Clasificación de alimentos con ResNet50

José Pablo Martínez Valdivia

25 de noviembre de 2024

Abstract

Convolutional Neural Networks (CNNs) have become the state-of-the-art approach for solving computer vision tasks, thanks to their ability to automatically learn relevant features from images. This study focuses on the classification of food images using a ResNet50 model. The dataset was curated and preprocessed to ensure balance and quality, while data augmentation was applied to increase variability. Two training approaches were compared: training the model from scratch and leveraging Transfer Learning with a pre-trained ResNet50 backbone.

1. Introducción

Este trabajo se centra en la clasificación de imágenes de alimentos por medio de redes neuronales convolucionales (CNN, por sus siglas en inglés). Utilizamos el modelo ResNet50, una arquitectura reconocida por ser pionera en el uso de conexiones residules, las cuales permitieron mitigar problemas en redes neuronales profundas.

El objetivo principal de este proyecto es comparar dos enfoques de entrenamiento: el entrenamiento desde cero y el uso de Transfer Learning con pesos preentrenados. A través de estos experimentos, se espera obtener una visión clara sobre las ventajas y limitaciones de cada método.

2. ETL

2.1. Extracción

El dataset utilizado como base fue "Food Image Classification", provisto por la plataforma Kaggle [1], el cual contenía inicialmente 23,837 fotos de 34 clases correspondientes a distintos platillos. Estas imágenes fueron obtenidas de varias páginas de Google Imágenes, según se menciona en la descripción del dataset.

El uso de todas las imágenes resultó problemático durante el entrenamiento debido a las limitaciones de memoria de la tarjeta gráfica empleada. Por esta razón, fue necesario reducir el dataset a 15 clases, con un máximo de 400 imágenes por clase. Las clases seleccionadas fueron:

- apple pie
- baked potato
- burger
- cheesecake
- crispy chicken
- donut
- fried rice
- fries
- hot dog
- ice cream
- omelette
- pizza
- sandwich
- sushi
- taco

Los directorios de estas clases fueron trasladados al nuevo dataset, y se realizó una selección manual de las imágenes de cada clase. Esto se llevó a cabo con el objetivo de eliminar aquellas imágenes que habían sido asignadas erróneamente a alguna categoría. Como resultado, el dataset final consistió en 5,481 imágenes.

Es importante destacar que el dataset original no contenía una cantidad balanceada de imágenes por clase. Por lo tanto, dependiendo de la disponibilidad de imágenes para cada categoría, se seleccionaron entre 300 y 400 imágenes por clase. Esto permitió balancear parcialmente la variabilidad del dataset y reducir los sesgos hacia clases específicas.

2.2. Aumentación de datos

Dado que las fotos fueron obtenidas de Google Imágenes, se observó que muchas de ellas provienen de productos cuyos fotógrafos tienden a capturarlas con una orientación similar. Para incrementar la variabilidad en estas clases, se aplicaron técnicas de aumentación de datos, como rotaciones y escalados aleatorios.

2.3. Transformación

Con el objetivo de evitar la necesidad de realizar transformaciones durante el entrenamiento, se decidió aplicar previamente todas las transformaciones requeridas y guardar los tensores resultantes en archivos serializados.

3. Marco Teórico

3.1. Capas convolucionales

La tarea de clasificación de imágenes requiere bloques convolucionales capaces de aprender características espaciales presentes en una imagen. Esto se logra mediante la convolución, una operación que consiste básicamente en una suma ponderada entre un arreglo multidimensional de tamaño fijo, conocido como kernel con los píxeles y canales de la imagen. Este kernel se centra sucesivamente en cada uno de los píxeles para realizar la operación de convolución (1).

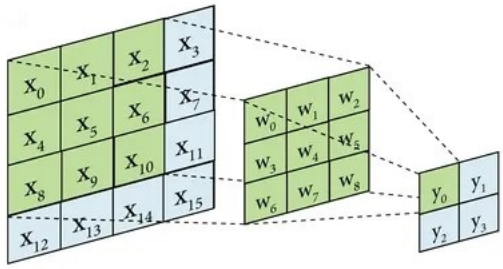

Figura 1: Operación de convolución [2].

La convolución permite generar mapas de activación, en los cuales los elementos del kernel actúan como pesos que activan o suprimen el resultado de la salida. En los modelos de visión, estos mapas inicialmente detectan características simples como bordes o colores. Sin embargo, después de varias capas convolucionales, el modelo es capaz de identificar patrones más complejos, como rostros u otros objetos específicos.

