), A food image recognition system with multiple kernel learning, en *Image Processing (ICIP)*, 2009 16th IEEE International Conference on, pp. 285-288, IEEE.
- [90] H. Hoashi, T. Joutou; K. Yanai (2010). Image recognition of 85 food categories by feature fusion, en *Multimedia (ISM)*, 2010 IEEE International Symposium en, pp. 296-301, IEEE.

- [91] M. Puri; Z. Zhu, Q. Yu; A. Divakaran; H. Sawhney (2009). Recognition and volume estimation of food intake using a mobile device, en Applications of Computer Vision (WACV), 2009 Workshop on, pp. 1-8, IEEE.
- [92] F. Kong and J. Tan (2012), Dietcam: Automatic dietary assessment with mobile camera phones, Pervasive and Mobile Computing, vol. 8, no. 1, pp. 147-163.
- [93] M. H. Rahmana; M. R. Pickering; D. Kerr; C. J. Boushey ; E. J. Delp (2012). A new texture feature for improved food recognition accuracy in a mobile phone based dietary assessment system, en Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference en, pp. 418-423, IEEE.
- [94] Y. Kawano; K. Yanai (2013). Real-time mobile food recognition system, en Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference en, pp. 1-7, IEEE.
- [95] Y. Kawano; K. Yanai (2013), Rapid mobile object recognition using fisher vector, en Pattern Recognition (ACPR), 2013 2nd IAPR Asian Conference on, pp. 476-480, IEEE.
- [96] F. Perronnin, J. Sánchez; T. Mensink (2010). Improving the fisher kernel for large-scale image classification, en Computer Vision, ECCV 2010, pp. 143-56, Springer.
- [97] Y. Kawano and K. Yanai (2015), Foodcam: A real-time food recognition system on a smartphone, Multimedia Tools and Applications, pp. 1-25.
- [98] M. Anthimopoulos; J. Dehais, P. Diem; S. Mougiakakou (2013). Segmentation and recognition of multi-food meal images for carbohydrate counting, en Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference en, pp. 1-4, IEEE.
- [99] P. Pouladzadeh; S. Shirmohammadi; T. Arici (2013). Intelligent svm based food intake measurement system, en Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CI-VEMSA), 2013 IEEE International Conference en, pp. 87-92, IEEE.

- [100] S. Patel. (2020). A Comprehensive Analysis of Convolutional Neural Network Models. *International Journal of Advanced Science and Technology*, 29(4), pp.771-777.
- [101] Kolsch Andreas; Afzal, Muhammad Zeshan; Liwicki Marcus. (2017). Multilevel Context Representation for Improving Object Recognition. 10.1109/ICDAR.2017.322.
- [102] Qassim H.; Verma A.; Feinzimer D. (2018). Compressed residual-VGG16 CNN model for big data places image recognition. En 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), pp. 169-175.
- [103] Niu Qiya, Teng Yunlai, Chen Lin. (2019). Design of gesture recognition system based on Deep Learning. *Journal of Physics: Conference Series*. 1168. 032082. 10.1088/1742-6596/1168/3/032082.
- [104] Dasiopoulou, Stamatia, et al. Knowledge-assisted semantic video object detection. *IEEE Transactions on Circuits and Systems for Video Technology* 15.10 (2005): 1210-1224.
- [105] Srivastava, Rupesh Kumar; Greff, Klaus; Schmidhuber, Jürgen (2015). Highway Networks. Extraído el 24 de Octubre de: [arXiv:1505.00387](https://arxiv.org/abs/1505.00387)
- [106] Srivastava, Rupesh K; Greff, Klaus; Schmidhuber, Juergen (2015). Training Very Deep Networks. *Advances in Neural Information Processing Systems* 28. Curran Associates, Inc. 28: 2377-2385.
- [107] Huang, Gao; Liu, Zhuang; Van Der Maaten, Laurens; Weinberger, Kilian Q. (2017). Densely Connected Convolutional Networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI: IEEE. pp. 2261-2269.
- [108] Extraído el 24 de Octubre de 2022 de: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md
- [109] Extraído de: <https://github.com/JorgeCabrero/FoodCompositionEstimation/blob/main/Image%20Collection.ipynb>
- [110] Extraído de: <https://github.com/nicknochnack/TFODCourse>