3.2. ResNet

El cálculo de la pérdida que se realiza en la última capa de una red neuronal durante el entrenamiento se propaga hacia atrás multiplicándose por las derivadas de las capas de activación y los pesos de la red para calcular el gradiente y ajustar estos parámetros. En redes profundas, este proceso puede causar que el gradiente se reduzca o aumente excesivamente a través de las capas, lo que da lugar al problema de vanishing o exploding gradients.

La arquitectura ResNet introduce un bloque residual (2) que mitiga este problema. Este bloque incorpora una conexión denominada skip, que consiste en sumar la salida de un bloque de capas convolucionales con el tensor de entrada. Esto permite que el gradiente fluya libremente hacia capas anteriores durante el ajuste de pesos, reduciendo significativamente el riesgo de pérdida o aumento excesivo del gradiente.

Cada bloque residual reduce las dimensiones de ancho y alto de la imagen, mientras incrementa el número de canales. Para lograr esto, es necesario un bloque de

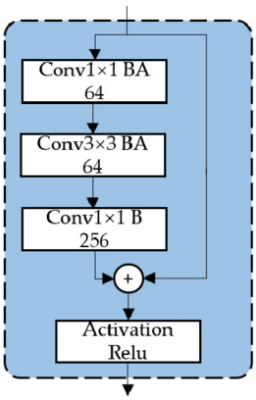

Figura 2: Conexión skip [3].

proyección (3), que emplea una convolución de 1x1 en lugar de la conexión skip, igualando los canales de entrada con los de salida.

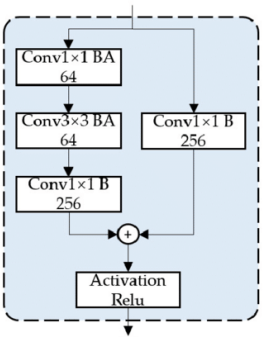

Figura 3: Bloque de proyección [3].

La arquitectura seleccionada para la clasificación fue ResNet50 (5), que cuenta con 50 capas convolucionales en su backbone para la extracción de características de las imágenes. Esta red finaliza con capas densas (3.2) que generan una salida correspondiente a cada una de las clases.

Capa	Salida	Details
Flatten	(2048)	-
Dense	(256)	ReLU Activation
Dense	(*NUM CLASES*)	-

Cuadro 1: Clasificador

4. Modelado

4.1. Entrenamiento con pesos aleatorios

El dataset fue dividido en un 80 % para datos de entrenamiento y un 20 % para validación. Se utilizó el optimizador *Adam* para ajustar los pesos y la función de pérdida *Entropía cruzada* Además, se implementó

un programador *scheduler* para reducir progresivamente la tasa de aprendizaje cada cierto intervalo de tiempo *step_size* en proporción al parámetro *gamma*, lo cual ayuda a estabilizar el ajuste de los pesos.

Asimismo, se utilizó la técnica de early stopping para finalizar el entrenamiento de forma anticipada si la pérdida no mejoraba después de un número máximo de épocas consecutivas. El modelo fue entrenado con los siguientes hiperparámetros:

- taza de aprendizaje: 0.01
- step_size (scheduler): 10
- gamma (scheduler): 0.5
- paciencia 30
- épocas máximas: 100
- tamaño de lote: 32

4.2. Evaluación con pesos aleatorios

El modelo finalizó el entrenamiento de forma anticipada en la época 40. Los pesos con el mejor desempeño en el conjunto de validación lograron una pérdida de 1.4032 y una precisión del 55.54 %.

En las gráficas de entrenamiento (4) se observa un claro sobreajuste del modelo, evidenciado por la diferencia significativa entre la precisión en el conjunto de entrenamiento y el de validación. Asimismo, al analizar la matriz de confusión (6), se aprecia que el modelo tiene dificultades para diferenciar ciertas clases.

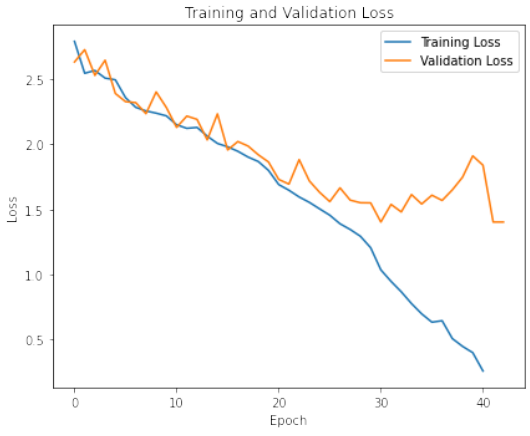

Figura 4: Pérdida modelo 1.

Este desempeño limitado puede atribuirse a la variabilidad del dataset y las características de las clases. Es posible que el modelo encuentre dificultades para identificar y generalizar patrones debido al dominio específico del dataset y al reducido número de imágenes y cantidad de clases.

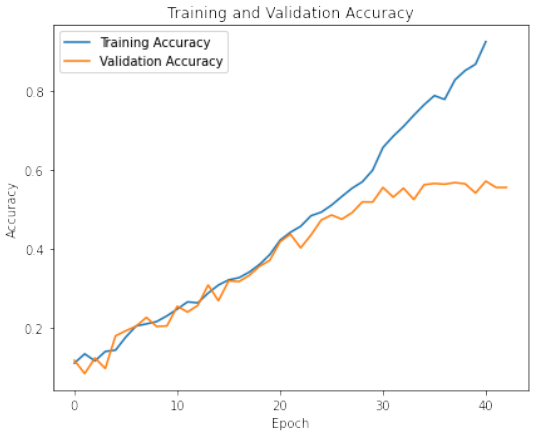

Figura 5: Precisión modelo 1.

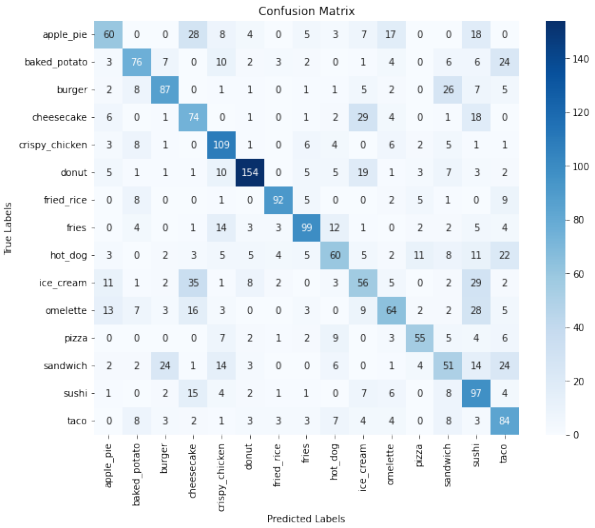

Figura 6: Matriz de confusión modelo 1.

4.3. Entrenamiento con Transfer Learning

Se utilizó la arquitectura ResNet50 previamente entrenada con el conjunto de datos ImageNet. Los pesos del backbone fueron congelados, de manera que únicamente se entrenó el clasificador (3.2). El entrenamiento se llevó a cabo utilizando los siguientes hiperparámetros:

- taza de aprendizaje: 0.001
- step_size (scheduler): 10
- gamma (scheduler): 0.5
- paciencia 10
- épocas máximas: 50
- tamaño de lote: 32

4.4. Evaluación del Transfer Learning

El modelo finalizó de forma anticipada en la época 21, logrando una pérdida de 0.2989 y una precisión del 92.29 %. Este resultado refleja una capacidad predictiva sobresaliente en el conjunto de datos de validación y una mejora significativa en comparación con el modelo sin pesos preentrenados.

Además, la matriz de confusión (9) muestra una reducción notable en la confusión entre las clases, lo que evidencia que el modelo puede diferenciar mejor entre las categorías presentes en el conjunto de datos de validación.

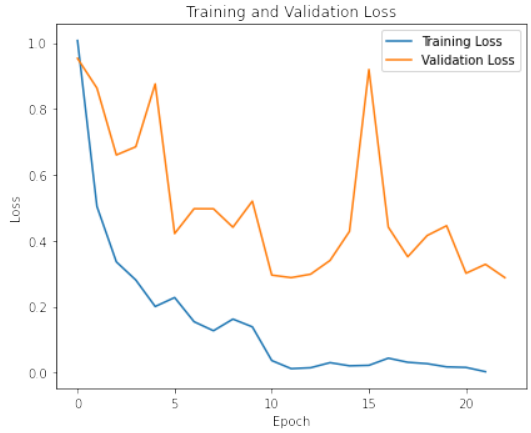

Figura 7: Pérdida modelo 2.