- [111] Extraído de: https://github.com/JorgeCabrero/FoodCompositionEstimation/blob/main/Training_and_Detection.ipynb
- [112] Extraído el 25 de Octubre de: <https://www.v7labs.com/blog/mean-average-precision>
- [113] Extraído de: [https://github.com/JorgeCabrero/FoodCompositionEstimation/blob/main/Experiments_Dashboard%20\(version%20\).xlsx](https://github.com/JorgeCabrero/FoodCompositionEstimation/blob/main/Experiments_Dashboard%20(version%20).xlsx)
- [114] Extraído de: [https://github.com/JorgeCabrero/FoodCompositionEstimation/blob/main/Experiments_Dashboard%20\(version%20\).xlsx](https://github.com/JorgeCabrero/FoodCompositionEstimation/blob/main/Experiments_Dashboard%20(version%20).xlsx)
- [115] Extraído de: https://github.com/JorgeCabrero/FoodCompositionEstimation/blob/main/Object_Detection%20with%20Cropping.ipynb
- [116] Hachcham, Aymane (2022). The KNN algorithm - Explanation, Opportunities, Limitations. extraído el 27 de Octubre de 2022 de: <https://neptune.ai/blog/knn-algorithm-explanation-opportunities-limitations>
- [117] Aha, David (2013). Lazy Learning. Springer Science & Business Media, 2013. p. 424.
- [118] Scott E Umbaugh (2005). Computer Imaging: Digital Image Analysis and Processing.
- [119] Canny, J. (1986). A Computational Approach To Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 8 (6): 679-714.
- [120] Shapiro, L. G.; Stockman, G. C. (2001). Computer Vision. pp. 137-150. Prentice Hall.
- [121] Extraído el 28 de Octubre de 2022 de: https://en.wikipedia.org/wiki/Gaussian_blur#/media/File:Cappadocia_Gaussian_Blur.svg
- [122] R.A. Haddad and A.N. Akansu (1991). A Class of Fast Gaussian Binomial Filters for Speech and Image Processing. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 39, pp 723-727.

- [123] Sobel, Irwin; Feldman G.(1968). A 3x3 Isotropic Gradient Operator for Image Processing. Presentado en el Stanford Artificial Intelligence Project (SAIL)
- [124] Duda,R. and Hart,P. (1973).Pattern Classification and Scene Analysis.John Wiley and Sons, pp. 271-2.
- [125] Extraído el 29 de Octubre de 2022 de: https://en.wikipedia.org/wiki/Sobel_operator#/media/File:Bikesgray.jpg
- [126] Extraído el 29 de Octubre de 2022 de: https://en.wikipedia.org/wiki/Sobel_operator#/media/File:Bikesgraysobel.jpg
- [127] Extraído el 29 de Octubre de 2022 de: https://en.wikipedia.org/wiki/Sobel_operator#/media/File:Bikesgraygv.jpg
- [128] Extraído el 29 de Octubre de 2022 de: https://en.wikipedia.org/wiki/Sobel_operator#/media/File:Bikesgraygh.jpg
- [129] Binmore, Ken; Davies, Joan (2007). Calculus Concepts and Methods. Cambridge University Press. p. 190
- [130] Lin, B., Sun, Y., Sanchez, J. E.; Qian, X. (2015). Efficient Vessel Feature Detection for Endoscopic Image Analysis. IEEE Transactions on Biomedical Engineering, 62(4), 1141-1150.
- [131] Extraído de: <https://github.com/JorgeCabrero/FoodCompositionEstimation/tree/main/Segmentation>
- [132] Extraído el 29 de Octubre de 2022 de: https://scikit-image.org/docs/stable/auto_examples/segmentation/plot_trainable_segmentation.html
- [133] Extraído el 29 de Octubre de 2022 de: https://github.com/scikit-image/scikit-image/blob/v0.19.2/skimage/feature/_basic_features.py#L100-L188
- [134] Extraído el 18 de Noviembre de 2022 de: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
- [135] Extraído de: https://github.com/JorgeCabrero/FoodCompositionEstimation/blob/main/Calories_Estimation.ipynb