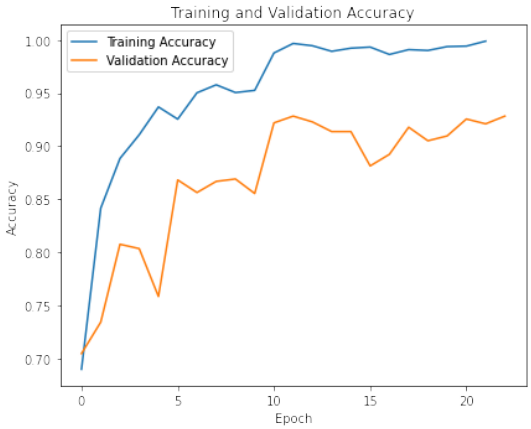

Figura 8: Precisión modelo 2.

5. Conclusión

Las redes neuronales convolucionales representan el estado del arte en tareas de visión computacional, gracias a su capacidad para extraer características espaciales en imágenes. Este estudio demuestra la importancia de utilizar un backbone preentrenado, como ResNet50, para mejorar el desempeño en tareas de clasificación de imágenes, especialmente cuando se trabaja con conjuntos de datos limitados o específicos.

El uso de Transfer Learning permitió alcanzar una precisión significativamente mayor en comparación con el entrenamiento desde cero, resaltando la ventaja de aprovechar conocimientos previamente adquiridos por modelos más grandes. Esto demuestra la importanci de contar con características bien aprendidas en las primeras capas del modelo para abordar problemas de clasificación en imágenes.

Referencias

[1] Food Image Classification Dataset. (2023, 4 julio). Kaggle. https://www.kaggle.com/datasets/harishkumardatalab/food-image-classification-dataset?select=Food+Classification+dataset

[2] Sahoo, S. (2024, 22 enero). 2D Convolution using Python & NumPy - Analytics Vidhya - Medium. Medium. https://medium.com/analytics-vidhya/2d-convolution-using-python-numpy-43442ff5f381

[3] Wang, S., Xia, X., Ye, L., & Yang, B. (2021). Automatic detection and classification of steel surface defect using deep convolutional neural networks. Metals, 11(3), 388. https://doi.org/10.3390/met11030388

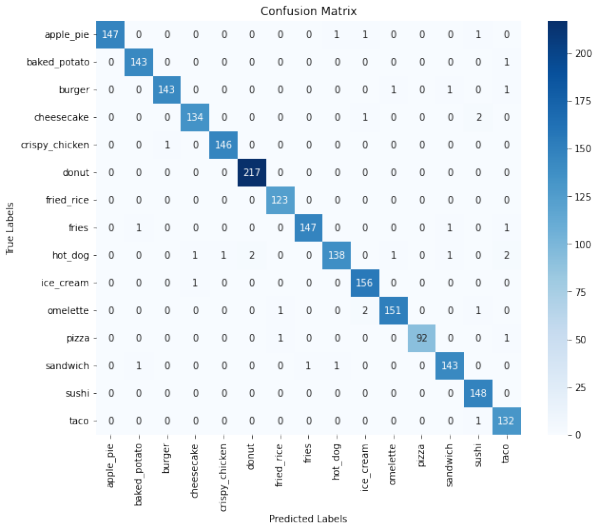

Figura 9: Matriz de confusión modelo 2.

Capa	**Salida**	**Detalles**
Input	(224, 224, 3)	Imagen RGB
Conv2D	(112, 112, 64)	7×7, stride=2
BatchNorm2D	(112, 112, 64)	-
ReLU	(112, 112, 64)	-
MaxPooling2D	(56, 56, 64)	2×2, stride=2
Residual (Capa 1)	(56, 56, 256)	3 capas
Residual (Capa 2)	(28, 28, 512)	4 capas
Residual (Capa 3)	(14, 14, 1024)	6 capas
Residual (Capa 4)	(7, 7, 2048)	3 capas
Global Avg Pooling	(1, 1, 2048)	Pooling adaptativo

Cuadro 2: Backbone ResNet50

Capa	**Salida**	**Details**
Conv2D	(H, W, inChannels)	1×1, stride=1
BatchNorm2D	(H, W, inChannels)	-
ReLU	(H, W, inChannels)	-
Conv2D	(H, W, inChannels)	3×3, stride=1
BatchNorm2D	(H, W, inChannels)	-
ReLU	(H, W, inChannels)	-
Conv2D	(H, W, outChannels)	1×1, stride=1
BatchNorm2D	(H, W, outChannels)	-
Residual Connection	(H, W, outChannels)	Identidad o proyección
ReLU	(H, W, outChannels)	-

Cuadro 3: Bloque Residual ResNet50